МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) Автоматизації і комп'ютерних систем
Кафедра Інформаційних систем

«До захисту в ЕК»		**«До захисту допущено»**	
Директор інституту(декан факультету)		Завідувач кафедри	
_____	Андрій Форсюк	_____	Сергій Чумаченко
(підпис)	(ім'я та прізвище)	(підпис)	(ім'я та прізвище)
«___» _____ 2022 р.		«___» _____ 2022 р.	

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

зі спеціальності 122 «Комп'ютерні науки»
(код та назва спеціальності)
освітньо-професійної програми Комп'ютерні науки

на тему: Розроблення інформаційної підтримки діяльності футбольної академії

Виконав: здобувач 4 курсу, групи 2

Левченко Олександр Олександрович _____
(прізвище, ім'я, по батькові повністю) (підпис)

Керівник Мазуренко Ольга Олександрівна _____
(прізвище , ім'я та по батькові повністю) (підпис)

Консультанти_____ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

Рецензент Ярослав Смітюх _____
(ім'я та прізвище) (підпис)

Я як здобувач(ка) Національного університету харчових технологій розумію і підтримую політику університету з академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволеної допомоги під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Здобувач_____
(підпис)

Київ – 2022 р.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) Автоматизації і комп'ютерних систем
Кафедра Інформаційних систем
Освітній ступінь бакалавр
Спеціальність 122 «Комп'ютерні науки»
(код і назва)
Освітньо-професійна програма «Комп'ютерні науки»
(назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри Чумаченко С.М.

"____" _____ 2022 року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

Левченко Олександр Олександрович
(прізвище, ім'я, по батькові)

1. Тема роботи: Розроблення інформаційної підтримки діяльності футбольної академії
керівник роботи Мазуренко Ольга Олександрівна,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від "31 " 03 2022 року №163-кс
2. Строк подання здобувачем роботи 06.06.2022
3. Вихідні дані до роботи *Дані про організаційну структуру академії ФК «Прикарпаття», використане системне програмне забезпечення та нормативні документи академії.*
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Системний аналіз діяльності академії «ФК Прикарпаття», задача автоматизації та їх вирішення.
5. Перелік графічного матеріалу
1. Організаційна структура академії ФК «Прикарпаття»
2. Контекстна діаграма функціональної моделі та діаграми декомпозиції
3.Скріншоти інтерфейсу системи

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Вступ			
I	Мазуренко Ольга Олександрівна	01.04.2022	06.06.2022
II	Мазуренко Ольга Олександрівна	01.04.2022	06.06.2022
Висновок			

7. Дата видачі завдання___01.04.2022_____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів виконання кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Системний аналіз діяльності ФК «Прикарпаття»	**15.03.2022 – 26.03.2022**	Виконано
2	Розробка моделі даних та створення бази даних	**07.04.2022 – 16.04.2022**	Виконано
3	Визначення та реалізація функції інформаційної системи	**19.04.2022 – 04.05.2022**	Виконано
4	Оформлення пояснювальної записки	**18.05.2022 – 30.05.2022**	Виконано
5	Розробка презентації	**30.05.2022 – 31.05.2022**	Виконано
6			
7			
8			
9			

Здобувач _____ _____Левченко О. О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____Мазуренко О.О.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Пояснювальна записка містить: 67 с. тексту, 38 рисунків, 12 таблиць, посилання на 21 літературне джерело, 6 додатків.

Кваліфікаційна робота присвячена питанню автоматизації діяльності футбольної академії.

Об'єктом дослідження є діяльність футбольної академії «Прикарпаття».

Предметом дослідження є методи та засоби діяльність футбольної академії.

Метою роботи є розробка автоматизованої інформаційної системи діяльності футбольної академії.

Дана робота складається з вступу, двох розділів, висновків, списку літератури та додатків.

У вступі представлено актуальність теми, визначено мету та завдання, об'єкт та предмет дослідження.

У розділі «Системний аналіз об’єкта дослідження та виявлення задач автоматизації» було проведено техніко-економічну характеристику діяльність футбольної академії, здійснено концептуальне моделювання та виконано постановку завдання, а також проведено аналіз існуючих розробок та обґрунтовано вибір технології проектування.

У другому розділі виконано логічне та фізичне моделювання інформаційної системи, описано технологічне забезпечення завдання, а також розглянуто контрольний приклад реалізації проекту.

У висновку представлені висновки щодо виконаної роботи.

Результатом роботи є розроблена автоматизована інформаційна система діяльності футбольної академії.

Ключові слова: ЗВІТ, БАЗА ДАНИХ, ПРЕДМЕТНА ОБЛАСТЬ, ФУТБОЛЬНА АКАДЕМІЯ, ДІАГРАМА, ПРОГРАМА, ІНТЕРФЕЙС, СЕРЕДОВИЩЕ РОЗРОБКИ.

ANNOTATION

The explanatory note contains: 67 p., 38 pictures, 12 tables, 21 references, and 6 appendices.

Qualification work is devoted to the issue of automation of football academy.

The object of research is the activity of the football academy "Prykarpattya."

The subject of research is the methods and means of the football academy.

The work aims to develop an automated information system for the football academy.

This work consists of an introduction, two chapters, conclusions, a bibliography, and appendices.

The introduction presents the relevance of the topic and defines the purpose and objectives, object, and subject of research.

In the section "System analysis of the object of research and identification of automation problems," the technical and economic characteristics of the football academy were performed, conceptual modeling and task sets were performed, as well as the analysis of existing developments and the choice of design technology.

In the second section, logical and physical modeling of the information system is performed, the technological support of the task is described, and a control example of project implementation is considered.

The conclusion presents conclusions about the work done.

The result is an automated information system for the football academy.

List of keywords: REPORT, DATABASE, SUBJECT AREA, FOOTBALL ACADEMY, DIAGRAM, PROGRAM, INTERFACE, DEVELOPMENT ENVIRONMENT.

Зміст

ВСТУП

Один із найшвидших шляхів для гравця пробитися до професійного футбольного клубу — це вступити до футбольної академії.

Футбольна академія – програма навчання для молодих та талановитих гравців, як правило, тривалістю від 6 місяців до кількох років. Спортсмени проживають у кампусі академії (якщо приїхали з іншого міста), тренуються, одночасно навчаються за шкільною програмою. Найчастіше спортсмени відбираються в академії при локальних футбольних клубах. Наочний приклад, футбольна академія при ФК «Прикарпаття» (Івано-Франківськ), ФК «Аякс» (Нідерланди) тощо.

Хороша програма навчання передбачає інтенсивні футбольні тренування, тренування у тренажерному залі, теоретичні заняття з відео-аналізом матчів, товариські ігри та офіційні матчі. Крім того, проводяться уроки та майстер-класи з дієти та правильного харчування, відновлення, профілактики травм, спортивної психології.

Чим краща футбольна академія, тим сильніший у ній колектив тренерів. Як правило, у багатьох міжнародних футбольних академіях тренери мають ліцензію УЄФА категорії А чи ПРО, а також значний досвід тренерської діяльності молодіжних чи дорослих команд.

Провідні академії також завжди фокусуються і на академічному навчанні спортсменів, щоб вони успішно закінчували свою шкільну або університетську освіту. Зокрема, практично завжди пропонують кілька опцій навчання – навчання у місцевій школі, віддалене навчання через інтернет (наприклад, за британською чи американською системою), навчання з репетиторами тощо.

Таким чином, основними відмінними рисами хорошої футбольної академії є:

– Мета академії: підготувати спортсмена для гри на напівпрофесійному (університетський рівень) чи професійному рівні (провідні ліги).

– Високий рівень гри студентів. Суворий відбір в академії. Перед зарахуванням часто запитуються відео-резюме та нарізки гри спортсмена.

– Тривалість від 6 місяців до 2 років.

– Тренувальний процес: як правило, власна розроблена методика навчання або запозичена у клубу, при якому працює академія. Покриваються усі аспекти гри.

– Фокус на академічну успішність за шкільною (університетською програмою). Успішність учнів – пріоритет.

– Тренери: ліцензії УЄФА А та ПРО. Рідше за ліцензію УЄФА Б.

– Вік спортсменів: від 12 до 24 років.

Об'єктом дослідження є діяльність футбольної академії «Прикарпаття».

Предметом дослідження є методи та засоби діяльність футбольної академії.

Метод дослідження – CASE-технології структурного аналізу та проектування.

Метою даної є розробка автоматизованої системи для організації діяльності футбольної академії.

Для досягнення цієї мети у роботі необхідно вирішити такі завдання:

– провести аналіз предметної галузі;

– провести аналіз існуючої технології автоматизації академії;

– описати існуючу технологію діяльності футбольної академії;

– спроектувати майбутню модель бізнес-процесів футбольної академії;

– розробити програмне забезпечення з урахуванням усунення виявлених під час аналізу предметної області недоліків.

Розділ 1. Системний аналіз об’єкта дослідження та виявлення задач автоматизації

1.1. Загальна характеристика футбольної академії

Футбольна академії «Прикарпаття» готує більше ніж 100 футболістів у п'яти вікових категоріях – U-13, U-14, U-15, U-16, U-17, які працюють під наставництвом досвідчених та кваліфікованих тренерів.

Академія футболу – високоефективна програма, спрямована на розвиток індивідуального потенціалу молодих спортсменів. Вона працює цілий рік, дозволяючи вихованцям не втрачати навичок та брати участь у спортивних заходах упродовж навіть міжсезоння у професійному футболі.

Всі професійні академії мають свою інфраструктуру, систему управління, базу спортсменів із їх особистими здобутками, а також сформовані команди для змагань.

Завданнями академії ФК «Прикарпаття» є:

– Залучення спортсменів у систематичні заняття футболом, залучення їх до спортивного життя та створення мотивації та інтересу до спортивних перемог.

– Створення сприятливих умов для тренувальних процесів, і навіть ефективної організації роботи зі спортсменами.

– Забезпечення участі спортсменів у різних змаганнях.

– Безперервне відстеження прогресу кожного зі спортсменів.

Академія здійснює такі види діяльності:

– Організація заходів, спрямованих на популяризацію спорту, включаючи лекції та змагання.

– Організація та контроль за проведенням тренувального процесу в галузі юнацького спорту.

– Реєстрація та облік спортивних досягнень спортсменів.

– Формування та підтримка діяльності спортивних команд.

– Ведення обліку проведених занять у тренерів.

Послуги надаються спортсменам на платній основі у формі спортивного абонементу згідно із затвердженим розкладом занять групи.

Спортивний абонемент може включати різні послуги, або спортсмен може придбати кілька спортивних абонементів для отримання кількох видів послуг. Футболісти у групах, залежно від практики академії можуть призупиняти дію своїх абонементів.

Крім групових занять в академії можуть проводитися індивідуальні, а також існує тренувальний режим, коли спортсмен тренується у вільний час.

Розклад занять для груп в академії створюється та затверджується адміністратором.

Футбольна академія орендує кілька тренувальне приміщення. В ній практикується прийом оплати після відвідування академії – зокрема після відвідування академії дітьми, оплату їх тренувань здійснюють батьки.

1.2 Організаційна структура футбольної академії, роль і взаємодія підрозділів

Організаційна структура футбольної академії «Прикарпаття» виглядає так як показано на рис.1.1. На чолі футбольної академії знаходиться директор. Бухгалтер, який займається фінансами. Менеджер, який займається рекламою та персоналом, у кожного менеджера свої підлеглі – адміністратор філії академії, бухгалтер, обслуговуючий персонал та медичний працівник. У підпорядкуванні у директора клубу знаходиться головний тренер, який, у свою чергу, підпорядковує тренерів всієї академії. Тренери не прив'язані до менеджерів, а підпорядковуються головному тренеру.

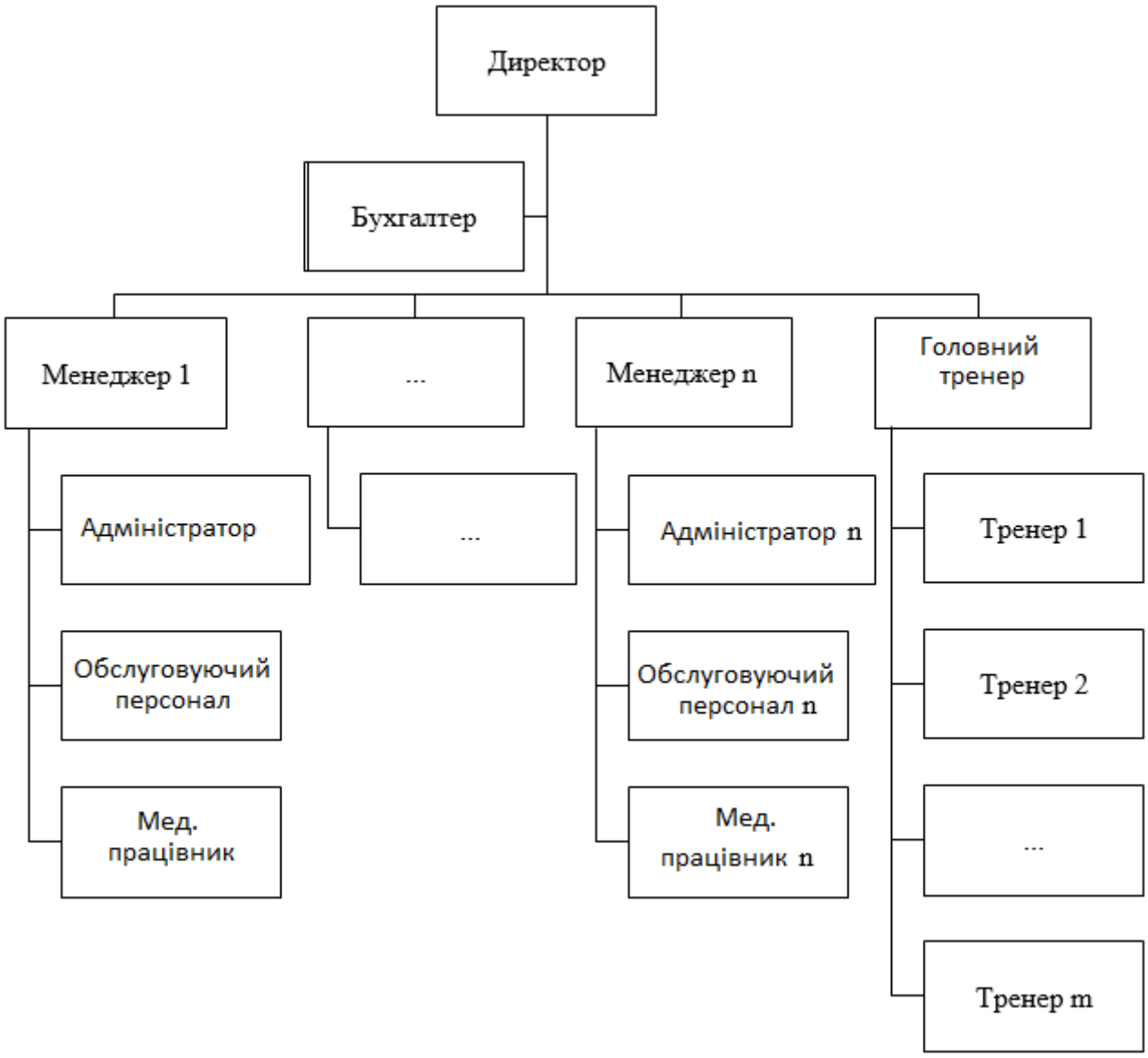

Рисунок 1.1 – Організаційна структура футбольної академії

Основні ролі в організації діяльності футбольної академії «Прикарпаття»:

– адміністратор ФА;
– тренер.

Адміністратор футбольної академії – персонал, який працює зі спортсменами поза тренувальним процесом, забезпечує роботу з обслуговування спортсменів та створення для них комфортних умов.

Основні функції адміністратора футбольної академії:

– підключення абонементів гравцям академії;

– ведення клієнтської бази;

– реєстрація відвідувань академії спортсменами;

– видача ключів від кабінок;

– запис на заняття;

– формування розкладу занять.

Тренер – персонал, який працює зі спортсменами під час тренувального процесу, забезпечує спортсмена необхідними спортивними навантаженнями та мотивує до нових досягнень. До основних обов’язків тренера входить:

– повідомлення спортсмену своїх знань та умінь, стимулювання його пізнавальної активності;

– різнобічна підготовка спортсмена, а також формування вольової готовності спортсмена до виступу на різноманітних змаганнях;

– оптимізація тренувального процесу та його вдосконалення в умовах регулярного контролю за спортсменами;

– вироблення у спортсменів морально-вольових якостей, певних рис характеру та самодисципліни;

– оцінка результатів спортсменів, показані на тренуваннях та змаганнях.

1.3 Стан автоматизації футбольної академії

У футбольній академії «Прикарпаття» немає відповідного програмного забезпечення для обліку спортсменів, складання документації, виведення звітів та статистики. Вся інформація про діяльність академії зберігається у різних місцях. Ведеться облік таких документів:

– Абонементи для спортсменів (інформація про абонементи: назва, опис, ціна, дата початку дата закінчення);

– База спортсменів (інформація про спортсменів: прізвище, ім’я, номер телефону, дата народження, адреса проживання та електронна адреса);

– База тренерів (інформація про тренерів: прізвище, ім’я, номер телефону, дата народження, адреса проживання та електронна адреса);

– розклади тренувань (інформація про розклади занять: назва заняття, дата заняття, година початку заняття та година закінчення заняття).

Запис інформації про спортсменів та тренерів здійснюється ручним способом у журналі. А для обліку та складання звітів використовується програма Microsoft Excel. Але при великому потоці спортсменів, збільшився і час виконання.

1.4 Розроблення функціональної моделі та аналіз існуючих бізнес-процесів

Футбольна академія працює відповідно до законів та положень, розроблених керівництвом самої академії. Ці обмеження діють протягом усієї її діяльності. Це діяльність футбольної академії пов'язана з роботою із спортсменами.

На вході процесу діяльність футбольної академії є: дані спортсмена, список тренувань, дані про тренера, дані про академію, запит спортсмена, список та опис вправ, список інвентаря. На виході процесу договір про надання послуг, оформлення пакету послуг, список груп та індивідуальних занять, список оформлених тренерів, розклад тренувань, журнал відвідування спортсмена, звіт по приміщенню та абонемент – те що отирмує спортсмен при запиті. Механізм, яким керується процес: персонал та обладнання.

Організація діяльності футбольної академії зображена на рис. 1.2 у вигляді контекстної діаграми IDEF0.

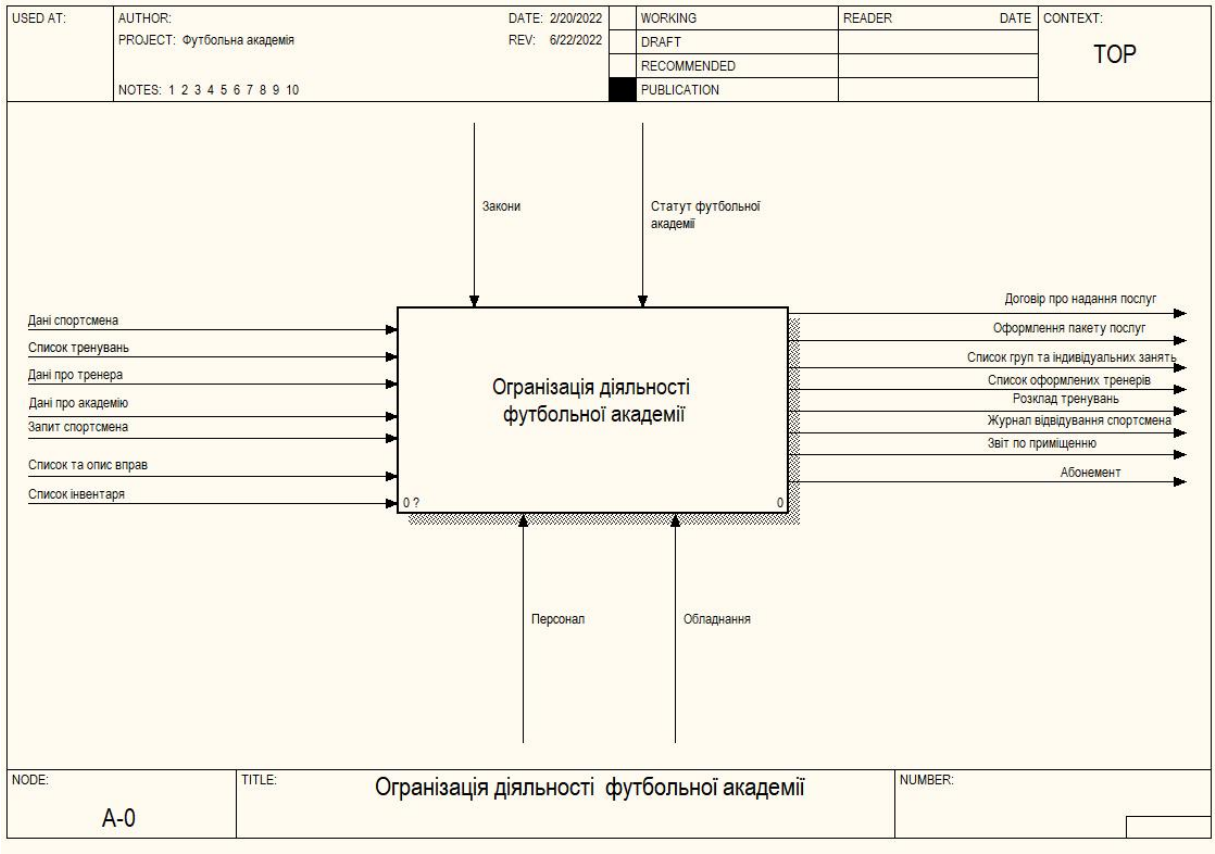

Рисунок 1.2 – Контекстна діаграма діяльності IDEF0 футбольної академії

Декомпозиція діаграми розбиває процес організація діяльності футбольної академії на підпроцеси, що відображають її детальний опис (рис. 1.3).

Підпроцес «Оформлення абонементу» має на вході три об'єкти: дані спортсмена, список тренувань та дані про академію; на виході – договір про надання послуг, оформлення пакету послуг і список груп та індивідуальних занять.

Підпроцес «Оформлення тренерів» має на вході об'єкти: дані про тренера, дані про академію, список тренувань. На виході – список оформлених тренерів.

Підпроцес «Складання розкладу по академії» на вході має запит спортсмена, а також список тренувань; на виході – розклад тренувань та абонемент.

Підпроцес «Підготовка академії до проведення занять» на вході має: список тренувань, список та опис вправ та список інвентаря. На виході формуються: журнал відвідування спортсмена та звіт по приміщенню.

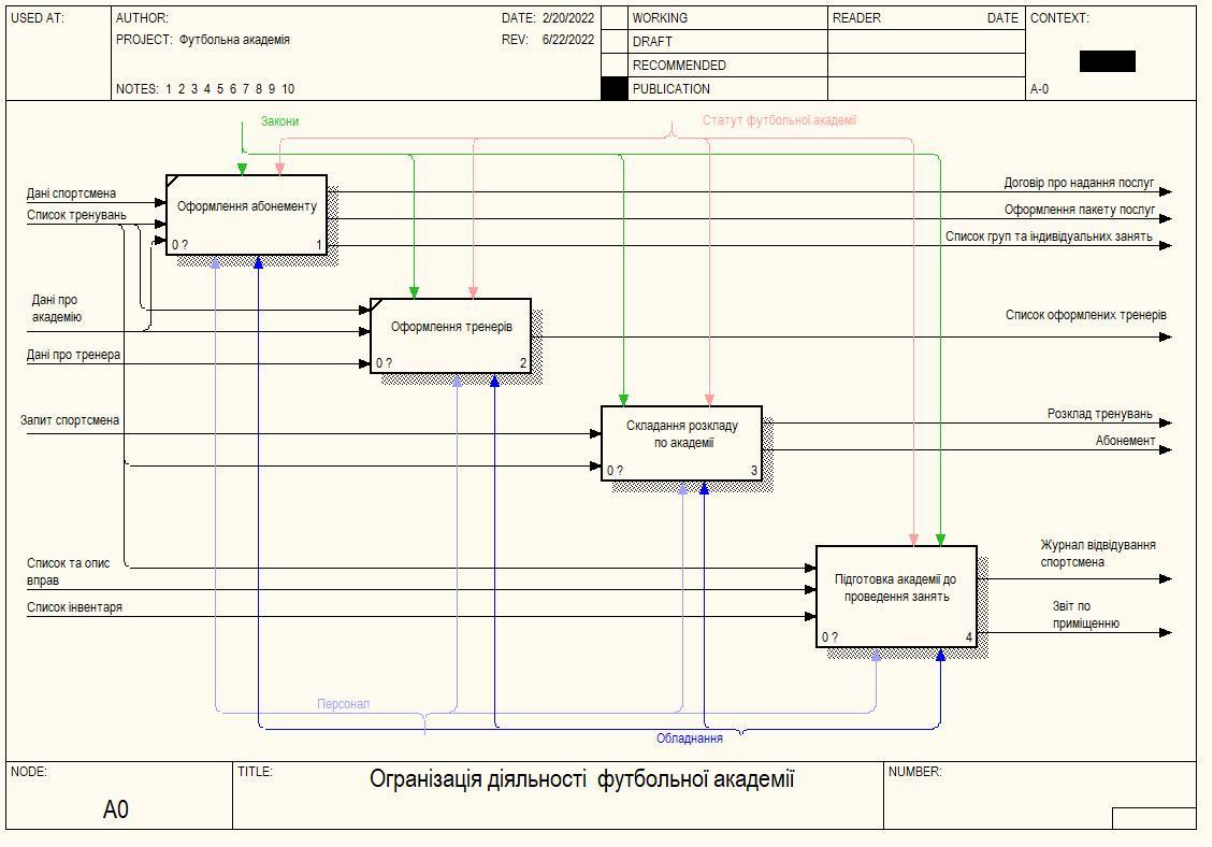

Рисунок 1.3 – Організація діяльності футбольної академії

Далі проведено декомпозицію підпроцесу «Складання розкладу по академії» (рис. 1.4), який в свою чергу складається з чотирьох підпроцесів.

Підпроцес «Складання розкладу» на вхід якого подаються: список тренувань. На виході – розклад.

Підпроцес «Продаж абонементу» отримує на вході об’єкт запит спортсмена. А на виході ми отримуємо об’єкт абонемент.

Підпроцес «Запис на тренування» має на вході запит спортсмена та розклад заняття. Проведення заняття та підготовку до нього проводять адміністратор та обслуговуючий персонал. На виході підпроцесу буде сформований список груп.

Підпроцес «Внесення інформації до БД» має на вході об’єкти: список груп та розклад. Механізмами упрівляння даного процесу є: тренер та обладнання. На виході отримаємо розклад тренувань.

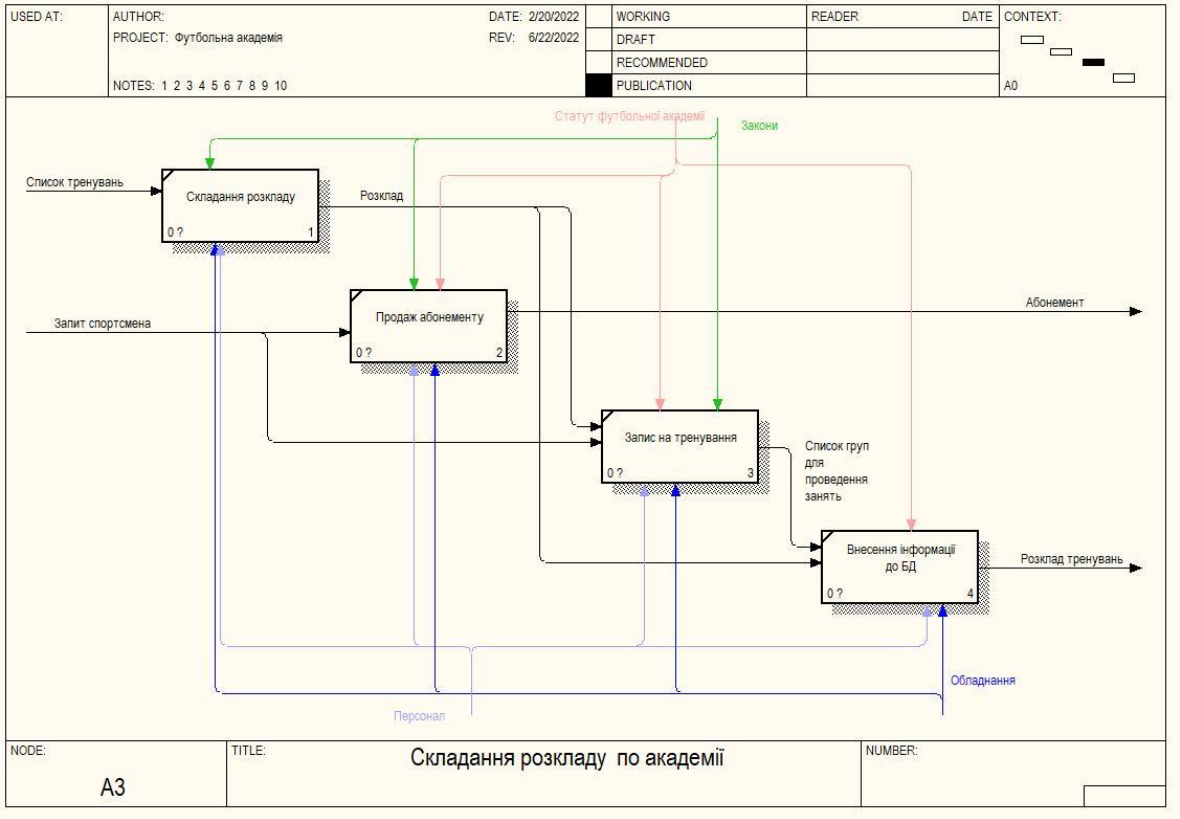

Рисунок 1.4 – Декомпозиція підпроцесу «Складання розкладу по академії»

Після цього необхідно провести декомпозицію підпроцесу «Підготовка академії до проведення занять» (рис.1.5), що в свою чергу складається із чотирьох підпроцесів.

Підпроцес «Підготовка академії до проведення тренування» має на вході список та опис вправ та список тренувань, на виході – види тренувань.

Підпроцес «Видача інвентаря» має на вході список інвентаря, а на виході об’єкт – список виданого інвентаря.

До підпроцесу «Тренувальний процес» на вхід подаються: список виданого інвентаря, на виході – список поверненого інвентаря та журнал відвідування спортсмена. Також, межанізмом управління даного підпроцесу буде обладнання, що буде використовуватися під час проведення тренувального процесу.

Останній підпроцес «Збір інвентаря» буде мати на вході список поверненого інвентаря, а на виході – звіт по приміщенню.

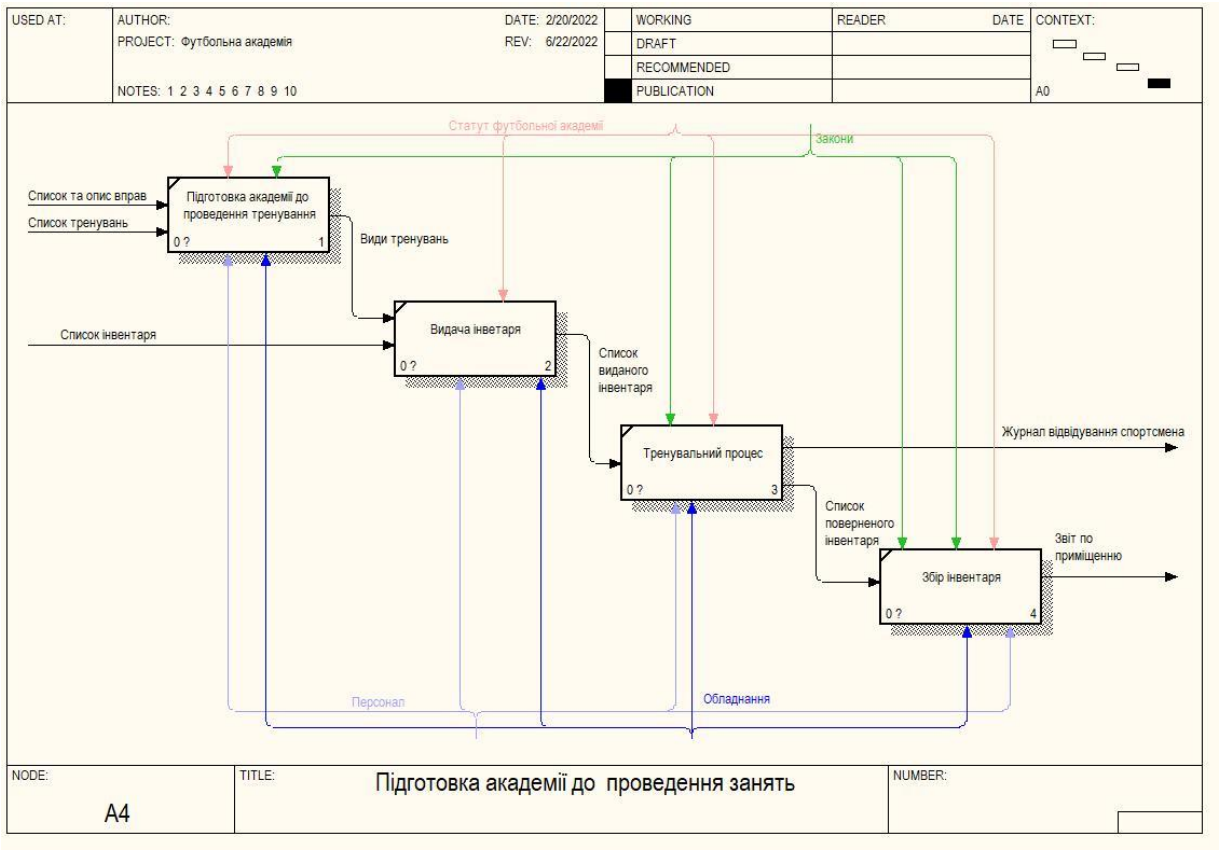

Рисунок 1.5 – Декомпозиція підпроцесу «Підготовка академії до проведення занять»

Отже, в даному підрозділі було описано процес діяльності футбольної академії у вигляді функціональної моделі контекстної діаграми IDEF.

1.5 Огляд існуючих рішень для розв’язання виявлених проблем

На сьогоднішній день електронно-обчислювальні машини є невід'ємною частиною життя суспільства, і вже складно уявити функціонування будь-якої організації без використання інформаційних систем та автоматизації частини чи всіх функціональних процесів. Серед особливо трудомістких можна назвати завдання ведення бухгалтерського та податкового обліку, консолідації, роботи з кадрами, обліку платежів, документообігу тощо. Однак на даному етапі для задоволення потреб сучасних організацій недостатньо автоматизації лише низки функціональних блоків. Потрібний комплексний продукт, що враховує специфіку конкретного виду діяльності.

Футбольні академії та інші організації, які здійснюють фізкультурно-оздоровчу діяльність, останнім часом стають дедалі популярнішими. У зв'язку з цим зріс попит на інформаційні системи, що дозволяють вирішувати як типові, а й специфічні завдання.

На ринку представлений ряд програмних продуктів, покликаних виконувати облік завдань, притаманних даного виду діяльності.

Для аналізу було обрано такі інформаційні системи: YOPLAYDO та ProTrainUp. Ці програмні продукти позиціонуються як програмне забезпечення для автоматизації діяльності футбольної академії.

1.5.1 YoPlayDo

YoPlayDo - футбольна платформа для обліку даних та статистики гравців. На основі історії даних YoPlayDo дозволяє тренеру оцінити прогрес і рівень розвитку гравців [3]. Інтерфейс для роботи з платформою представлений на рис. 1.6.

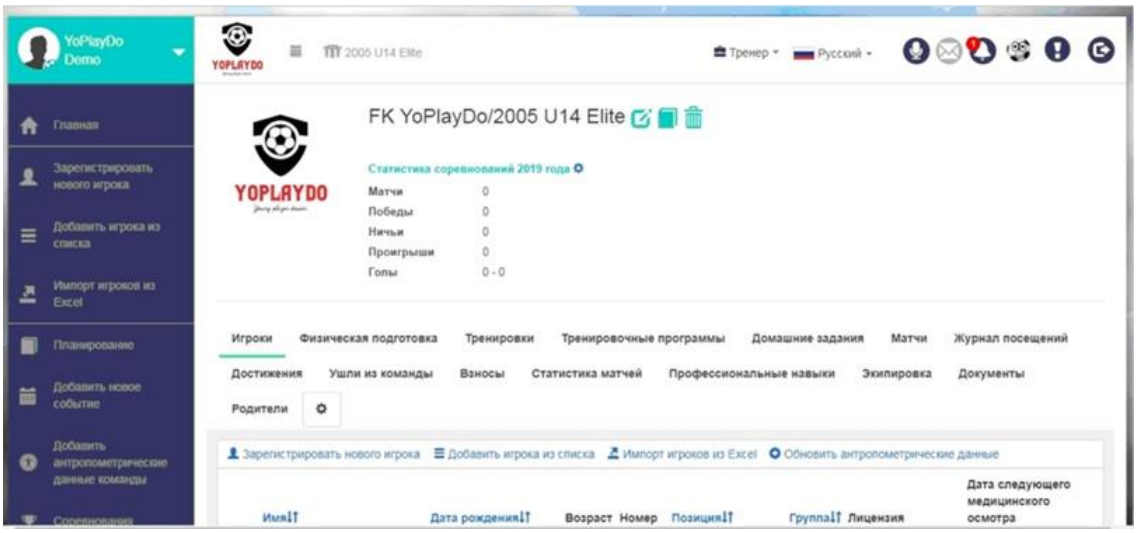

Рисунок 1.6 – Інтерфейс платформи «YoPlayDo»

YoPlayDo є високотехнологічним рішенням, яке виводить керування даними та статистикою равців на новий рівень.

Найбільша цінність даної платформи, — це можливість здійснення аналітики гравців та формування досьє для футболістів. Далеко не у всіх клубах

ведеться пильний аналіз розвитку гравців дитячо-юнацької ліги. Працюючи цілий день на полі, досить складно об'єктивно уявляти, як розвивається той чи інший гравець. Постійна зміна ваги, зростання, розвиток умінь, зміна поведінки на полі через психологічний розвиток та ін. Щоб оцінити перспективу гравця, необхідно оцінювати гравця в динаміці.

Команда YoPlayDo зосереджена на тому, щоб у простому форматі надати тренеру необхідні функції для оцінки розвитку гравця. Рутина роботи з програмою виглядає так, що тренер фіксує у програмі фізичні характеристики футболіста та змінює їх у міру необхідності, вносячи результати матчів, виконання нормативів, результати роботи на тренуванні. На виході автоматично формуються звіти за тиждень, місяць, за рік за відвідуваністю, фізичною динамікою, активністю на матчах, прогресом гравця.

Список функціональних можливостей:

– Усі дані гравців під рукою в електронному вигляді;

– Календар з розкладом тренувань, ігор та змагань;

– Графічний редактор тренувань, бібліотеки вправ та інструменти відеоаналізу;

– Особиста сторінка тренера зі списком усіх досягнень;

– Єдина CRM для контролю роботи тренерів, фінансового обліку та планування заходів;

– Закритий електронний архів клубних методик та доступ до відкритих даних інших шкіл;

– Дані захищені та залишаються у вас навіть після закінчення підписки на YOPLAYDO;

– Особистий прогрес та спортивні досягнення дитини у наочній формі;

– Цифрове резюме дитини з накопичених даних та відеофрагментів ігор;

– Актуальний розклад тренувань та матчів.

1.5.2 ProTrainUp

Додаток призначений для користувачів веб-сайту ProTrainUp. Додатком можуть користуватися як власники академії чи клубу, так і координатори, тренери, гравці та їх опікуни. Додаток дозволяє керувати роботою академії через клубний календар, контролювати сплату внесків, контролювати відвідування та надзвичайно просте та ефективне спілкування між усіма користувачами даного клубу [4]. Більше того, на рівні програми тренер може спланувати тренування, матч або перевірити відвідуваність.

Інтерфейс для роботи з платформою представлений на рис. 1.7.

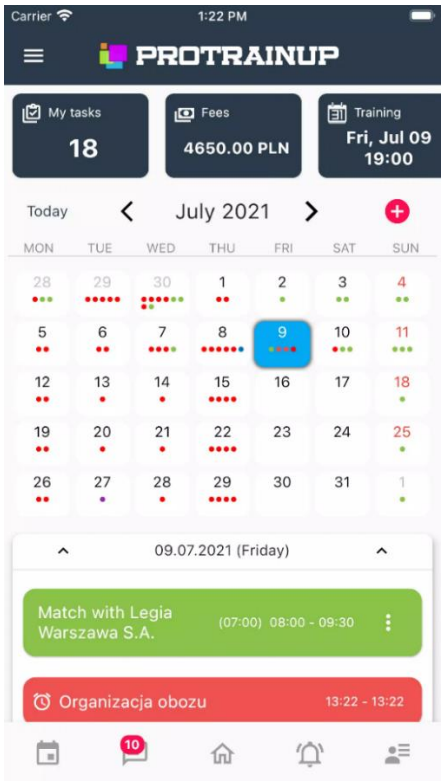

Рисунок 1.7 – Інтерфейс платформи «ProTrainUp»

Інформаційна система ProTrainUp корисна не тільки тренерам та керівникам школи, але гравцям та їхнім батькам, адже доступ до програми мають усі учасники процесу. У кожного представника футболіста є свій особистий кабінет у системі, з усією необхідною інформацією про гравця, графік тренувань

команди, розклад ігор та результатами зіграних матчів, актуальною інформацією про клубне життя, можливістю внутрішньої комунікації, оплатою абонементів тощо.

Як мінімум, батьки зможуть в режимі реального часу дізнаватися, чи дійшла їхня дитина до тренування, максимум, аналізувати за допомогою графіків, як юний футболіст прогресує, у чому він гарний, а де ще треба додати.

Гравці теж мають доступ до всіх статистичних даних, як з тренувань та тестів, так і контрольних та офіційних матчів. Ця інформація дозволить об'єктивно оцінювати себе.

Тренер має можливість відзначати прогрес футболіста не лише після кожної гри, а й після кожного тренування. У програмі також вбудована система відеоаналізу SportLiveTag, яка дозволяє генерувати статистичний аналіз поведінки кожного гравця окремо. Ці дані також будуть доступні гравцям та їхнім представникам.

ProTrainUp полегшить та покращить роботу співробітникам футбольної академії, але найголовніше, дозволить батькам та футболістам бути повністю інтегрованими у процес навчання наших вихованців.

Основні функціональні можливості:

– Щоденник та календар тренера. Тренер отримує інструмент, що поєднує кількісні та якісні дані в одному місці у вигляді коментарів, залишених у даних тренувань або матчах.

– Присутність на тренуваннях. Можливість швидкої перевірки присутності на тренуваннях через стандартну та мобільну версію системи.

– Месенджер. Спілкування за допомогою чату та особистих повідомлень, безумовно, допомагає в обміні інформацією між гравцями в команді.

– Мобільний додаток. Мобільний додаток підготовлений для систем IOS та Android.

– Віртуальна роздягальня та клубна дошка новин. Віртуальна роздягальня дає футбольним тренерам практично недосяжну можливість зібрати всю команду в одній віртуальній кімнаті, в нашій віртуальній роздягальні. Тут може відбуватися щоденний передтренувальний інструктаж, після якого гравці можуть тактично підготуватись до тренувань або матчу або запланувати розподіл енергетичних ресурсів. Віртуальна роздягальня також дозволяє заощадити дорогоцінний час для тренувань, необхідний для частого пояснення нових ігрових схем та користувачами системи. Через мобільний додаток можна також відслідковувати прямі трансляції матчів команд.

– Тренувальні конспекти у PDF.

– Спостереження за гравцем. Можливість створювати свої власні форми спостереження та наступне оцінювання обраних атрибутів гравця, що спостерігається.

– Система документообігу. Можливість завантажувати та обмінюватися документами в клубі чи академії – це ще одна можливість, пропонована ProTrainUp.

– Оцінка тренування. Цей модуль дозволяє оцінити ступінь та якість вправ, які підготували для гравців.

– Модуль медичних оглядів. Завдяки модулю медичних оглядів можна швидко заповнити профіль гравця з детальною інформацією про дату закінчення терміну дії медичних оглядів.

– Фінансовий модуль. Фінансовий модуль дозволяє контролювати внески гравця. Система дозволяє вводити щомісячні та разові внески. Завдяки цьому можна буде контролювати баланс зібраних внесків та інформувати гравців та батьків про будь-які заборгованості. Інформація буде надіслана у вигляді попередження на обліковому записі гравця та батьків.

1.5.3 Порівняльний аналіз розглянутих систем

Провівши аналіз існуючих розробок для підтримки діяльності футбольної академії, було зроблено висновок, що вони мають свої переваги та недоліки.

Головним недоліком став великий функціонал, який не потрібний для академії ФК «Прикарпаття», а також вартість розглянутих продуктів є дуже висока.

У таблиці 1.1 представлена зведена таблиця аналізу існуючих програмних продуктів.

Таблиця 1.1 – Аналіз існуючих програмних продуктів

Характеристики	*YoPlayDo*	*ProTrainUp*	*Розроблене програмне забезпечення*
Інтерфейс програми	Зручний	Зручний	Зручний
Функціонал	Календар з розкладом тренувань / ігор, бібліотеки вправ, інструменти відеоаналізу, всі дані про гравців в електронному вигляді, цифрове резюме гравця з накопичених	Щоденник та календар тренера, перевірка присутності на тренуванні, месенджер, віртуальна роздягальня, тренувальні конспекти у PDF, система документообігу,	Електронний облік гравців та тренерів, електронний розклад занять, вибір необхідного інвентарю для проведення заняття, можливість придбання

	даних та відеофрагментів	модуль медичних оглядів	абонементів різних видів
Процес впровадження	Платне	Платне	Безкоштовне
Технічні вимоги	Мінімальні	Мінімальні	Мінімальні
Вартість	25000 грн. за місяць	20000 грн. в рік	Безкоштовна

1.6 Обґрунтування доцільності проектування й розроблення

Ретельно вивчивши предметну область та провівши аналіз існуючих розробок для підтримки діяльності футбольної академії можна зробити висновок, що розробка нового програмного забезпечення є доцільною.

Кожній футбольній академії потрібне програмне забезпечення, що враховує особливості її діяльності, специфіку функціонування, що дозволяє виконувати певні функції обліку, які є вкрай важливими за умови постійного збільшення гравців академій, а також можливості налаштування та швидкого розширення реалізованого продукту, за такої потреби, в процесі його використання.

1.7 Концептуальна модель системи

Футбольна академія працює відповідно до законів та положень, які розробило керівництво футбольної академії, які називаються статутом футбольної академії, а також відповідно до законів прописаних в Конституції України. А саме положеннях про захист і права дітей та положень про їхню охорону здоров’я. Ці обмеження діють протягом усієї її діяльності. На вхід процесу «Організація діяльності футбольної академії» подається об’єкт дані

спортсмена, запит спортмена, список тренувань, дані про тренера, дані про академію, список та опис вправ, список інвентаря.

Це означає, що діяльність футбольної академії пов'язана з роботою зі спортсменами та тренерами. Механізми, якими управляється процес: персонал, обладнання та інформаційна система.

Декомпозиція діаграми розбиває процес «Організація діяльності футбольної академії» на підпроцеси, що відображають її детальний опис (рис. 1.8).

Перший підпроцес «Оформлення абонементу» має на вході три об'єкти: дані спортсмена, список тренувань та дані про академію. На виході він має такі об’єкти: договір про надання послуг, оформлення пакету послуг, список груп та індивідуальних занять.

Другий підпроцес «Оформлення тренерів» має на вході: список тренувань, дані про академію та дані про тренера, на виході - список оформлених тренерів.

Третій процес «Складання розкладу по академії» на вході складає об’єкт список тренувань та запит спортсмена, на виході ж буде – розклад тренувань та абонемент.

До підпроцесу «Підготовка академії до проведення занять» входять список тренувань, список та опис вправ, список інвентаря виданого для проведення заняття, на виході ж із підпроцесу – журнал відвідування спортсмена та звіт по приміщенню. Також, механізмом управління даного підпроцесу буде обладнання, що буде використовуватися під час проведення тренувального процесу. Та інформаційна система, яка буде значно облегшувати і пришвидшувати процес на кожному з кроків.

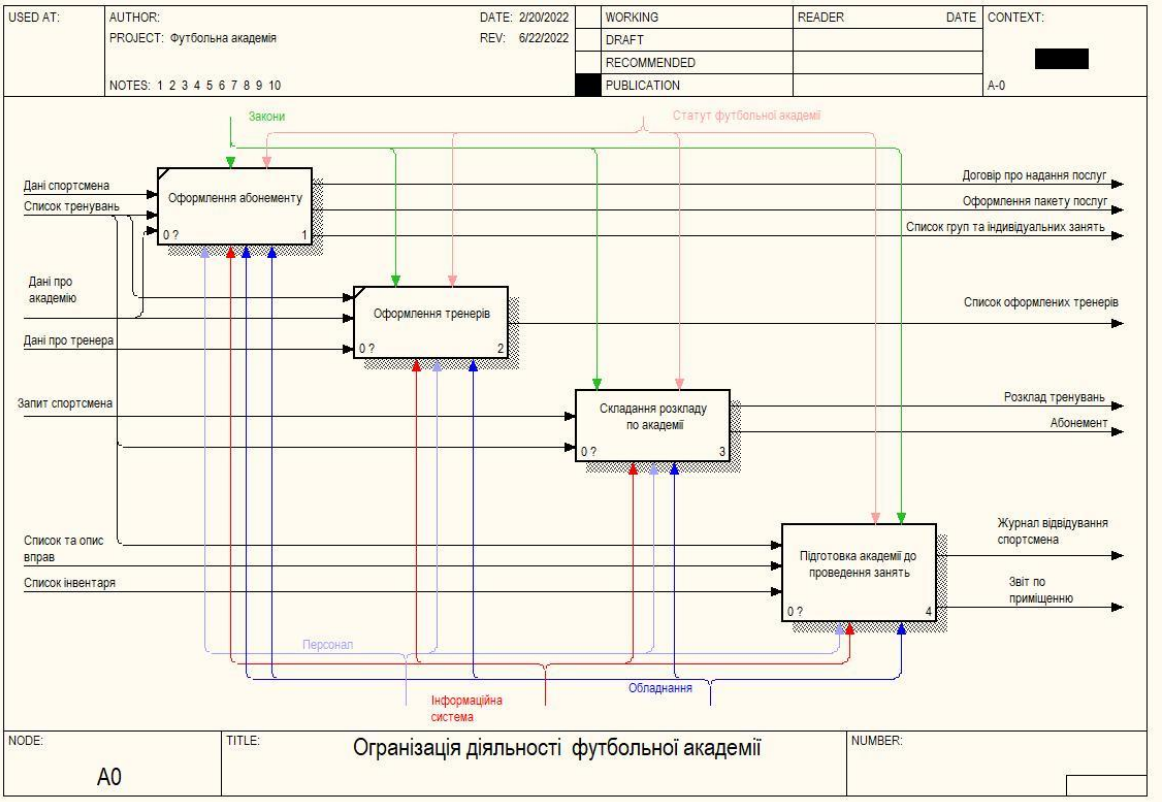

Рисунок 1.8 – Декомпозиція контекстної діаграми футбольної академії

Отже, за допомого декомпозиції процесу організації діяльності футбольної академії розроблювальне програмне забезпечення, що дозволить спростити процес планування розкладів занять та видачу інвентаря.

1.8 Постановка задачі

1.8.1 Цілі створення та призначення системи

Користувачі системи будуть: менеджер, адміністратор та тренера футбольної академії.

Головним завданням інформаційної підтримки діяльності футбольної академії «Прикарпаття» є підвищення ефективності праці адміністратора футбольної академії та тренерів, так як розроблена ІС дозволить зібрати всю інформацію в єдину систему

ІС створюється з метою:

– можливість швидко додавати, зберігати та редагувати дані;
– оптимізація роботи по обслуговуванню спортсменів;

– формування звітів;
– можливість складання розкладу тренувань;
– пошук необхідної інформації;
– перегляд розкладу занять за вибраний день.

1.8.2 Основні вимоги до системи

В запропонованій системі всі дані повинні розміщуватись в центральному сховищі даних на мережевому диску сервера, тобто ІС повинна бути централізованою.

В системі будуть виділені наступні функціональні підсистеми:

– підсистема реєстрації та авторизації;
– підсистема збору та обробки даних;
– підсистема збереження даних.

Інформаційна система повинна містити:

– відомості про спортсменів;
– відомості про тренерів;
– відомості про абонементи;
– відомості про інвентар.

Для джерела даних було використано СУБД MySQL, яка буде одночасно надавати доступ до читання даних для декількох комп’ютерів.

Інтерфейс користувача повинен бути зручним та зрозумілим, з малою кількістю частин функціонування. Інтерфейс буде розроблений на основі форм, за їх рахунку передбачено, що кінцевий користувач буде проводити менше часу задля вводу всієї інформації з клавіатури, за рахунок наявності списків і таблиць та зручного вибору даних серед них.

Вимоги до кваліфікаційного персоналу:

Персонал, який використовує програму, повин дотримуватись наступних вимог:

– Мати базові навички користування ПК;

– Дотримуватись важливих рекомендацій під час роботи в програмі, зокрема технологічних;

– Дотримуватись техніки безпеки при роботі з програмою та ПК.

Користувачами інформаційної системи будуть: адміністратора футбольної академії та тренер.

1.8.3 Функції, які повинна виконувати система

Інформаційна система повинна виконувати функції:

– Опрацювання даних спортсменів;
– Опрацювання даних тренерів;
– Опрацювання даних абонементів;
– Облік даних розкладів тренувань;
– Облік даних інвентаря.

Інформаційна система повинна містити засоби, що дозволяють проводити пошук даних та необхідної інформації:

– За прізвищем спортсмена;
– За датою народження;
– За групою тренування.

1.8.4 Вхідні та вихідні дані системи

Вхідні дані:

– Назва абонемента, ціна, дата початку дата закінчення;

– Прізвище та ім’я спортсмена, номер телефону, дата народження, адреса проживання та електронний адрес;

– Прізвище та ім’я тренера, номер телефону, дата народження, адреса проживання та електронний адрес;

Вихідні дані:

– Дані про спортсмена;
– Дані про тренера;
– Дані про абонементи;
– Розклади тренувань.

1.8.5 Обґрунтування вибору технічних засобів

При виборі впровадження програмного забезпечення на підприємство вжливим фактором є наявність необхідних обчислювальних ресурсів, які є в організації.

Мінімальними вимогами для запуску та належного функціонування розробленого програмного забезпечення є:

– Процесор PentiumI V/Xeon 2.4 ГГц.
– Оперативна пам'ять: 1024 Мб і більше.
– Вільний дисковий простір не менше 120 Мб.
– Мережева картка.
– Миша.
– Клавіатура.

З інформації наведеної вище, можна зробити висновок, що для повноцінного використання програми не потребується спеціального («потужного») технічного забезпечення.

2 Опис комплексу задач автоматизації

2.1 Інформаційне забезпечення системи

Маючи всі необхідні дані, проводимо детальний аналіз роботи футбольної академії. Спочатку необхідно створити фізичну модель бази даних засобами СУБД mySQL. Вигляд фізичної схеми даних наведений на рис. 1 у додатках.

Логічне проектування - це створення логічної моделі на основі вибраної моделі даних. Для початку, необхідно обрати СУБД, яка буде застосовуватися в системі. Потім, для перевірки вірності логічної моделі застосовується нормалізація. Перетворення локальної концептуальної моделі даних в локальну логічну модель полягає насемперед, в видаленні небажаних частин та модифікації отриманих моделей в локальні логічні моделі. До не бажаних елементів відносяться:

– зв’язки типу «багато-до-багатьох»;
– рекурсивні зв’язки;
– зв’язки з атрибутами.

Розроблена мною база даних складається із 8-ми таблиць.

Таблиця «Logs» містить інформацію про події системи (табл. 2.1).

Таблиця 2.1 – «Logs»

Поле таблиці	Тип даних	Опис поля
LogsId	INT	Унікальний ідентифікатор події
UsersId	INT	Ідентифікатор користувача
EventNameShow	TEXT	Назву події, що відбулася в системі
EventDate	DATETIME	Дата події

Таблиця «Schedule» містить інформацію про розклади занять (табл. 2.2).

Таблиця 2.2 – «Schedule»

Поле таблиці	Тип даних	Опис поля
ScheduleId	INT	Унікальний ідентифікатор розкладу
ScheduleName	VARCHAR(255)	Назва заняття
DateOfOccupation	DATETIME	Дата заняття
StartDate	DATETIME	Початок заняття
EndDate	DATETIME	Кінець заняття
TrenerId	INT	Ідентифікатор тренера

Таблиця «ScheduleL» містить інформацію про розклад занять окремо для кожного футболіста академії (табл. 2.3).

Таблиця 2.3 – «ScheduleL»

Поле таблиці	Тип даних	Опис поля
ScheduleLId	INT	Унікальний ідентифікатор списку розкладів для кожного футболіста
TrenerId	INT	Ідентифікатор тренера
SoccerPlayerId	INT	Ідентифікатор футболіста
GroupId	INT	Ідентифікатор групи
ScheduleId	INT	ідентифікатор розкладу занять

Таблиця «SoccerPlayer» містить інформацію про гравців футбольної академії (табл. 2.4).

Таблиця 2.4 – «SoccerPlayer»

Поле таблиці	Тип даних	Опис поля
SoccerPlayerId	INT	Унікальний ідентифікатор футболіста

FirstName	VARCHAR(45)	І’мя футболіста
LastName	VARCHAR(45)	Прізвище футболіста
Phone	VARCHAR(25)	Номер телефону
BirthDate	DATETIME	Дата народження
Address	TEXT	Домашня адреса футболіста
Email	nvarchar(45)	Електронний адрес
GroupId	INT	Ідентифікатор групи

Таблиця «Trener» містить інформацію про тренерів футбольної академії (табл. 2.5).

Таблиця 2.5 – «Trener»

Поле таблиці	**Тип даних**	**Опис поля**
TrenerId	INT	Унікальний ідентифікатор тренера
FirstName	VARCHAR(45)	І’мя тренера
LastName	VARCHAR(45)	Прізвище тренера
Phone	VARCHAR(25)	Номер телефону
BirthDate	DATETIME	Дата народження
Address	TEXT	Домашня адреса тренера
Email	nvarchar(45)	Електронний адрес

Таблиця «Users» містить інформацію про користувачів програми (табл. 2.6).

Таблиця 2.6 – «Users»

Поле таблиці	**Тип даних**	**Опис поля**
UsersId	INT	Унікальний ідентифікатор користувача
FirstName	VARCHAR(45)	І’мя користувача

LastName	VARCHAR(45)	Прізвище користувача
UsersName	VARCHAR(45)	Номер телефону
UsersPassword	VARCHAR(45)	Дата народження
RoleId	INT	Домашня адреса користувача
Description	TEXT	Електронний адрес

Таблиця «Clusters» містить інформацію про групи тренувань (табл. 2.7).

Таблиця 2.7 – «Clusters»

Поле таблиці	**Тип даних**	**Опис поля**
GroupsId	INT	Унікальний ідентифікатор групи
GroupsName	VARCHAR(255)	Назва групи
Description	TEXT	Опис
TrenerId	INT	ідентифікатор тренера

Таблиця «Abonement» містить інформацію про абонементи (табл. 2.8).

Таблиця 2.8 – «Abonement»

Поле таблиці	**Тип даних**	**Опис поля**
AbonementId	INT	Унікальний ідентифікатор абонементу
AbonementName	VARCHAR(255)	Назва абонементу
Description	TEXT	Опис
Price	FLOAT	Ціна абонементу
StartDate	DATETIME	Початок дії абонементу
EndDate	DATETIME	Кінець дії абонементу

Таблиця «Inventorys» містить інформацію про абонементи (табл. 2.9).

Таблиця 2.9 – «Inventorys»

Поле таблиці	Тип даних	Опис поля
InventorysId	INT	Унікальний ідентифікатор інвентаря
InventorysName	VARCHAR(255)	Назва інвентаря
Description	TEXT	Опис

Таблиця «IssuanceOfInventory» містить інформацію про список інвентаря, що необхідний для проведення заняття (табл. 2.10).

Таблиця 2.10 – «IssuanceOfInventory»

Поле таблиці	Тип даних	Опис поля
IssuanceOfInventoryId	INT	Унікальний ідентифікатор списку інвентаря
ScheduleId	INT	Ідентифікатор розкладу тренування
InventorysId	INT	Ідентифікатор інвентаря
Amount	INT	Кількість одиниць

В наш час бази даних можуть містити величезні масиви інформації, тому можливість обробляти її вручну, через методи післідовного редагування значень в таблицях стає як ніколи неефективним способом. Для вирішення цієї проблеми, та збільшення загальної ефективності користуванням базами даних застосовують запити, які дозволяють проводити обробку великої кількості даних, тобто отримувати з великого масиву даних саме ту інформацію яка нас цікавить найбільше.

Запит – це засіб отримання інформації з бази даних.

Запит представляє чітко описану вимогу, яка визначає склад операцій, які виконуються над базою даних.

В даній роботі для створення запитів було використано мову SQL (Structured Query Language).

В базі даних, було написано запити на редагування, видалення та додавання інформації до таблиць. Вони були реалізовані безпосередньо в самому додатку.

2.2 Алгоритмізація та реалізація комплексу задач автоматизації

Після створення бази даних, нам необхідно її під’єднати до нашого середовища розробки, а саме Visual Studio 2019. За допомогою цього середовища розробки в нас є можливість підключення наших таблиць до існуючих форм, на яких буде реалізовано додавання та видалення даних.

Для під’єднання БД до середовища Microsoft Visual Studio 2019 у файлі проекту "App.config" створюємо змінну "CONNECTSQL" та задаємо значення параметрів (Лістинг 2.1).

Лістинг 2.1 Параметри змінної "CONNECT"

```
  <appSettings>
    <!-- Підключення до бази даних -->
    <add key="CONNECTSQL" value="server=localhost;
user=root;database=fa;password=12345;" />
  </appSettings>
```

Для роботи з створеною базою даних використовуємо простір імен MySql.Data.MySqlClient.

Для того, щоб створити меню, додаємо елемент menuStrip (рис. 2.1).

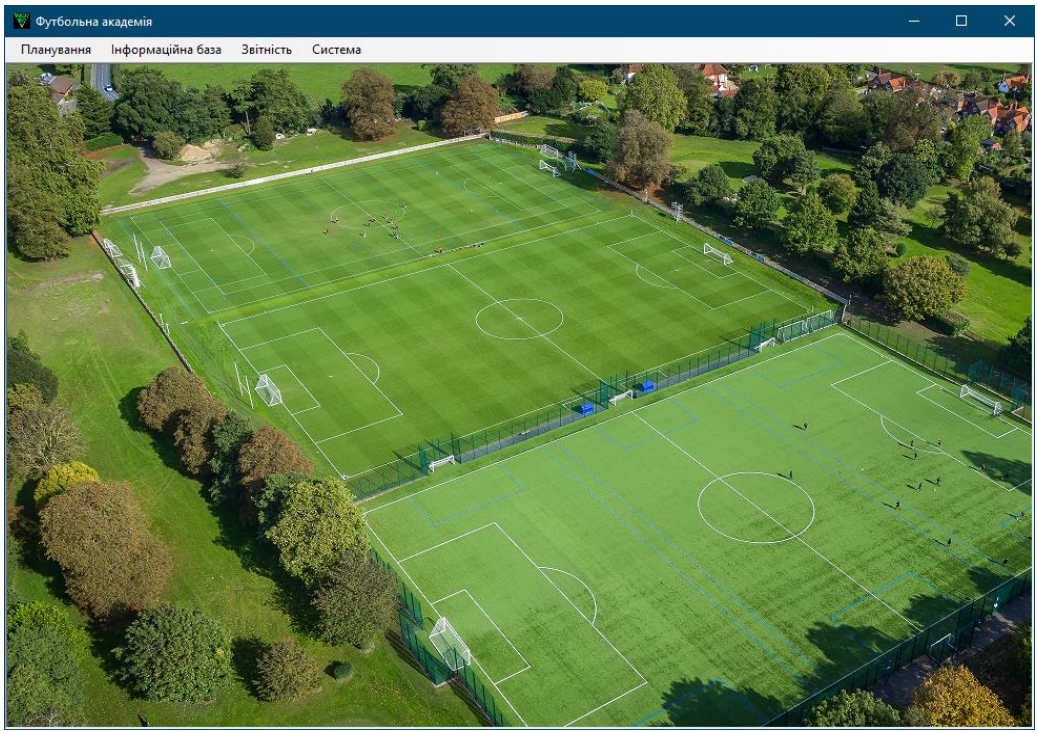

Рисунок 2.1 – Створене меню системи

До кожного з елементу меню додано код, який і відповідає за створення макетів форм та закриття попередньо відкритого вікна.

Лістинг 2.2 Код для події "ToolStripMenuItem_Click"

```
CloseAllWindows();
ScheduleRaportForm scheduleRaportForm = new ScheduleRaportForm();
scheduleRaportForm.MdiParent = this;
scheduleRaportForm.WindowState = FormWindowState.Maximized;
scheduleRaportForm.Show();
```

Для робити в БД було створено класи, рохташовані вони в папці, яка має назву «Providers» в проекті рішення.

Для з’єднання з розробленою базою даних використовується метод «Open», який належить екземпляру класу «SqlConnection». Для закриття з’єднання з базою даних використовується метод «Close».

Для опрацювання даних створення нового розкладу був створений клас «StatementProvider», який містить в собі 6 публічних методів для опрацювання інформації: додавання нового розкладу для тренування, вибірка всіх розкладів, вибірка конкретного розкладу по його ідентифікатору, редагування вибраного

розкладу із списку, видалення вибраного розкладу, та вибір останнього доданого розкладу.

Лістинг 2.3 Код методу "InsertSchedule"

```
public void InsertSchedule(string ScheduleName, DateTime DateOfOccupation,
DateTime StartDate, DateTime EndDate, int TrenerId) {
MySqlConnection connection = new MySqlConnection(_ConnString);
string query = "INSERT INTO Schedule (ScheduleName, DateOfOccupation, StartDate,
EndDate, TrenerId) VALUES(@ScheduleName, @DateOfOccupation, @StartDate, @EndDate,
@TrenerId)";
MySqlCommand cmd = new MySqlCommand(query, connection);
cmd.Parameters.AddWithValue("@ScheduleName", ScheduleName);
cmd.Parameters.AddWithValue("@DateOfOccupation",
DateOfOccupation.ToString("yyyy-MM-dd HH:mm:ss"));
cmd.Parameters.AddWithValue("@StartDate", StartDate.ToString("yyyy-MM-dd
HH:mm:ss"));
cmd.Parameters.AddWithValue("@EndDate", EndDate.ToString("yyyy-MM-dd
HH:mm:ss"));
cmd.Parameters.AddWithValue("@TrenerId", TrenerId);
connection.Open();
cmd.ExecuteNonQuery();
connection.Close();
connection.Dispose();
connection = null;
}
```

Код методу для редагування даних розкладу приведений в лістингу 2.4.

Лістинг 2.4 Код методу "UpdateSchedule"

```
public void UpdateSchedule(string ScheduleName, DateTime DateOfOccupation, DateTime
StartDate, DateTime EndDate, int TrenerId, int ScheduleId) {
MySqlConnection connection = new MySqlConnection(_ConnString);
string query = "UPDATE Schedule SET ScheduleName = @ScheduleName,
DateOfOccupation = @DateOfOccupation " +
" WHERE ScheduleId = @ScheduleId";
…. (див додаток)
}
```

Код методу для вибірки всіх пунктів розкладу представлений в лістингу 2.5.

Лістинг 2.5 Код методу "GetAllSchedule"

```
 public List<Schedule> GetAllSchedule() {
int i = 0;
List<Schedule> ScheduleList = new List<Schedule>();
MySqlConnection connection = new MySqlConnection(_ConnString);
string query = "SELECT * FROM Schedule ";
```

…. (див додаток)

Код для вибору конкретного пункту розкладу продемонстрований в лістингу 2.6.

Лістинг 2.6 Код методу "SelectedScheduleByScheduleId"

```
public Schedule SelectedScheduleByScheduleId(int ScheduleId) {
    Schedule selectedSchedule = new Schedule();
    MySqlConnection connection = new MySqlConnection(_ConnString);
    string query = "SELECT * FROM Schedule WHERE ScheduleId=" +
ScheduleId.ToString();
    MySqlCommand command = new MySqlCommand(query, connection);
    connection.Open();
```

…. (див додаток)

Код методу для вибірки останнього запису в розклад приведений в лістингу 2.7.

Лістинг 2.7 Код методу "GetLastRecords"

```
public int GetLastRecords() {
    Schedule selectedSchedule = new Schedule();
    MySqlConnection connection = new MySqlConnection(_ConnString);
    string query = "SELECT ScheduleId FROM Schedule WHERE ScheduleId=(SELECT
MAX(ScheduleId) FROM Schedule)";
    MySqlCommand command = new MySqlCommand(query, connection);
    connection.Open();
    MySqlDataReader reader = command.ExecuteReader();
    while (reader.Read()) {
      selectedSchedule.ScheduleId= Convert.ToInt32(reader["ScheduleId"]);
    }
```

```
    reader.Close();
    connection.Close();
    return selectedSchedule.ScheduleId;
  }
```

Код методу для видалення вибраного пункту розкладу приведений в лістингу 2.8.

Лістинг 2.8 Код методу "DeleteSchedule"

```
public void DeleteSchedule(int ScheduleId) {
  MySqlConnection connection = new MySqlConnection(_ConnString);
  string query = "DELETE FROM Schedule WHERE ScheduleId = @ScheduleId";
  MySqlCommand cmd = new MySqlCommand(query, connection);
  cmd.Parameters.AddWithValue("@ScheduleId", ScheduleId);
  connection.Open();
  cmd.ExecuteNonQuery();
  connection.Close();
  connection.Dispose();
  connection = null;
}
```

Наступним кроком було створення форми для створення розкладу тренувань (рис. 2.2).

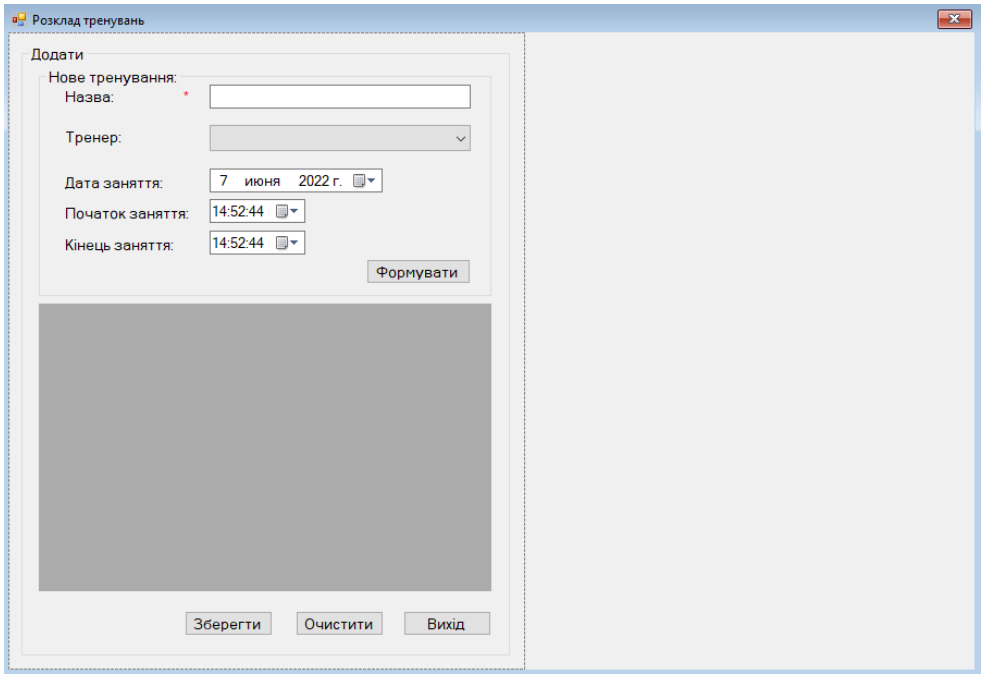

Рисунок 2.2 – Форма для створення розкладу тренувань

При завантаження форми створення розкладу тренувань у конструкторі форми викликаються два методи «LoadAllDate» та «DataLoad», які завантажують дані про тренерів та розклади тренувань (ліст. 2.9).

Лістинг 2.9 Код методів "LoadAllDate" та "DataLoad"

```
  private void LoadAllDate() {
    _TrenerList = _TrenerProvider.GetAllTrener();
    TrenerCBox.DataSource = _TrenerList;
    TrenerCBox.ValueMember = "TrenerId";
    TrenerCBox.DisplayMember = "FIO";
   }
…. (див додаток)
```

Для додавання нового тренування в базу даних у формі вікна реалізовано метод «AddBtn_Click», що спрацьовує на натискання кнопки «Додати» (ліст. 2.10).

Лістинг 2.10 Код методу "AddBtn_Click"

```
    private void AddBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect() && _ScheduleLList.Count > 0) {
        _ScheduleProvider.InsertSchedule(ScheduleNameTBox.Text,
DateOfCompletionDTP.Value,
       StartDateDTP.Value, EndDateDTP.Value, …. (див додаток)
```

Як можна побачити, в методі "AddBtn_Click" ми викликаємо метод «IsDataEnteringCorrect», який відповідає за перевірку даних на правильність введення, для належного їх запису в БД. Код методу «IsDataEnteringCorrect» приведений в лістингу 2.11.

Лістинг 2.11 Код методу «IsDataEnteringCorrect»

```
    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
      if (_validation.IsDataEntering(ScheduleNameTBox.Text)) {
        ScheduleNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        ScheduleNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
```

```
        }
        return isCorrect;
    }
```

За видалення створеного тренування із списку відповідає метод «ScheduleGridView_CellClick». Після натискання кнопки «Видалити» з'явиться вікно в якому необхідно підтвердити видалення вибраного тренування (ліст. 2.12).

Лістинг 2.12 Код методу «ScheduleGridView_CellClick»

```
        private void ScheduleGridView_CellClick(object sender, DataGridViewCellEventArgs e)
{
            if (e.ColumnIndex == 6) {
                if (MessageBox.Show("Ви дійсно бажаєте видалити це тренування?", "Видалити",
MessageBoxButtons.YesNo) == DialogResult.Yes) {
                    int selectedScheduleId = Convert.ToInt32(ScheduleGridView[0,
e.RowIndex].Value.ToString());
                    _ScheduleProvider.DeleteSchedule(selectedScheduleId);
                    _ScheduleLProvider.DeleteScheduleL(selectedScheduleId);
                    DataLoad();
    …. (див додаток)
```

2.3. Інструкція користувача

Для того, щоб розпочати роботу з програмою необхідно запустити програму із назвою «FootballAcademy». Після запуску програми буде відкрито вікно вводу логіну та паролю, де користувач зможе ввести цю інформацію (рис.2.3).

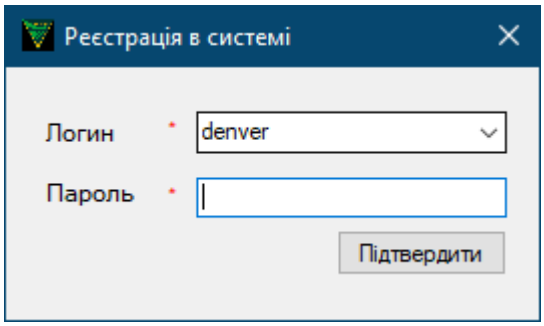

Рисунок 2.3 – Реєстрація користувача в системі

Якщо ж інформація введеня користувачем не є коректною, то він буде попереджений про це відповідним повідомленням (рис.2.4).

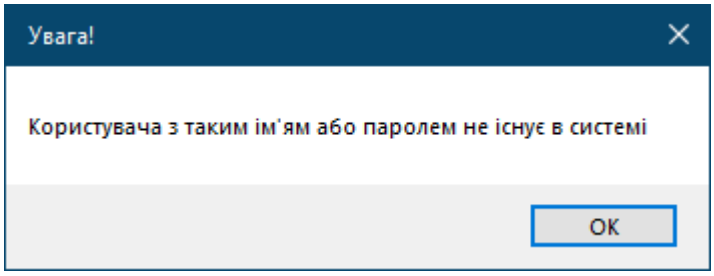

Рисунок 2.4 – Попередження про неправильне введення даних

Для швидшого пошуку необхідного акаунту є можливість вибору ідентифікаційних даних з запропонованого списку.

Далі, після успішної авторизації користувача в програмі буде відкрито головне меню програми (рис.2.5).

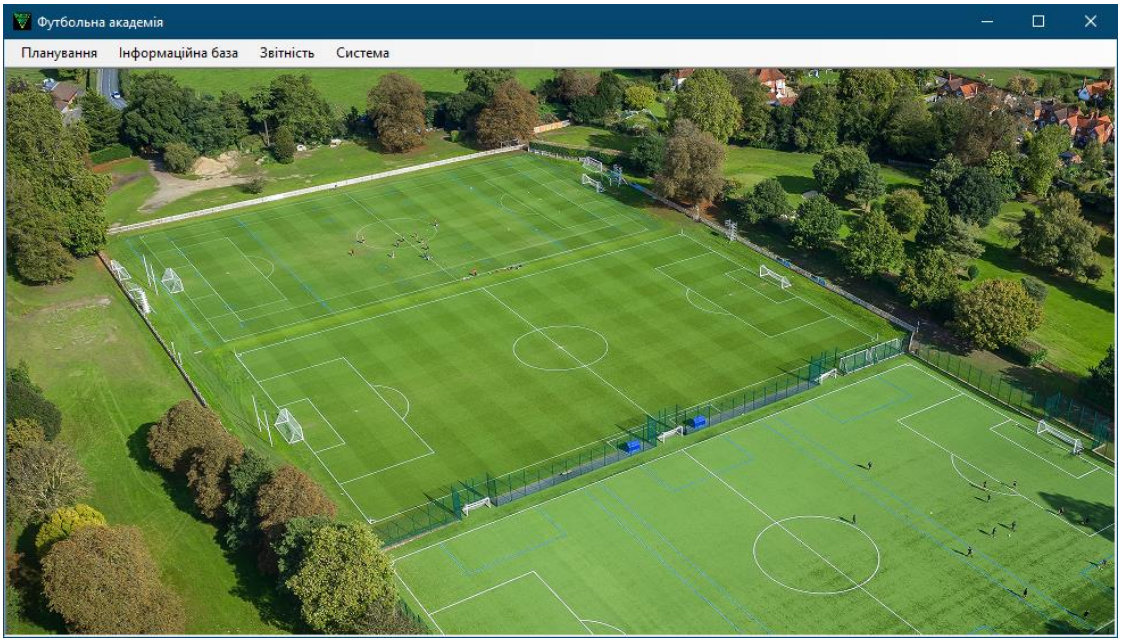

Рисунок 2.5 – Головне вікно програми

Оскільки ми ввійшли в програму від імені адміністратора, то для нас відкрити всі можливі функції програми.

Для початку роботи необхідно додати інформацію про тренерів футбольної академії, для цього необхідно ввести дані про тренерів (рис 2.6).

Також в даному вікні відображається список всіх тренерів, які вже були додані раніше.

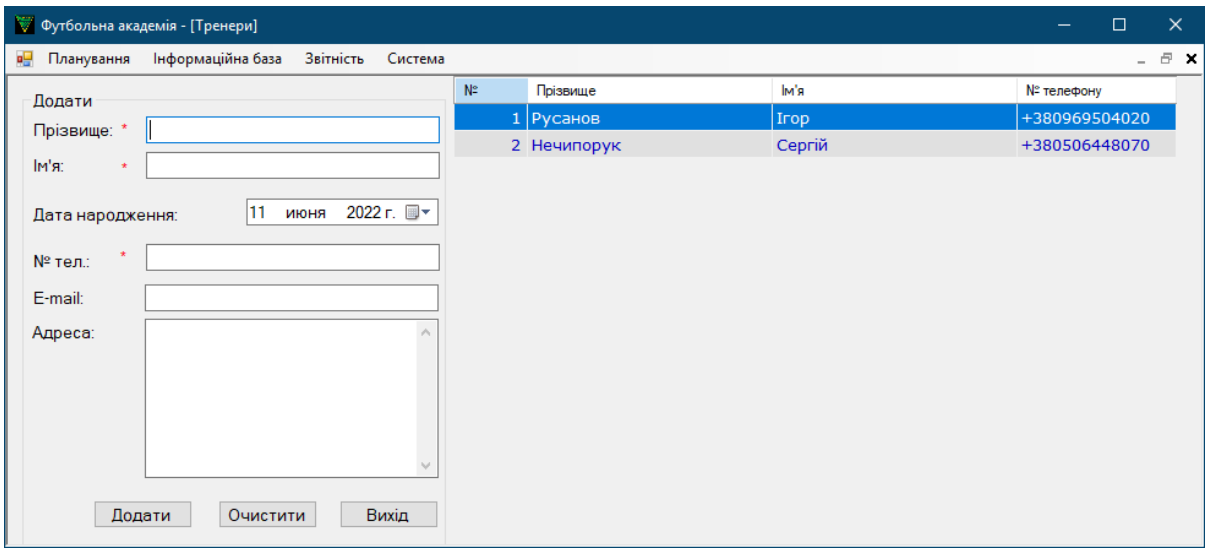

Рисунок 2.6 – Вікно для опрацювання даних про тренерів

Дана програма звіряє всі дані введені користувачами на їхню коректність. Варто зазначити, що в програмі поля, які є обов'язковими для заповнення, позначені зірочкою.

Також в програмі є опція зміни та/або видалення інформації у випадку якщо вона є застарілою або просто вже не потрібна. На рис 2.7 зображено вікно для редагування даних про вибраного із списку тренера.

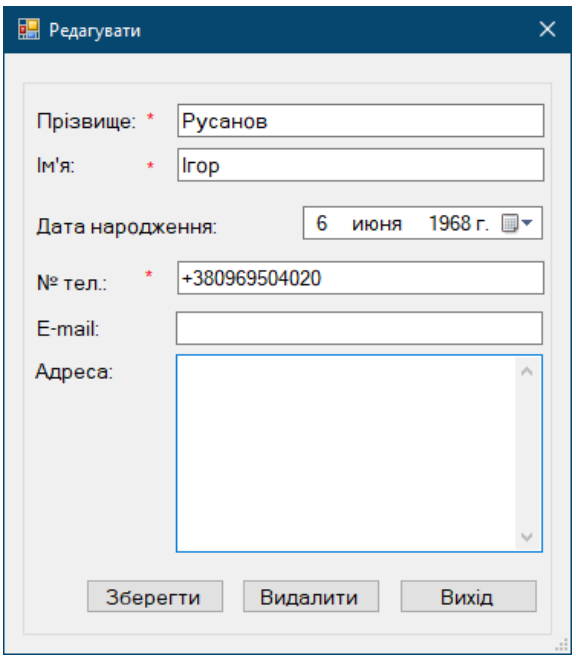

Рисунок 2.7 – Вікно для редагування даних вибраного тренера

Далі необхідно створити групи, в яких будуть тренуватись футболісти. Для цього необхідно перейти по головному меню «Інформаційна база» –> «Групи»., після чого відкриється відповідне вікно (рис. 2.8).

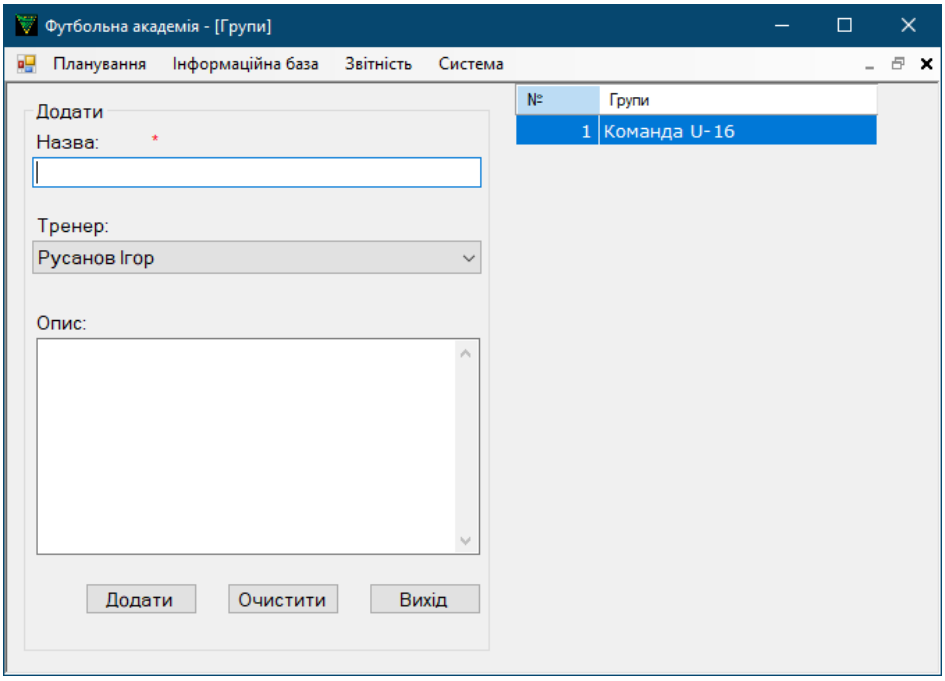

Рисунок 2.8 – Вікно для додавання нової групи

Також у випадку помилки, дані про групу можна редагувати, або якщо вона є не актуальною, то її можна видалити з системи (рис. 2.9).

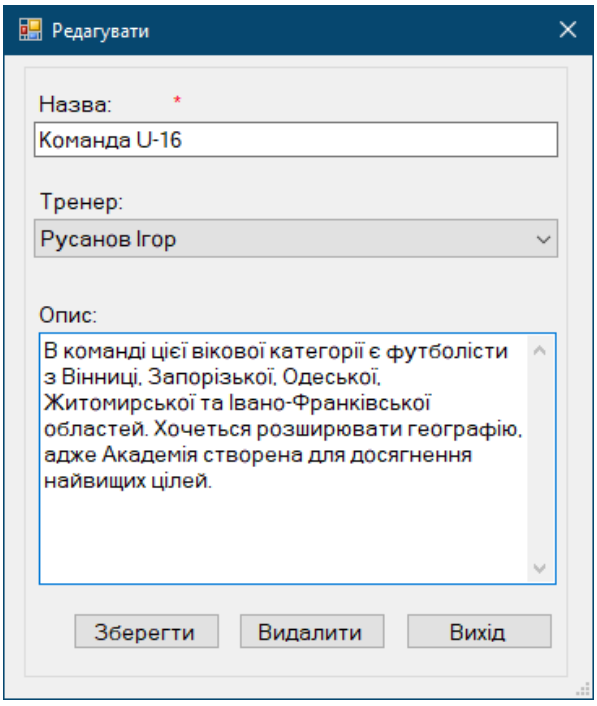

Рисунок 2.9 – Вікно для редагування даних групи

Для ведення інформаційної бази гравців створена можливість додавання та перегляду їх особистих даних. Задля того щоб потрапити на цю сторінку необхідно перейти по меню «Інформаційна база» –> «Гравці». Після цього виведеться вікно зі списком всіх доданих в систему гравців (рис. 2.10).

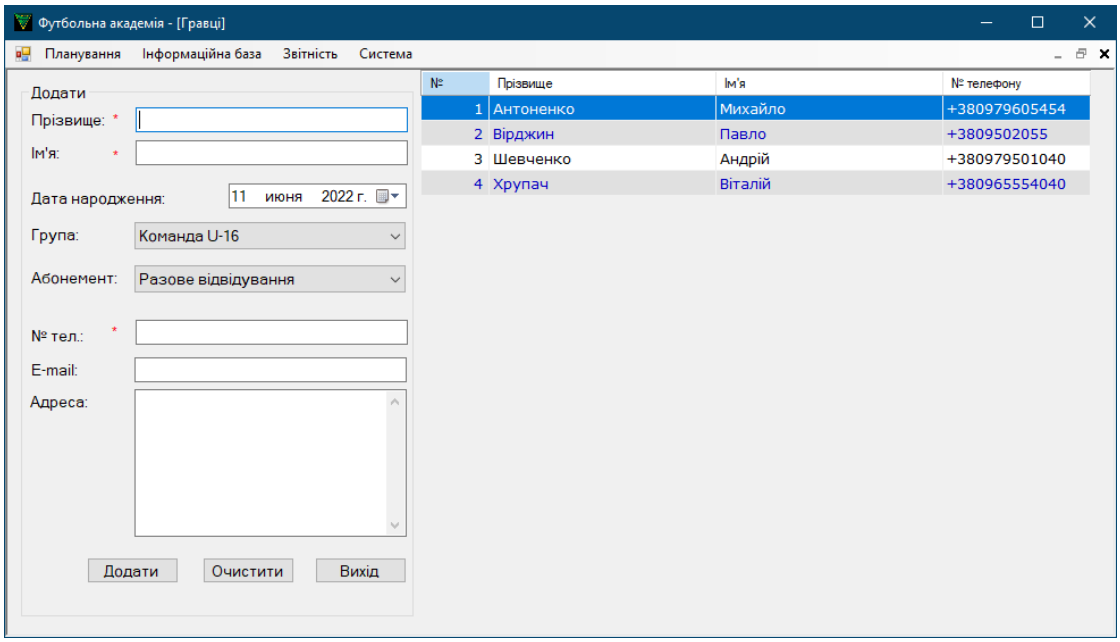

Рисунок 2.10 – Вікно для опрацювання інформації про гравців

В разі необхідності інформацію про гравця можна відредагувати або видалити (рис. 2.11).

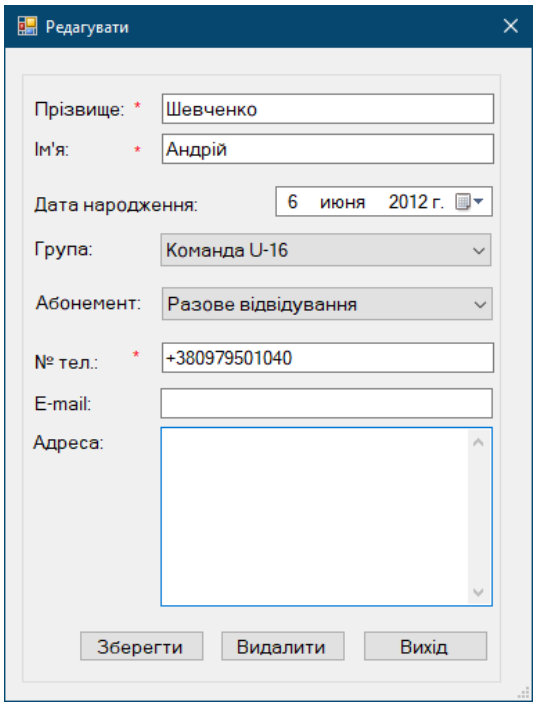

Рисунок 2.11 – Вікно для редагування інформації про гравця

Після доданої інформаційної бази можна приступити до планування розкладів тренувань. Для цього необхідно перейти по головному меню програми «Планування» –> «Розклад тренувань», після чого відкриється вікно з можливістю планування розкладів тренувань для створених футбольних груп. (рис. 2.12).

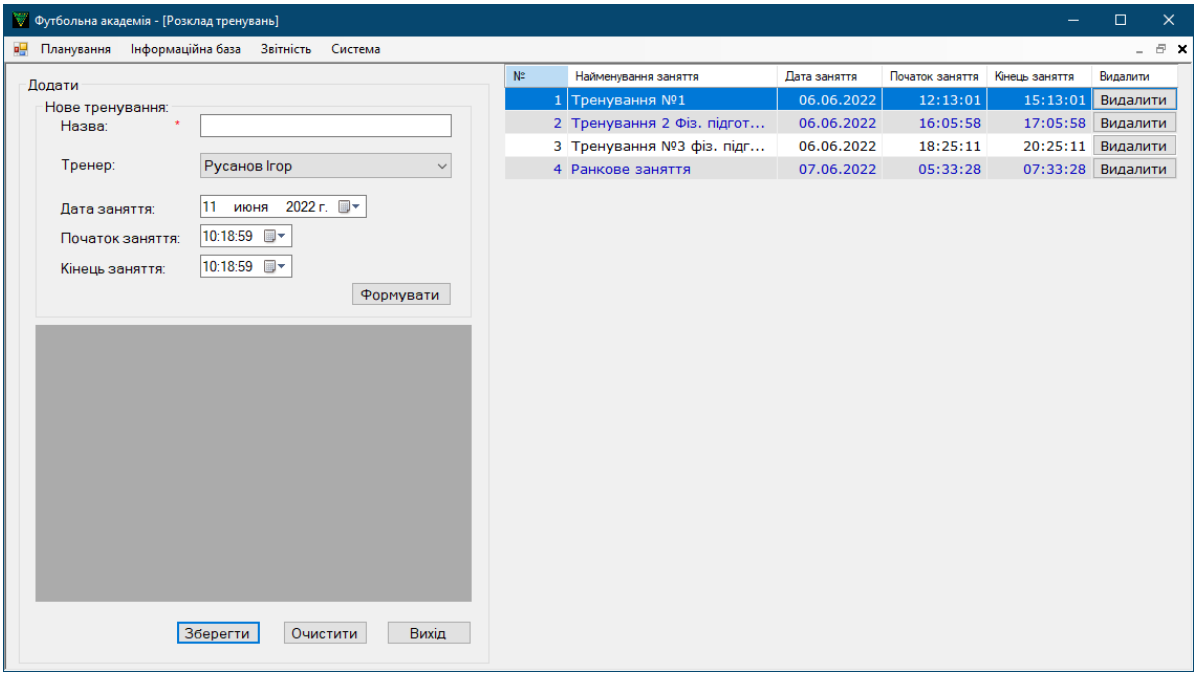

Рисунок 2.12 – Вікно для планування розкладів занять

Для опрацювання інформації про абонементи необхідно перейти по меню програми «Інформаційна база» –> «Абонементи», після чого відкриється відповідне вікно (рис. 2.13).

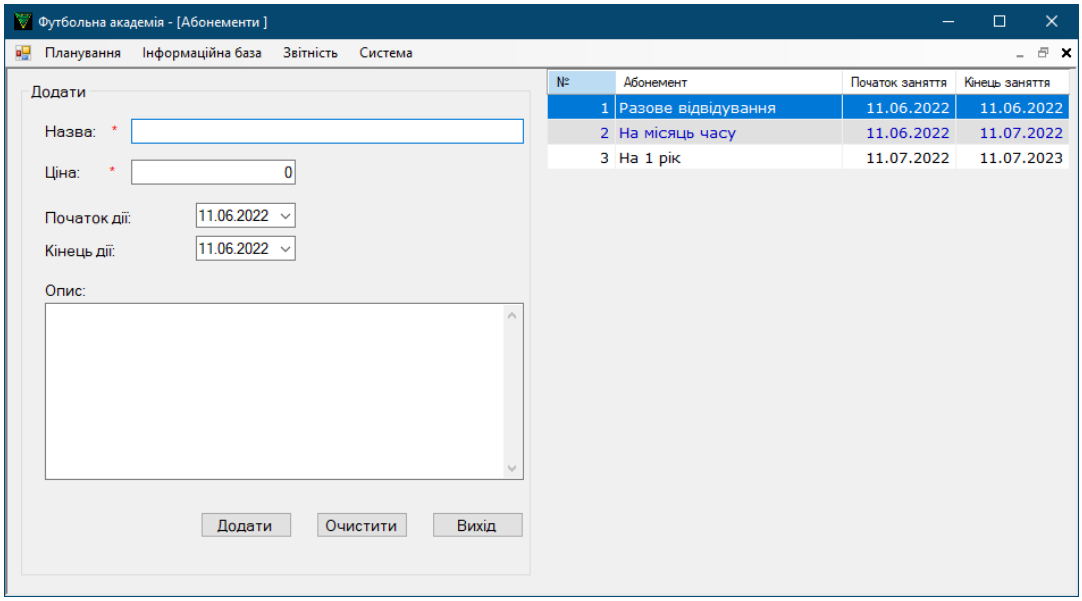

Рисунок 2.13 – Опрацювання інформації про абонементи

Кожне заняття на проводиться без відповідного інвентаря, тому, щоб його додати до тренування необхідно у правій частині вікна вибрати неохідне заняття, після чого з’явиться відповідне вікно, де можна опрацювати інформацію по інвентарю для кожного заняття (рис. 2.14).

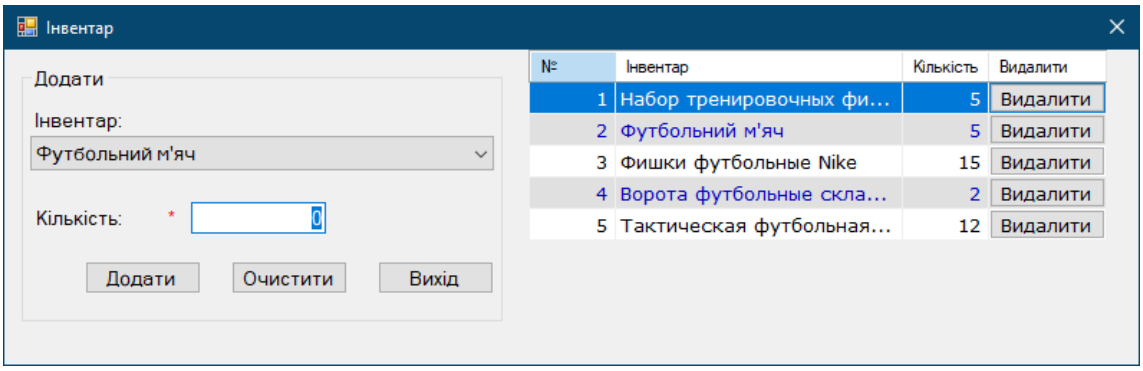

Рисунок 2.14 – Вікно для опрацювання необхідного інвентаря для заняття

Організація заняття не обходиться без інвентаря, тому у програмі реалізована можливість обліку інвентаря. Для того, щоб опрацювати інформацію про інвентар, користувачу системи необхідно перейти по меню програми «Інформаційна база» –> «Інвентар», після чого відкриється вікно з можливістю перегляду інвентаря у футбольній академії. Якщо ж потрібного інвентаря немає, його завжди можна додати. На рис. 2.15 зображено вікно для опрацювання даних інвентаря.

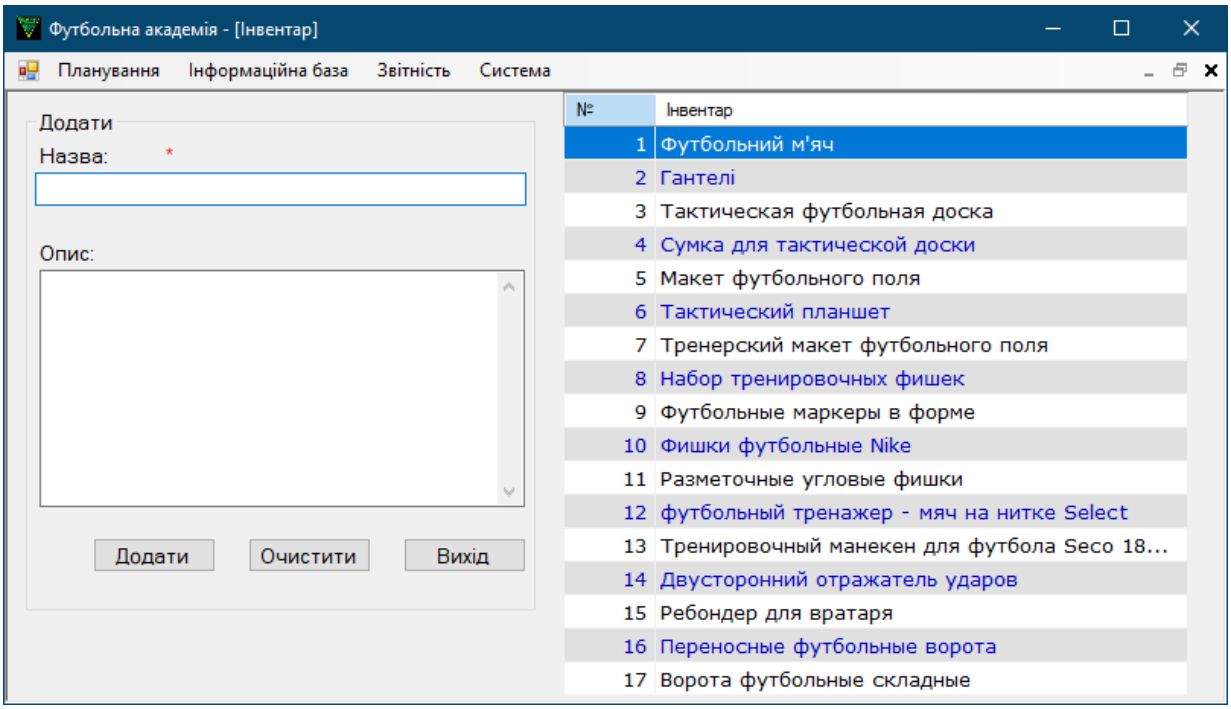

Рисунок 2.15 – Вікно для опрацювання даних інвентаря

Щоб відредагувати інвентар необхідно у правій частині вікна вибрати необхідний інвентар та натиснути на нього лівою кнопкою миші, після чого зявиться вікно для редагування або видалення даних інвентаря (рис. 2.16).

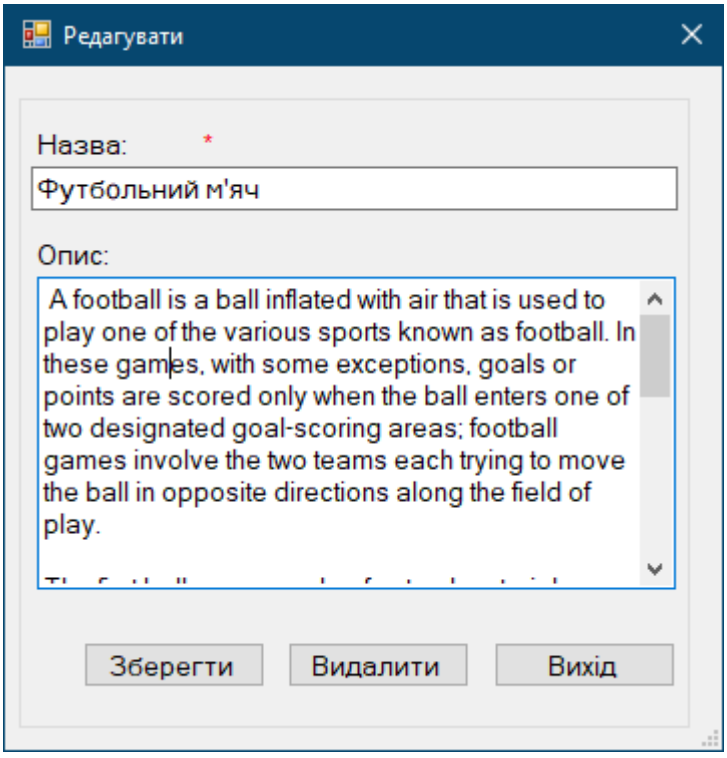

Рисунок 2.16 – Вікно для редагування інформації вибраного інвентаря

Для перегляду запланованого розкладу тренувань користувачу системи необхідно перейти по меню програми «Звіти» –> «Розклад», після чого відкриється вікно з можливістю перегляду розкладу тренувань (рис. 2.17).

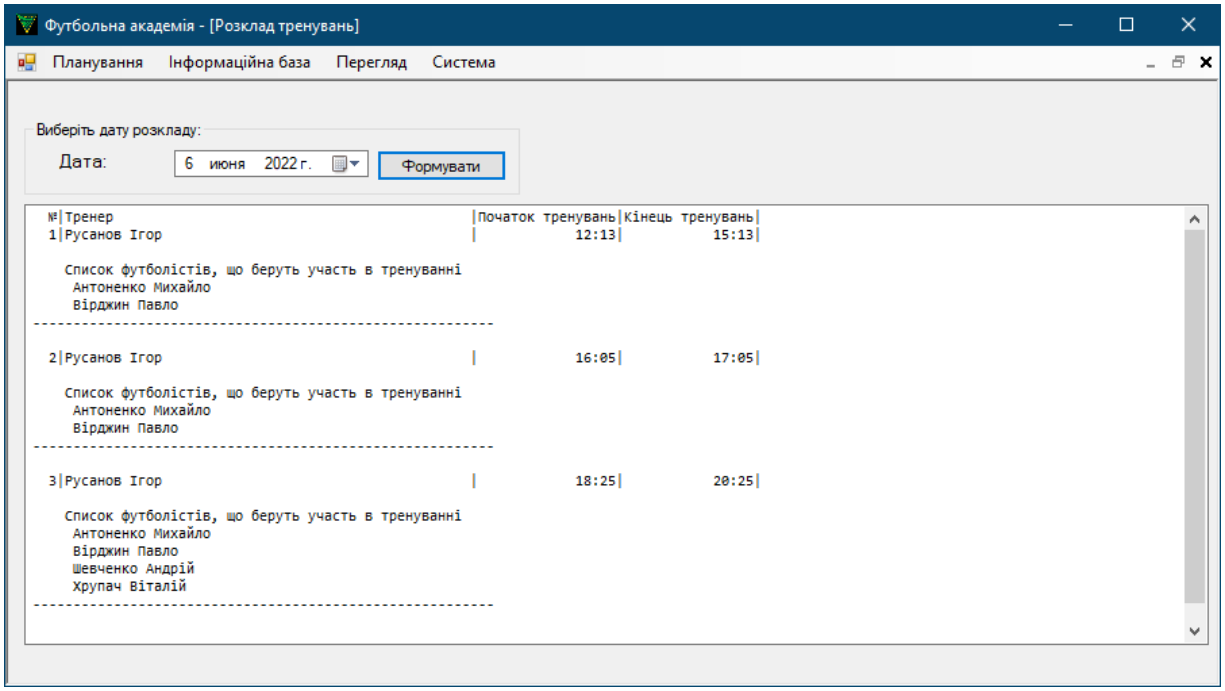

Рисунок 2.17 – Розклад тренувань

Щоб створити звітність по вибраному спортсмену, користувачу системи потрібно перейти по меню програми «Звіти» –> «По спортсмену», після чого відкриється вікно для формування звітності про вибраного спортсмена (рис. 2.18).

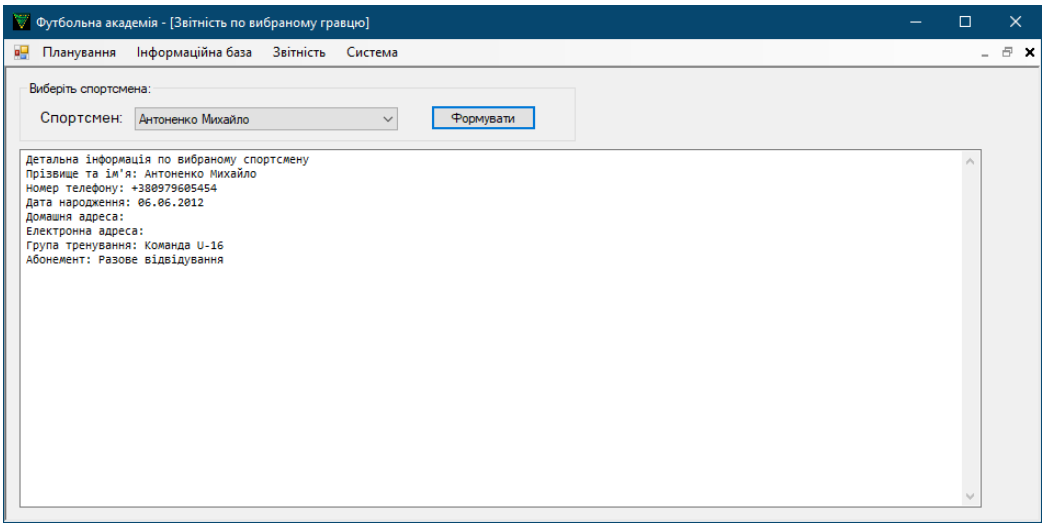

Рисунок 2.18 – Вікно для формування звітності про спортсмена

Для створення звітності по вибраному тренеру, користувачу системи потрібно перейти по меню програми «Звіти» –> «По тренеру», після чого відкриється вікно для формування звітності про вибраного тренера (рис. 2.19).

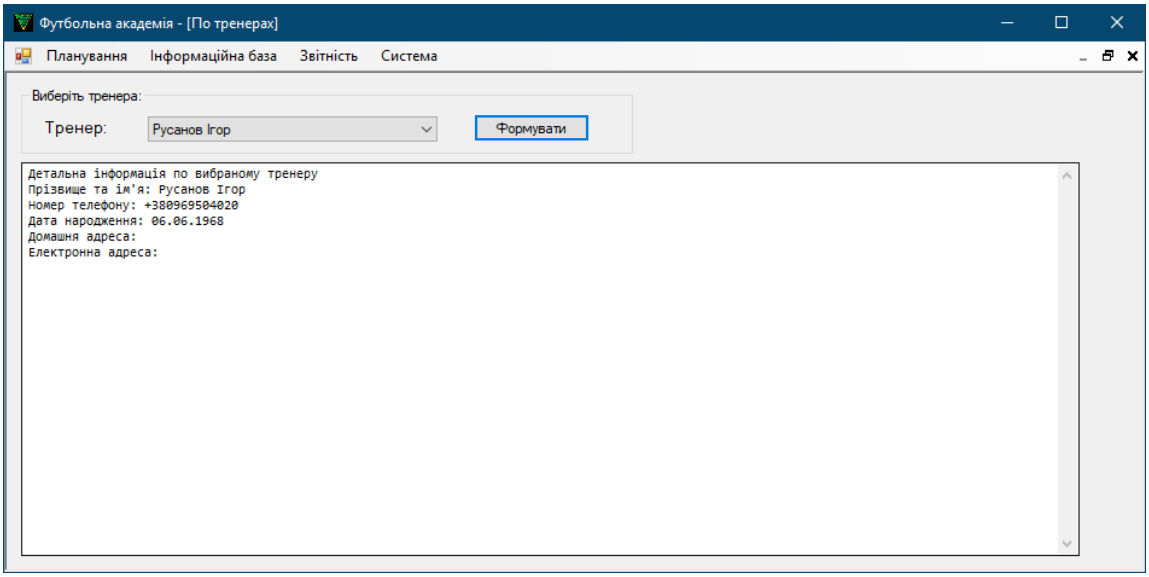

Рисунок 2.19 – Вікно для формування звітності про тренера

Для створення звітності по вибраному абонементу, користувачу системи потрібно перейти по меню програми «Звіти» –> «По абонементу», після чого відкриється вікно для формування звітності про вибраний абонемент (рис. 2.20).

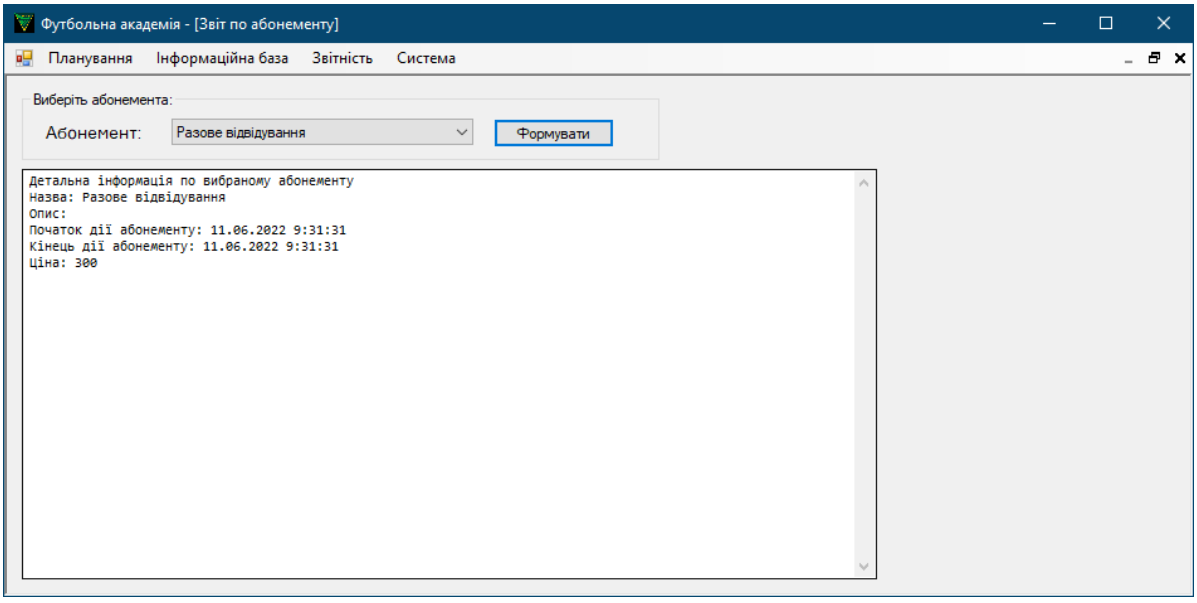

Рисунок 2.20 – Вікно для формування звітності про абонемент

Також в системі є можливість додавання нових користувачів, цю функцію можуть виконувати тільки системні адміністратори. Для додавання нового користувача системи необхідно перейти по меню програми «Система» –>

«Користувачі», після чого відкриється вікно з можливістю додавання користувача(рис. 2.21).

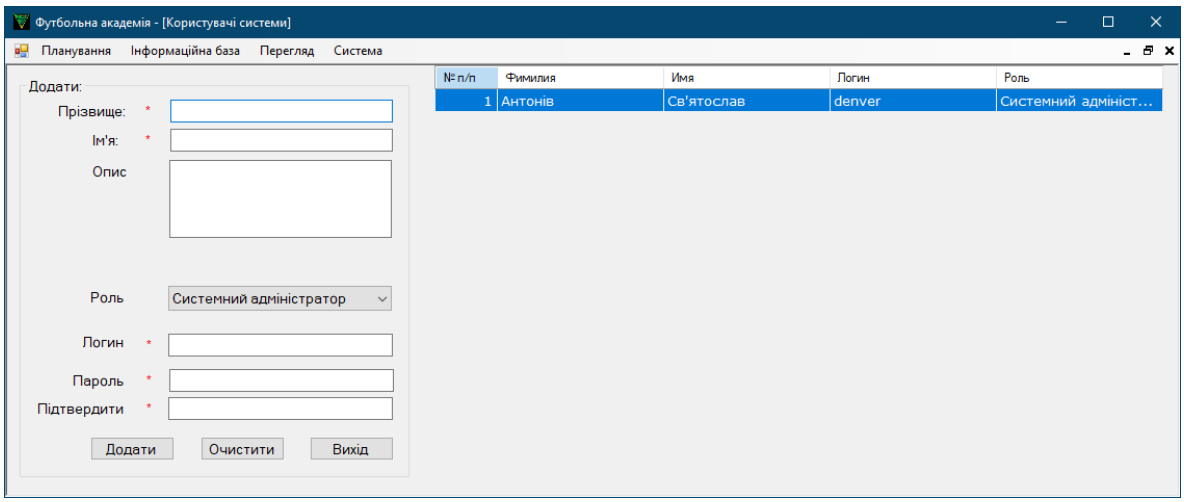

Рисунок 2.21 – Додавання нового користувача системи

Також дані користувача можна змінити, вибравши необхідного із списку облікового запису. Для цього обов’язково необхідно ввести новий пароль та підтвердити його, інакше буде виведене відповідне повідомлення (див. рис. 2.22).

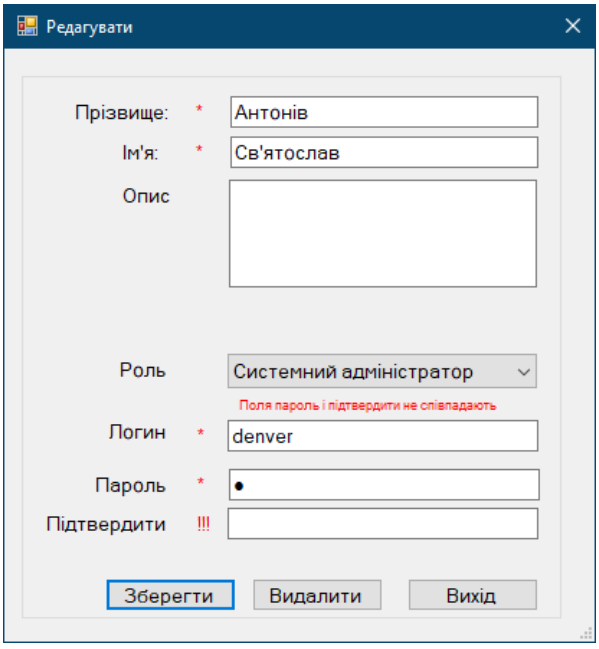

Рисунок 2.22 – Повідомлення про не співпадіння паролю

Ще однією досить важливою функцією є історія змін в системі, за допомогою неї можна бачити активність користувачів, та внесені ними зміни в систему. Вона є доступною лише для системного адміністратора. Щоб потрапити

в вікно цієї функції потрібно перейти за наступним шляхом «Система» –> «Системний журнал». В цьому вікні виводяться всі події системи (2.23).

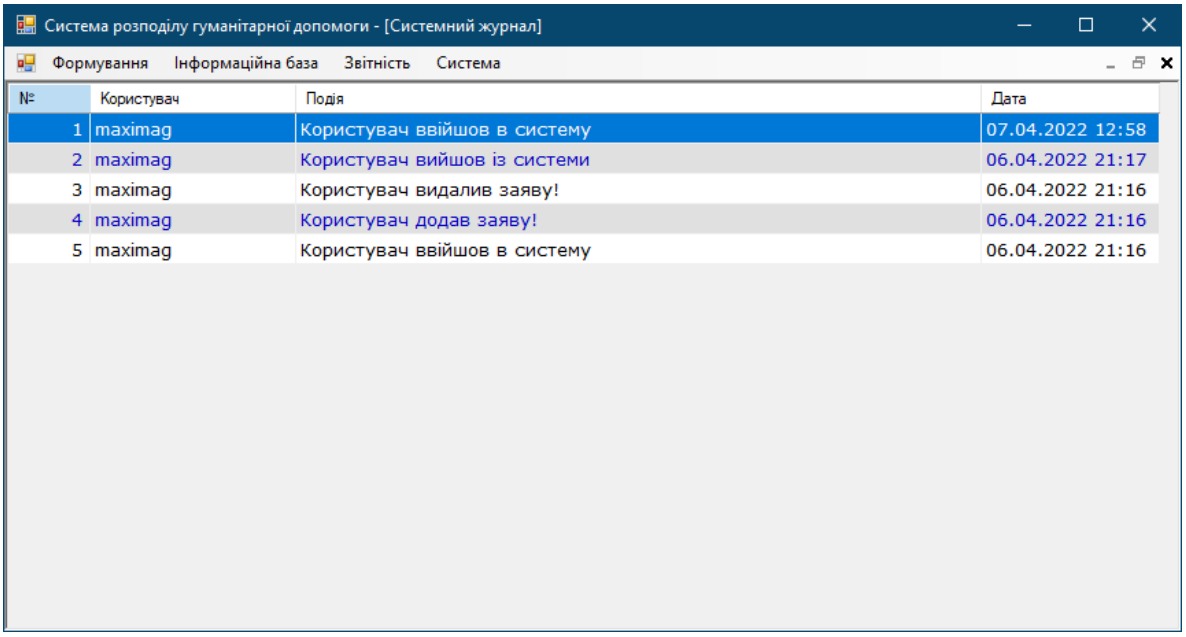

Рисунок 2.23 – Вікно «Системний журнал»

Потрібно відзначити, що дана система не є занадто багатою на набір різних функцій. Але це одночасно є і її перевагою, адже вона є зрозумілоб для використання, і процес ознайомлення з новим продуктом майже не забирає додаткового часу і не відлякуж користувачів через перегружений інтерфейс.

Під час тестування програми не було виявлено жодних помилок системи.

2.4 Технічне та системне забезпечення розробки

2.4.1 Розрахунок та визначення топології комп’ютерної мережі

При розрахунку та визначення топології комп’ютерної мережі небхідно враховувати розташування та розміри кімнат приміщення футбольної академії.

Будівля академії складається з восьми кімнат:

1. Кімната, яка призначається для розміщення ігрового інвентарю (м’ячі, фішки, форма). Вона має розміри: 15х8,3 м. Ця кімната не містить вікон і знаходиться в підвалі академії.

2. Кімната бухгалтера. Розміри кімнати: 16х9,40 м. Кабінет, який має 2 вікна.

3. Кабінет директора. Розміри кімнати: 6,41х3,2 м. Також кімната містить одне велике вікно та один вхід.

4. Серверна кімната, в якій розташовані сервери підприємства для організації роботи комп'ютерної мережі. Розміри кімнати: 6,41х3,2 м. Кімната містить один вхід.

5. Зал для проведення нарад. Розміри залу: 6,41х3,02 м. Кімната містить один вхід.

6. У шостій кімнаті розташований адміністративний відділ. Розміри кімнати: 11,66х7,5 м. Кімната містить одне велике вікно та один вхід.

7. Кімната «Обслуговуючого персоналу». Розміри кімнати: 14х6,57 м. Кімната містить три малих та одне велике вікно. В кімнату є тільки один вхід.

8. У восьмій кімнаті розташовано роздягалку для гравців. Кімната містить 4 середні вікна та один вхід.

9. Між кімнатами розташований коридор.

Під час аналізу плану будівлі та проведення мережі було отримано такі дані:

1. Види мережного обладнання, яке буди використовуватись для забезпечення обміну інформаіцєю:

a) насамперед це системи безпеки;

b) локальна обчислювальна мережа;

d) інше.

2. Вимоги до характеристик продуктивності системи:

a) пропускну здатність;

b) ємності підсистеми внутрішніх та зовнішніх магістралей;

c) перспективи розширення системи;

d) інше.

3. Способи прокладання кабелів. Можливість впровадження системи без надмірної шкоди інтер’єру академії.

4. Вимоги щодо сумісності з обладнанням, яке передбачається встановити у приміщенні.

5. Інші вимоги.

Проблема з якою певну всі найбільше стикаються в подібних мережах – це швидкість інтернету. Для запобігання цьому будть встановлені точки доступу, які передбачають більшу пропускну здатність.

Зазвчиай, це виникає через неправильне налаштування мережі. Що в свою чергу веде до послаблення сигналу Wi-Fi.

Також, це проблема може ще виникати через високий рівень перешкод від бездротового обладнання. Оскільки, частоти в яких діє Wi-Fi, мають дуже мале вікно, через це є вірогідність що пристрої почнуть конфліктувати між собою.

Такий рівень суперечності найчастіше є наслідком не дуже успішного планування мережі. Оскільки, в такому випадку мережам, які знаходяться по сусідству присвоєні однакові (або близькі до однакових) діапазони роботи, за рахунок цього вони і конфліктують, сигнали однієї мережі суперечаться з сигналами іншої.

Звісно, це також може бути визвано якщо передавач мережі користувача має низьку потужнність. Оскільки, зазвичай, передавачі гаджетів є слабшими, тому вони можуть бачити точку доступу до якої хочуть підключитись, а вона в свою чергу неспроможна їх помітити.

На даний час, велике частина несправностей з Wi-Fi зв’язана з неправильною оцінкою навантаження або проблема в не підходящому розміщені точок доступу, тому всі ці нюанси потрібно враховувати заздалегідь.

Отже, щоб не стати учасниками цих проблем на етепі плануванняя мережі потрібно звернутись до додаткових інтсрументів. Тут мова йде про програми для планування бездротових мереж, їхні переваги в тому, що працюють з різними форматами креслень приміщень та реалістично оцінюють якість сигналів, також не менш важливим є те, що вони є простими в застосуванні.

Після цього було спроектовано модель мережі у програмі моделювання роботи мережі Cisco Packer Tracer, основне вікно якої наведено на рис. 2.24.

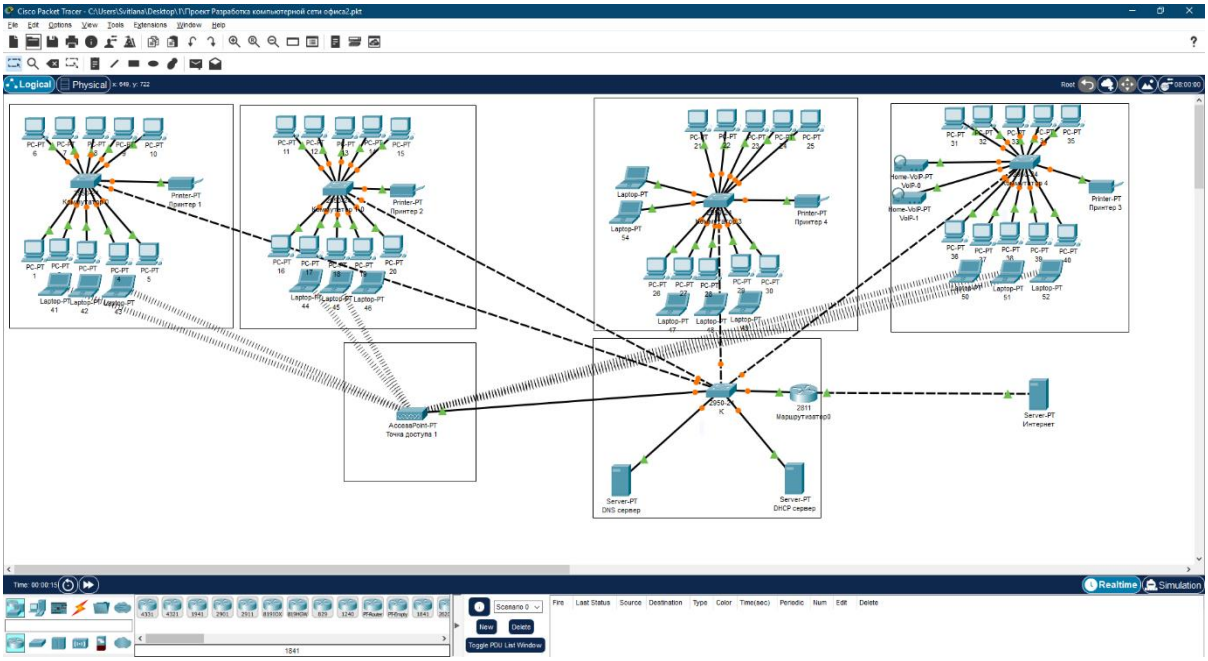

Рисунок – 2.24 – Топологія комп’ютерної мережі футбольної академії

На даному малюнку продемонстрована мережа, яка відповідає вимогам безпеки. У ключових місцях системи передачі даних встановлені маршрутизатори Cisco.

До параметрів конфігурації робочої станції відносяться параметри вказівки IP-адреси та маски підмережі, якщо не встановлено режим динамічного розподілу адрес. Параметри конфігурації робочої станції показані рисунку 2.25.

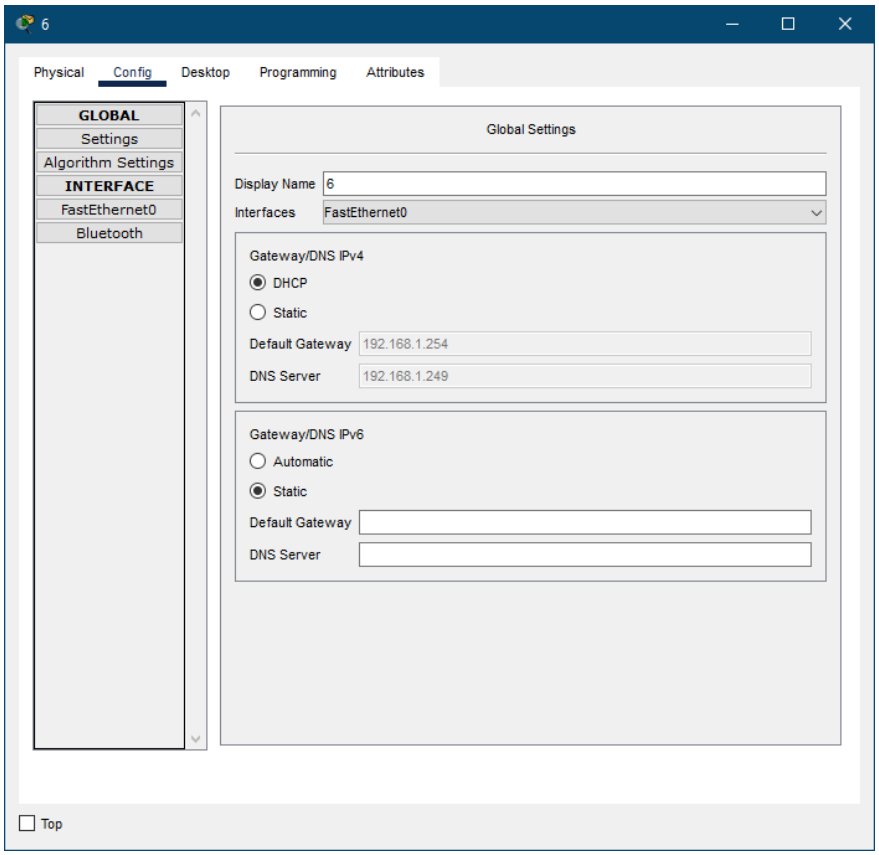

Рисунок 2.25 – Динамічне задання адресації

Проводити дані налаштування можна через розширений інтерфейс користувача, так і через командний рядок терміналу обслуговування системи IOS комутатора.

На малюнку 2.26 наведено приклад налаштування мережного з'єднання між комутатором і хостом, що входить до мережі обслуговування даного комутатора.

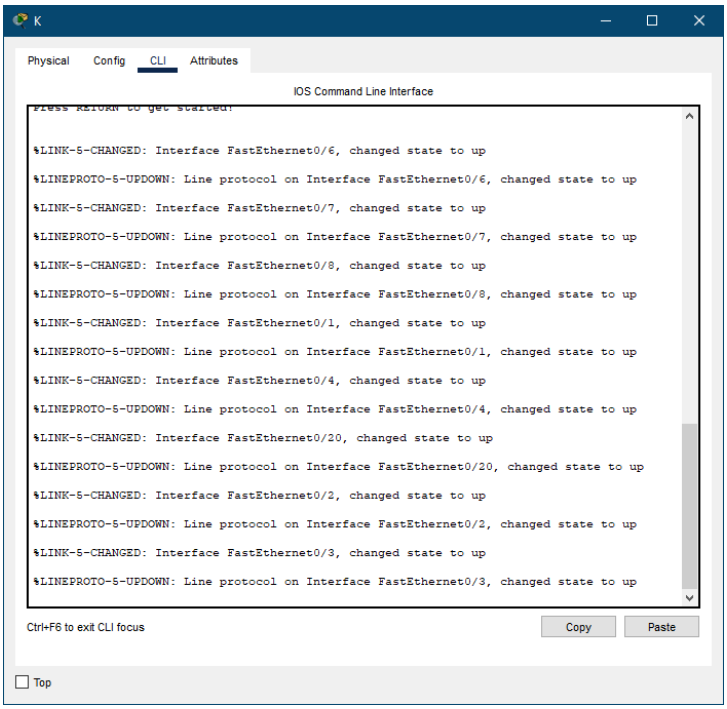

Рисунок 2.26 – Налаштування роботи комутатора

На рисунку 2.27 проводиться перевірка стану з'єднання за допомогою команди ping.

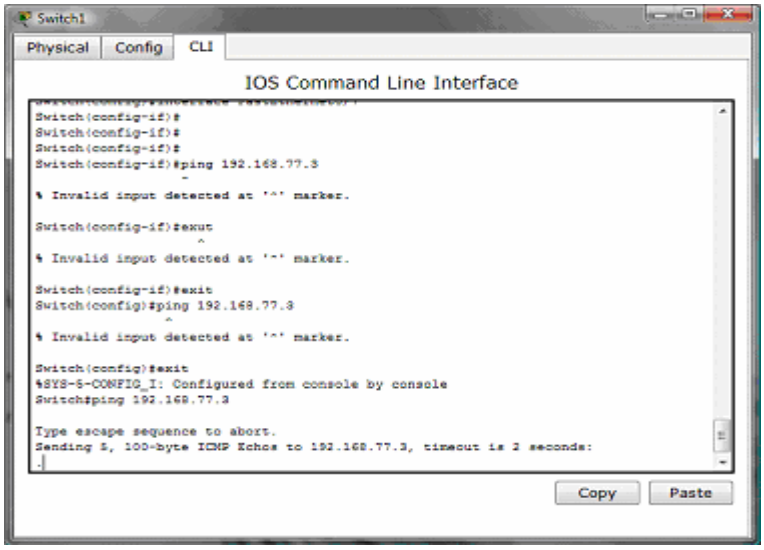

Рисунок 2.27 – Перевірка стану з'єднання

2.4.2 Обґрунтування вибору ОС та протоколу обміну даними

Для функціонування системи було обрано наступне системне забезпечення:

– Підтримка роботи обраної СУБД MySQL;
– Незалежність від типу апаратної архітектури;
– Підтримка багатопроцесорної обробки даних;
– Налжна швидкість мережі для нормально функціонування;
– Забезпечення надійності роботи;
– Загальна поєднуваність з іншими видами ПЗ;

Цим вимогам для ОС робочої станції відповідають Windows Server, Windows 10 та Unix.

Результати порівняльного аналізу параметрів ОС можна як таблиці 2.11.

Таблиця 2.11 – «Порівняльні характеристики ОС»

Вимога	**Windows Server**	**Windows 10**	**Unix**
Незалежність від типу апаратної архітектури	+	+	–
Підтримка роботи з обраною СУБД	+	+	–
Хороша мережева швидкодія	+	–	+
Надійність	+	+	+
Зручний інтерфейс користувача	+	+	–
Велика кількість утиліт та підтримуй створюваних програмних продуктів	+	+	–
Висока вартість	+	+	–

За результатами із таблиці 2.8 для розроблюваної ІС найращим варіантом може бути обрана систмеа Windows Server, оскільки вона максимально повно відповідає заданим вимогам.

2.4.3 Розробка і обґрунтування стратегії адміністрування системи

Задля належної роботи мережі налаштуємо DHCP та DNS сервери.

DNS (система доменних імен) відповідає за відповідність доменних імен та IP-адрес. Доменне ім'я - це не тільки адреса в інтернеті, наприклад selectel.ru, але також ім'я комп'ютера в локальній мережі, наприклад Director PC. DNS проводить з'єднувальну лінію між IP та літерно-числовим доменним ім'ям комп'ютера або веб-сайту. DHCP займається виділенням та призначенням IP з області. Очевидно, що два протоколи мають тісно взаємодіяти між собою.

DHCP — це стандартний протокол прикладного рівня, який дозволяє комп'ютерам автоматично отримувати IP-адресу та інші параметри, необхідні для роботи в мережі. Для цього комп'ютер звертається відповідно — до DHCP-сервера. Мережевий адміністратор може задати діапазон адрес, які будуть розподілені між комп'ютерами.

DHCP-сервер має певну базу IP-адрес, ці адреси можна розподіляти між користувачами. DNS-сервер відповідає за те, щоб співставити IP-адреси та доменні імена.

Ця взаємодія між DHCP і DNS необхідня для того, щоб DNS-сервер мав змогу отримувати актуальну інформацію про новий IP і міг з’єднати його з ім’ям клієнта в мережі.

Налаштувати цю взаємодію можна за чотири кроки за допомогою пакету dnsmasq, доступного у стандартних репозиторіях Ubuntu та Debian.

Для початку потрібно визначитися з комп'ютером, це буде пристрій на в якого буде роль сервера. Тут важливо обирати комп’ютер, який не буде вимикатись занадто часто, адже якщо вимкнути його – то вимкнеться і вся система.

Даному пристрою встановлюється статичний IP. Робиться це редагуванням файлу конфігурації в директорії /etc/network/interfaces.

Установка dnsmasq

Встановимо пакет dnsmasq командою з терміналу:

```
sudo apt-get install dnsmasq -y
```

Рисунок 2.28 – Встановлення пакету dnsmasq

Далі йдемо в файл конфігурації /etc/dnsmasq.conf. Він є досить об’ємним, але містить коментарі які пояснюють вказані там налаштування. Щоб додати потрібні налаштування, відкрийте файл і видаліть решітку (#), що означає коментар, на початку потрібних рядків.

Налаштування фаєрволу

Для зміни налаштування фаєрволу можна використовувати Ubuntu Uncomplicated Firewall. Використовуємо такі команди:

```
sudo ufw allow bootps
sudo ufw allow 53/udp
sudo ufw allow 53/tcp
```

Рисунок 2.29 – Зміна налаштування фаєрволу

Зміна налаштувань роутера

Зайдемо в налаштування роутера з браузера, відключіть DHCP для локальної мережі, змініть всі налаштування DNS так, щоб вони вказували на ваш щойно налаштований сервер. Остання дія – перезапуск мережі на сервері. Для цього ви можете просто перезавантажити комп'ютер або використовувати команди:

```
sudo service dnsmasq restart
sudo service network-manager restart
```

Рисунок 2.30 – Зміна налаштування роутера

Варто зазначити, що DHCP. Що DHCP має проблеми з точки зору кібербезпеки, а саме: процес DORA передбачає розсилку повідомлень і коли сервер DHCP, який перший відгукнеться на розсилку, зможе отримати можливість вказати IP зі своєї області. То ж, якщо зловмисник матиме змогу використовувати свій сервер, відповідно той, який пришле відповідь найшвидше на повідомлення, в такому разі він зможе отримати доступ до дій користувача.

Ще один прокол у безпеці у тому, що DHCP використовує UDP-протокол. А це протокол обміну без встановлення з'єднання, отже, і без шифрування. Інформація, що передається по UDP, не захищена і може бути «підслухана», що також може бути використане зловмисниками.

Не дивлячись на деякі мінуси протокол DHCP є затребуваним і широко використовується в сучасних мережах.

2.4.4 Заходи захисту від несанкціонованого доступу до системи

Для реалізації захисту системи від несанкціонованого доступу було розроблено клас "EncryptData" із використанням простору імен "System.Security.Cryptography". Було вибрано симетричний алгоритми шифрування паролів користувачів. Код класу "EncryptData" приведений в додатках.

Є кілька способів формування блокових шифрів. Найбільш простий і інтуїтивно зрозумілий спосіб полягає в тому, щоб розбити вихідний текст на блоки відповідного розміру, а потім окремо кожен блок шифрує перетворення. Такий режим використання блокових шифрів називають електронною кодовою книгою (ECB – electronic codebook). Його головний недолік полягає в тому, що однакові блоки вихідного тексту при шифруванні дадуть однакові блоки шифр-тексту, а це може суттєво полегшити противнику завдання злому. Тому режим ECB не рекомендується використовувати при шифруванні текстів, що за довжиною перевищують один блок.

За замовчуванням у CryptoAPI блокові шифри використовуються як зчеплення блоків шифр-тексту (CBC — cipher block chaining). У цьому режимі при шифруванні черговий блок вихідного тексту спочатку комбінується з попереднім блоком шифр-тексту (за допомогою побітового АБО, що виключає), а потім отримана послідовність бітів надходить на вхід блочного шифру. Блок шифр-тексту, що утворюється на виході, використовується для шифрування наступного блоку. Перший блок вихідного тексту також повинен бути скомбінований з деякою послідовністю бітів, але «попереднього блоку шифр-тексту» ще немає; тому режими шифрування із зворотним зв'язком вимагають використання ще одного параметра - він називається ініціалізуючим вектором (IV - initialization vector). IV - несекретне двійкове значення, розмір якого дорівнює довжині блоку шифру. Щоб створити новий ключ, потрібно викликати метод GenerateKey, а для вектора ініціалізації — метод GenerateIV. Наприклад, для алгоритму RC2, який підтримується базовим криптопровайдером Microsoft, розмір блоку становить 64 біти (8 байтів).

ВИСНОВКИ

Даний проект розроблений для підтримки діяльності футбольної академії «Прикарпаття». Розробку інформаційної системи було виконано у середовищі Microsoft Visual Studio 2019 при використанні мови програмування C # та СУБД MySQL. Дана інформаційна система повинна значно полегшити роботу адміністраторів та тренерів футбольної академії. Вона має зручний інтерфейс для перегляду даних, додавання та вилучення даних та виконання пошуку та фільтрації потрібних записів за допомогою запитів.

У розробленому продукту реалізовано такі функціональні можливості:

– можливість додавати, редагувати та видаляти тренерів;

– можливість додавати, редагувати та видаляти даних про спортсменів;

– можливість додавати, редагувати та видаляти дані про інвентар;

– можливість створення списку необхідного інвентаря для кожного заняття;

– можливість формування розкладів тренувань;

– можливість додавання абонементів;

– можливість перегляду планів тренувань.

Дану програму можна використовувати для футбольних академій, що значно полегшить їхню роботу.

У першому розділі диплома було дано характеристику футбольній академії «Прикарпаття» та побудовано її організаційну структуру. Виконано концептуальне моделювання за допомогою програми All Fusion Process Modeler r7, у результаті яких було виявлено недоліки та методи їх усунення. Визначено мету та призначення автоматизованого варіанту. Здійснено аналіз існуючих розробок шляхом визначення критеріїв порівняння та розбір за ними вибраних програмних продуктів.

У другій частині роботи було зроблено опис комплексу задач автоматизації, а саме:

– Інформаційне забезпечення системи;

– Алгоритмізація та реалізація комплексу задач автоматизації;
– Інструкція користувача;
– Технічне та системне забезпечення розробки.

В результаті роботи розроблено програмне забезпечення для забезпечення діяльності футбольної академії «Прикарпаття» і таким чином визначеної мети досягнуто.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Організаційна структура футбольної академії [Електронний ресурс] Режим доступу: https://zz.te.ua/u-ternopoli-zapratsiuvala-futbol-na-akademiia/.

2. М’якшило О.М. Проектування інтерфейсу користувача: Конспект лекцій з дисципліни “Автоматизоване проектування комп‘ютерних систем” для студ. спец 7.080401 напряму підготовки 0804 “Комп‘ютерні науки” ден. та заоч. форм навчання – К.: НУХТ, 2006 – 67с.

3. Платформа YOPLAYDO [Електронний ресурс] Режим доступу: https://yoplaydo.ru/

4. Програма ProTrainUp [Електронний ресурс] Режим доступу: https://protrainup.com/uk

5. Алексей Федоров. Visual Studio 2010 — Первое знакомство -2010

6. Пол Дейтел, Харві Дейтел. Как программировать на Visual C# 2019

7. Нейгел К., Ивьен Б.. Professional C# 5.0 and .NET 4.5

8. Pro C# 7: With .NET and .NET Core 8-th edition / Andrew Troelse, Philip Japikse. –2017

9. Об'єктно-орієнтоване програмування [Електронний ресурс] Режим доступу: http://ruslan.rv.ua/python-essential/oop/oop_basis/

10. Об'єктно-орієнтований підхід до програмування [Електронний ресурс] Режим доступу: http://www.znannya.org/?view=csharp-oop

11. Офіційна документація .NET [Електронний ресурс] Режим доступу: https://docs.microsoft.com/ru-ru/dotnet/

12. Мова програмування C# і платформа .NET Core [Електронний ресурс] Режим доступу: https://metanit.com/sharp/tutorial/1.1.php

13. What's new in .NET 4.7 [Електронний ресурс] Режим доступу: https://docs.microsoft.com/en-us/dotnet/core/dotnet-five

14. .NET Core/5+ vs. .NET Framework for server apps [Електронний ресурс] Режим доступу: https://docs.microsoft.com/en-us/dotnet/standard/choosing-core-framework-server

15. .NET 4.7 [Електронний ресурс] Режим доступу: https://temofeev.ru/info/articles/predstavlyaem-net-5/

16. Джон Скит, C# для професіоналів: тонкощі програмування, 608 ст. ISBN 978-5-8459-1909-0, 978-1-617-29134-0; 2014, Вільямс.

17. Литвинов О.А., Карпенко Н.В. Тестування інформаційних систем: модульне, інтеграційне, системне [Текст] – Д.: Ліра, 2016. – 283 с.

18. Литвинов О.А., Герасимов В.В., Карпенко Н.В. Об’єктно-орієнтована розробка інформаційних систем [Текст] – Д.: Ліра, 2018. – 448 с.

19. Програма для роботи з базами даних Microsoft Access: [Електронний ресурс] // Режим доступу: https://www.microsoft.com/uk-ua/microsoft-365/access.

20. Сайт для побудови ER-діаграм : [Електронний ресурс] // Режим доступу: https://app.diagrams.net/.

21. Архітектурні шаблони: [Електронний ресурс] // Режим доступу: https://devzone.org.ua/post/nayvazhlivishi-arkhitekturni-shabloni-yaki-neobkhidno-znati.

ДОДАТКИ

Додаток А. Організаційна структура підприємства футбольнії академії «Прикарпаття»

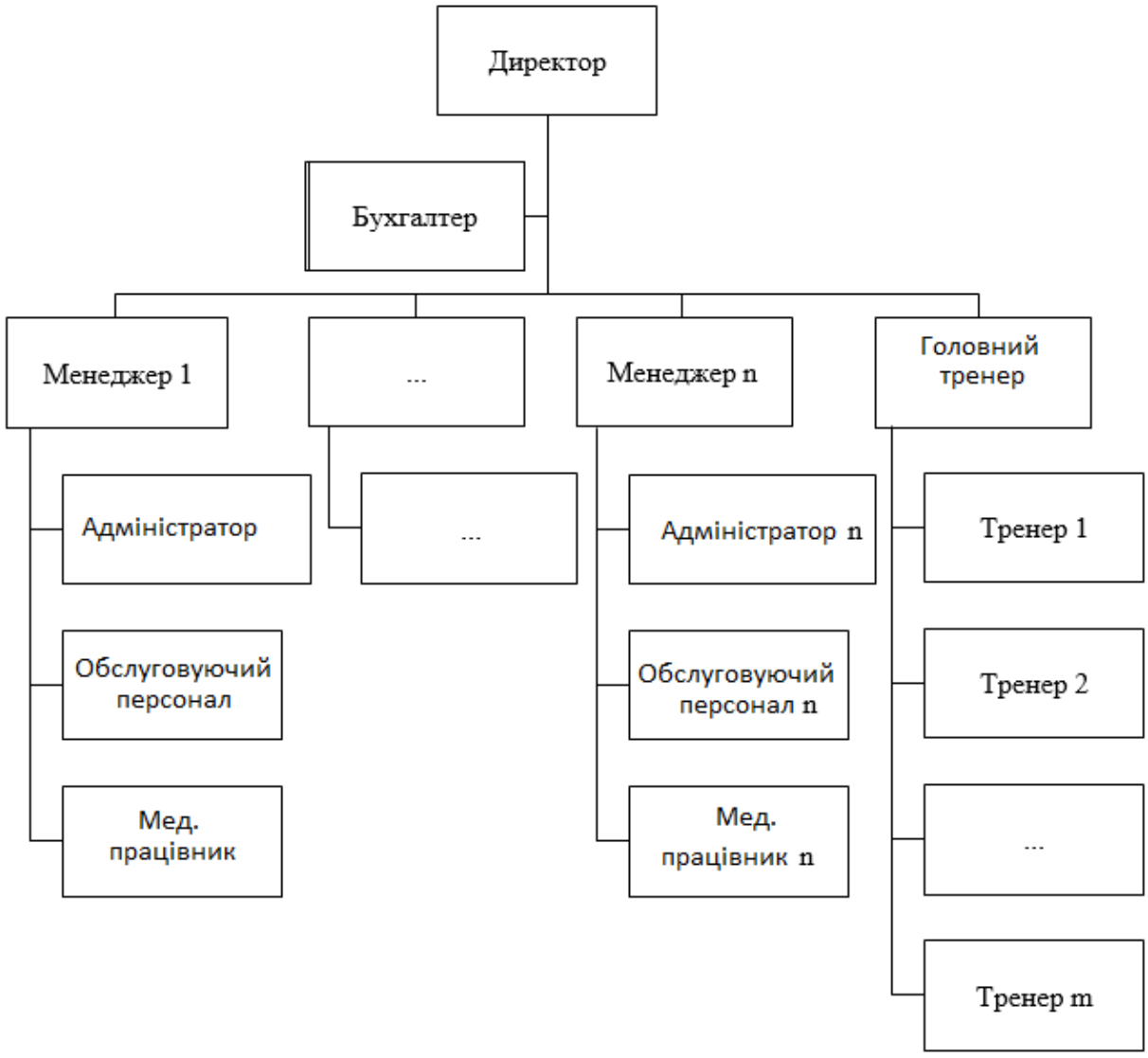

Рисунок 1 – Організаційна структура підприємства футбольнії академії «Прикарпаття»

Додаток Б. Функціональні моделі

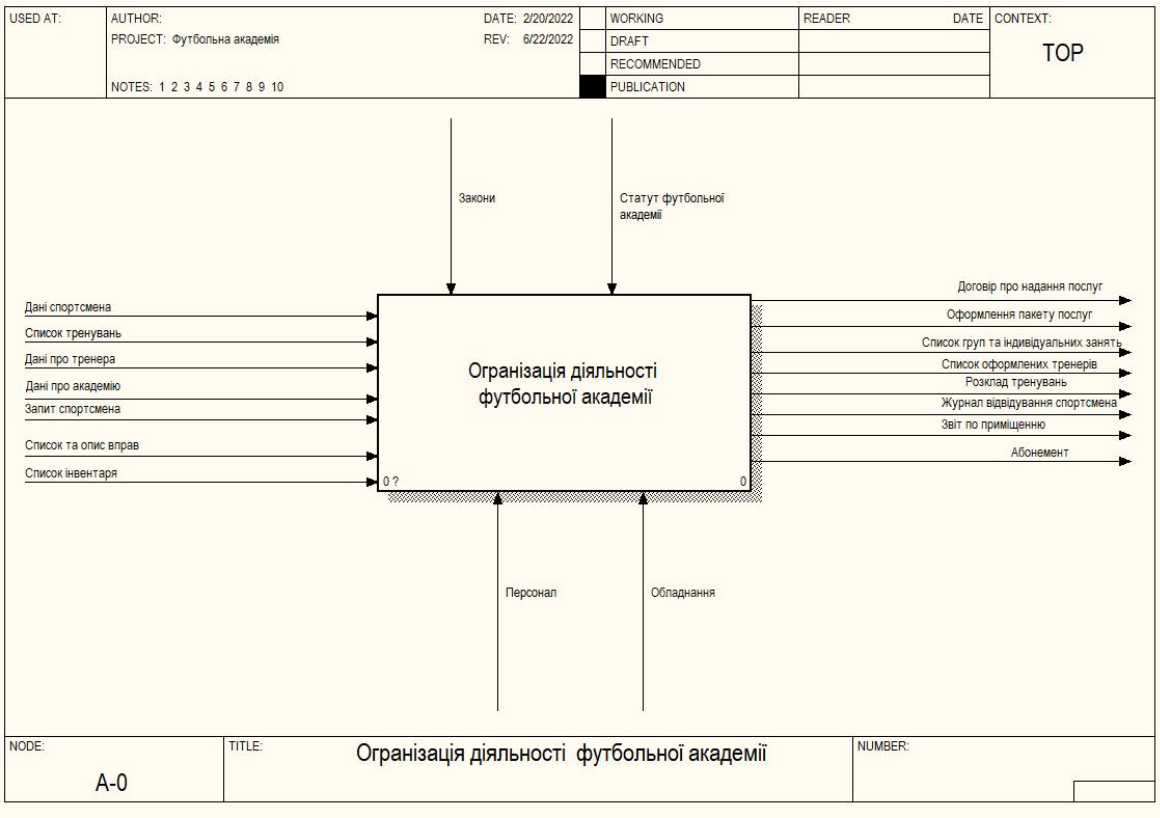

Рисунок 2 – Контекстна діаграма діяльності IDEF0 футбольної академії

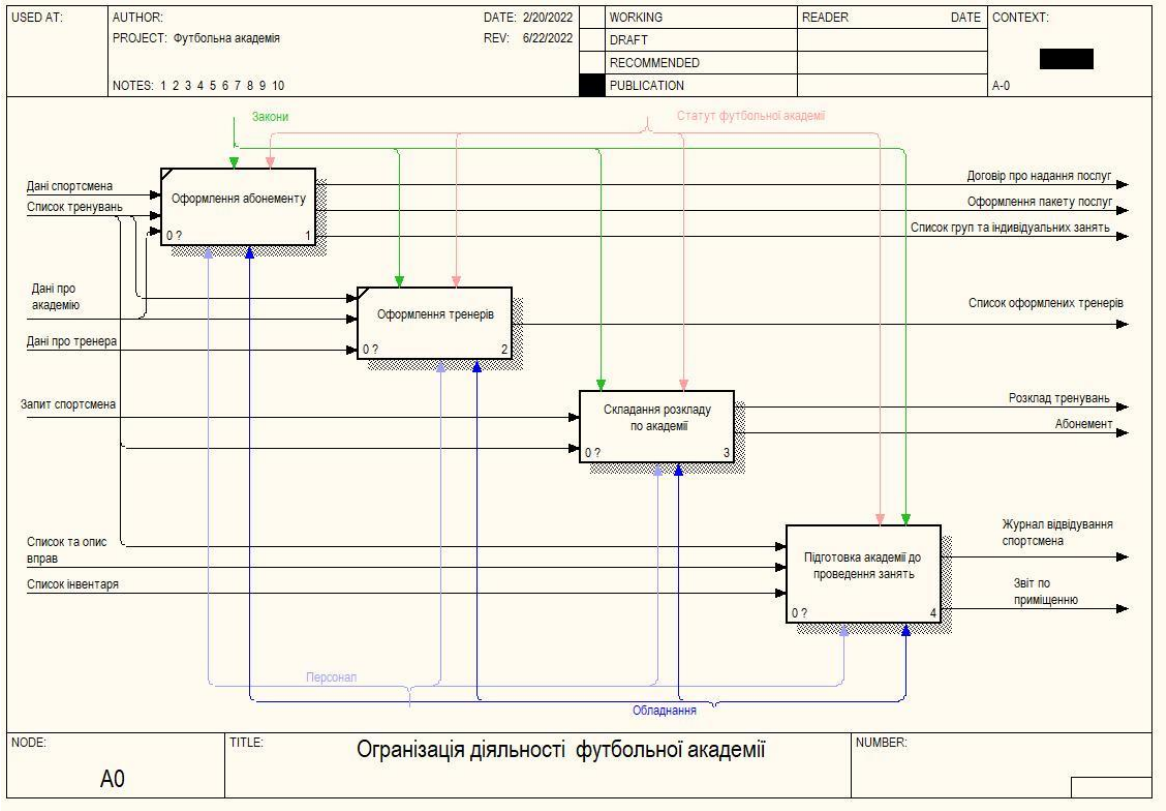

Рисунок 3 – Організація тренувань футбольної академії

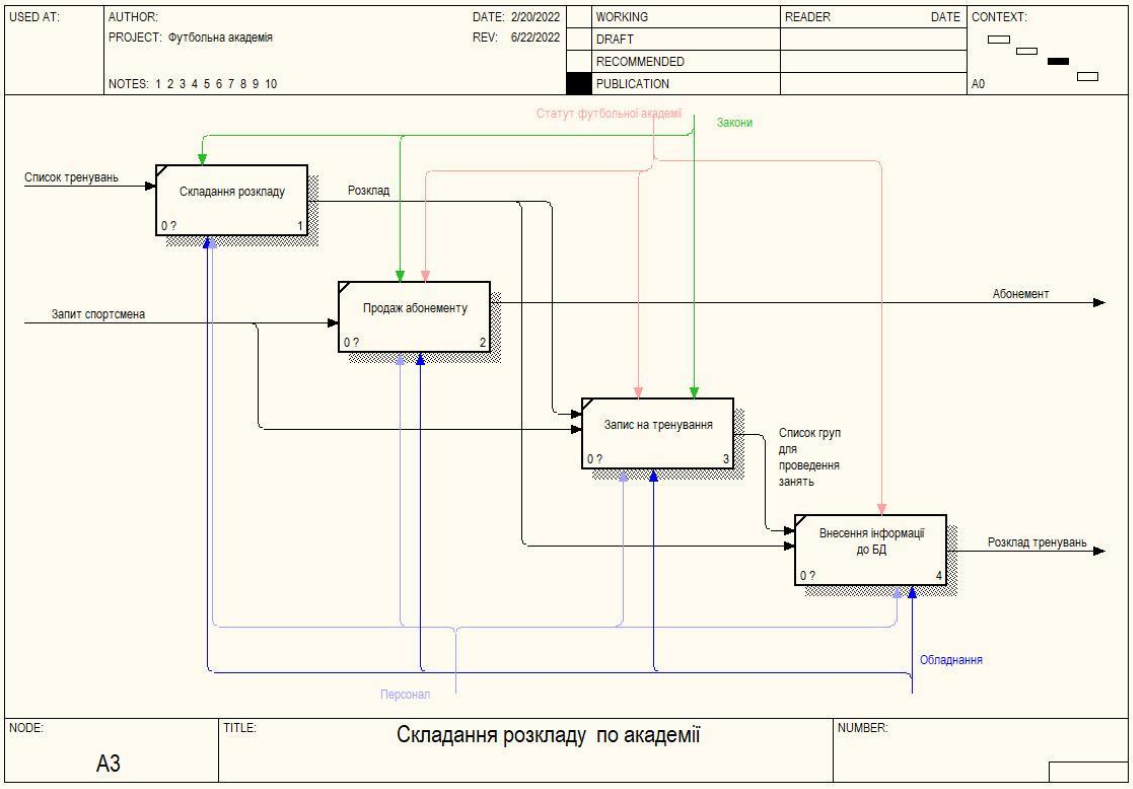

Рисунок 4 – Декомпозиція підпроцесу «Складання розкладу по академії»

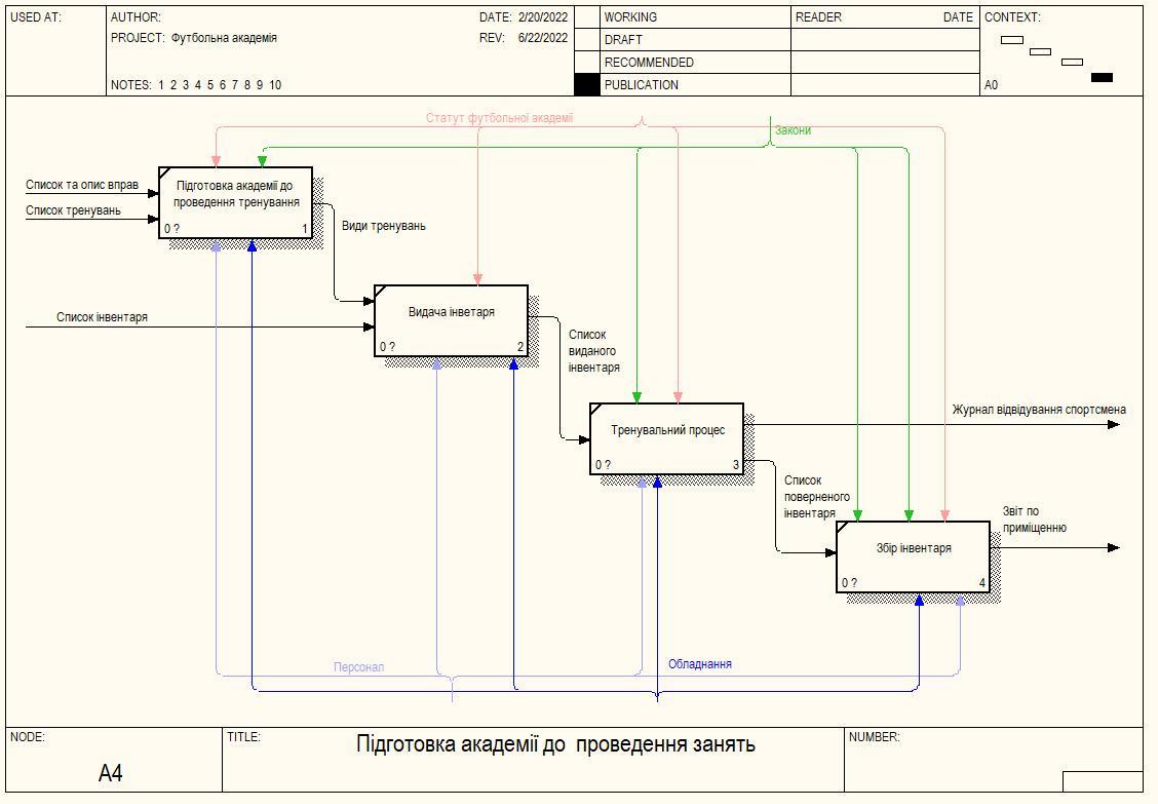

Рисунок 5 – Декомпозиція підпроцесу "Проведення занять"

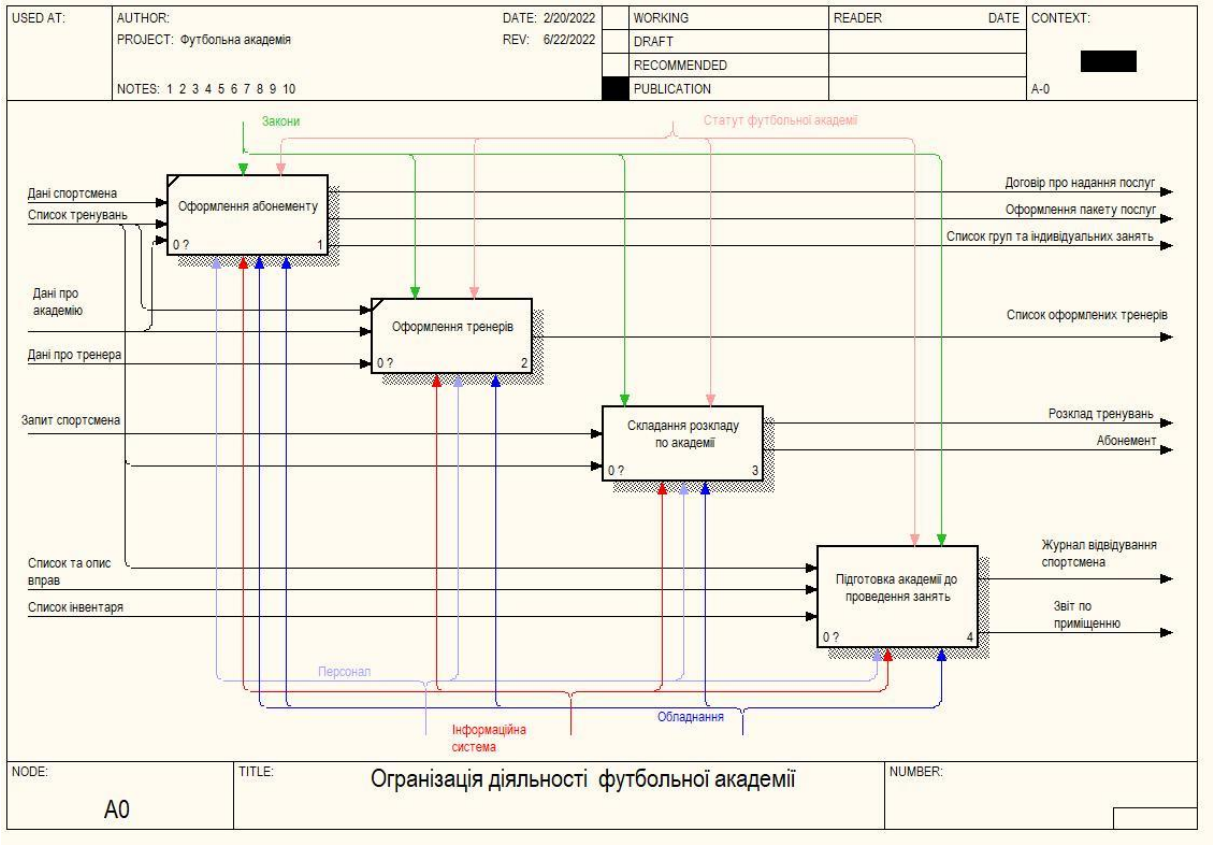

Рисунок 6 – Декомпозиція контекстної діаграми футбольної академії

Додаток В. Моделі та схеми бази даних

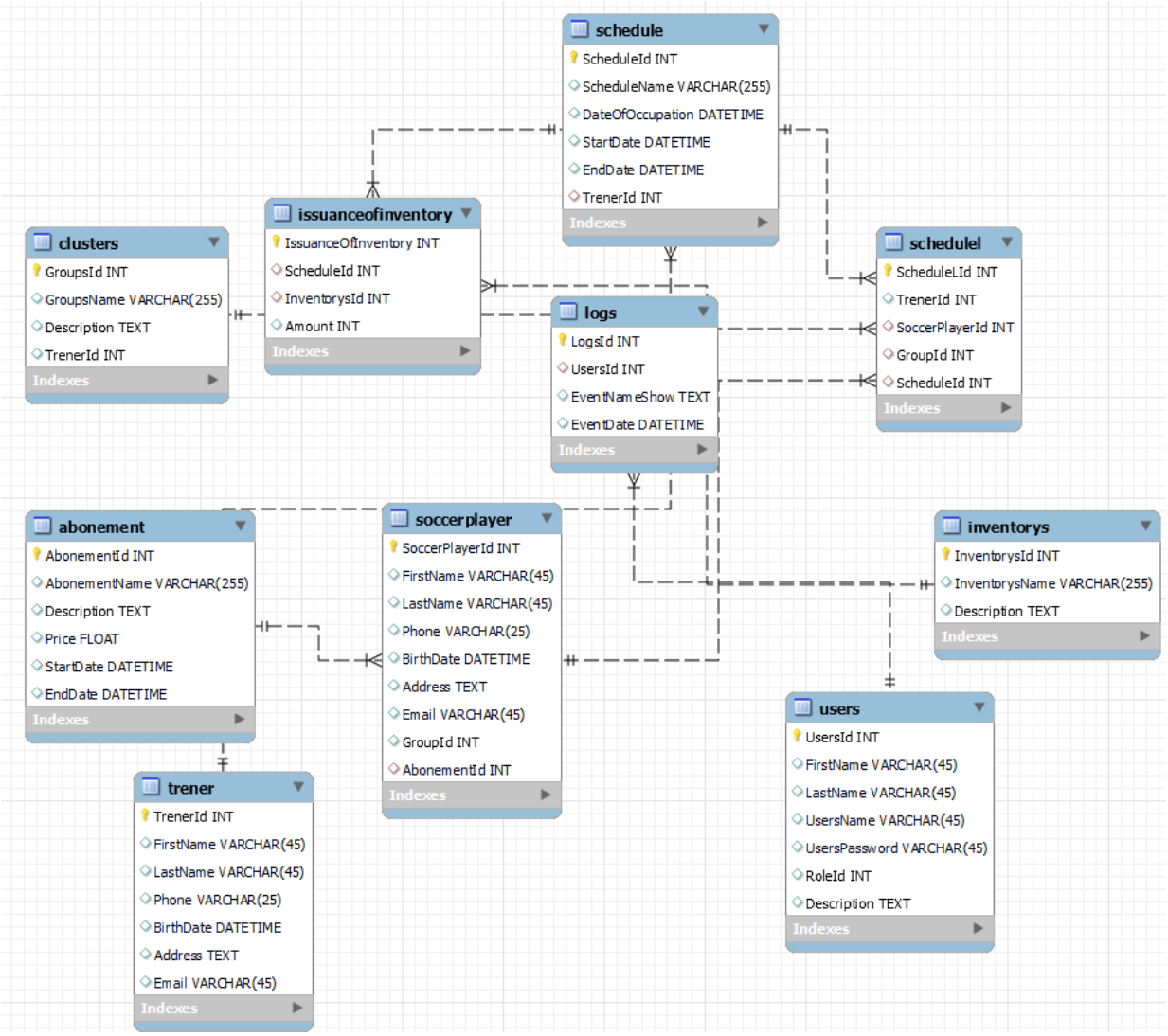

Рисунок 7 – Фізична модель бази даних

Додаток Г. Схема комп'ютерної мережі

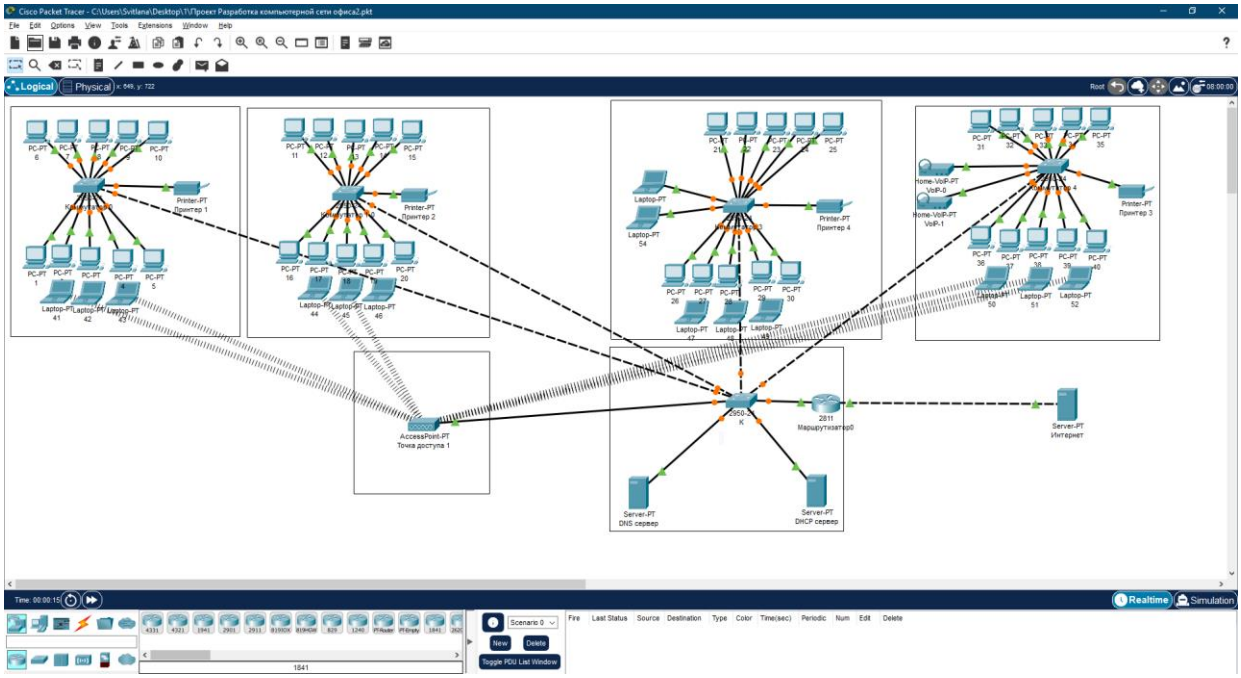

Рисунок 8 – Схема комп’ютерної мережі

Додаток Д. Скрін-шоти розробленої програми

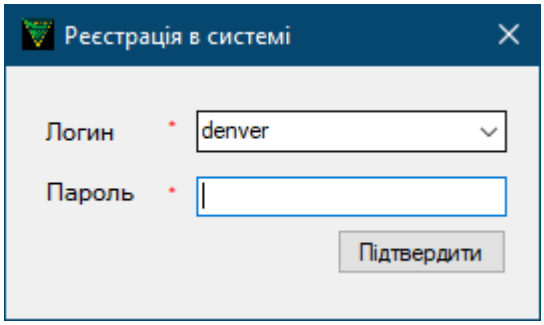

Рисунок 9 – Реєстрація користувача в системі

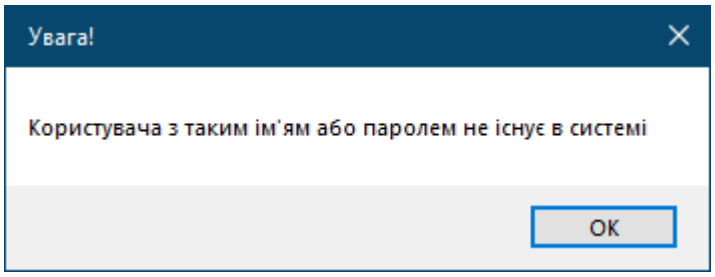

Рисунок 10 – Попередження про неправильне введення даних

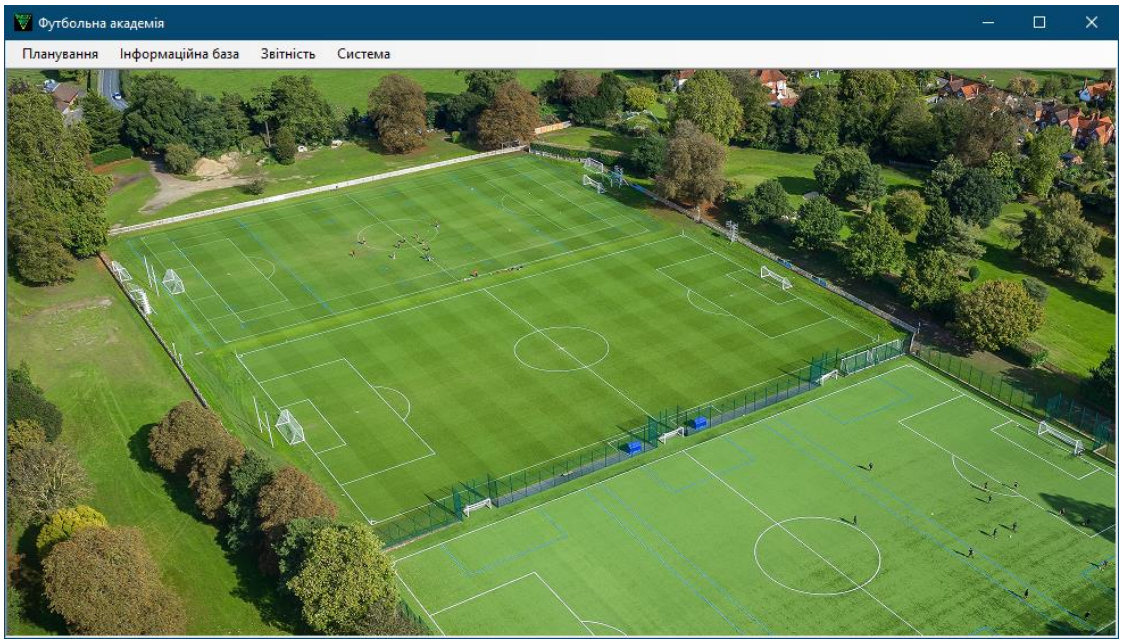

Рисунок 11 – Головне вікно програми

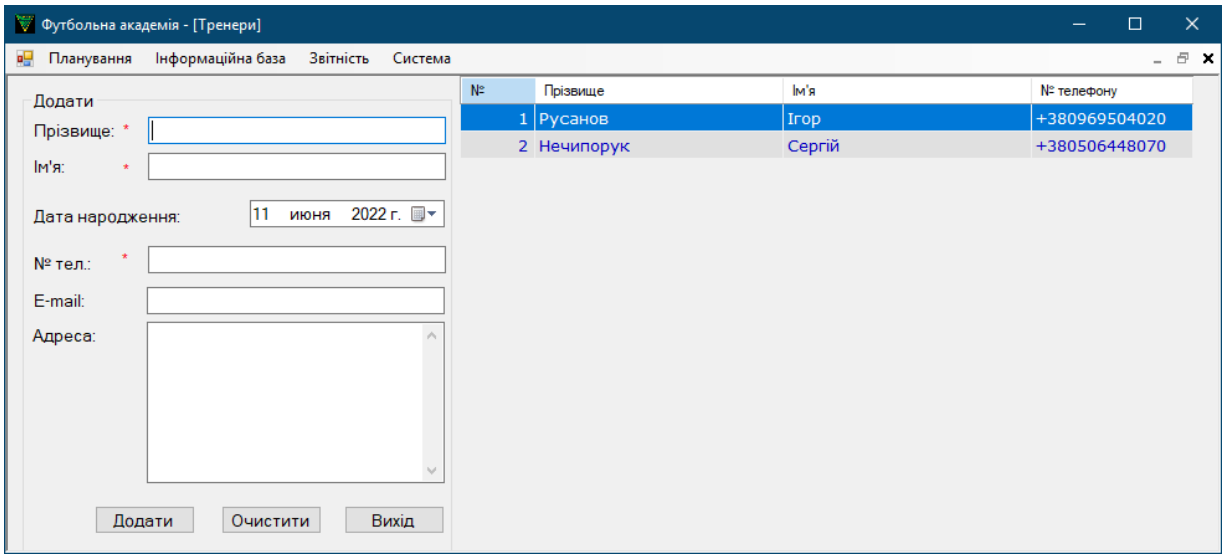

Рисунок 12 – Вікно для опрацювання даних про тренерів

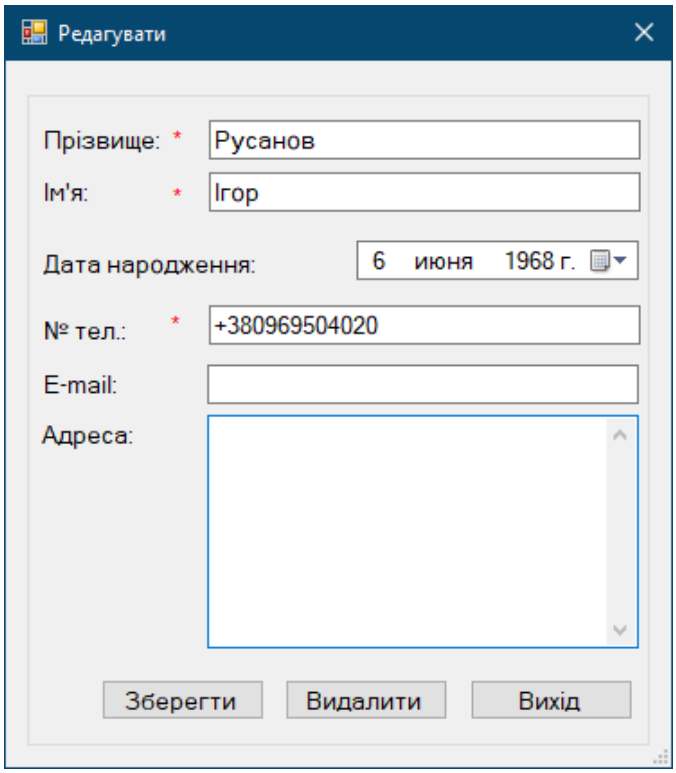

Рисунок 13 – Вікно для редагування даних вибраного тренера

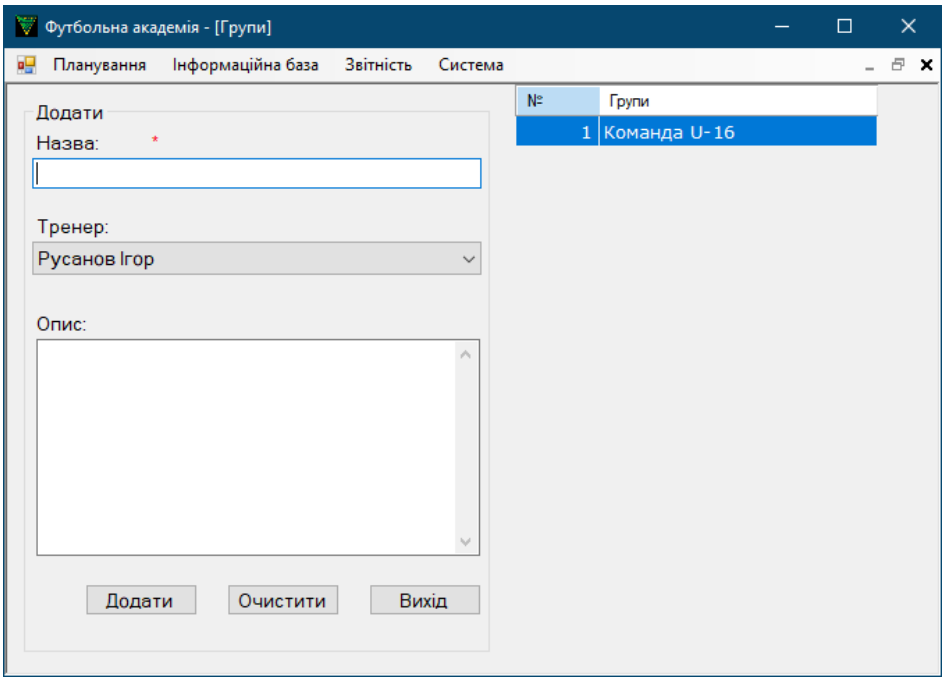

Рисунок 14 – Вікно для додавання нової групи

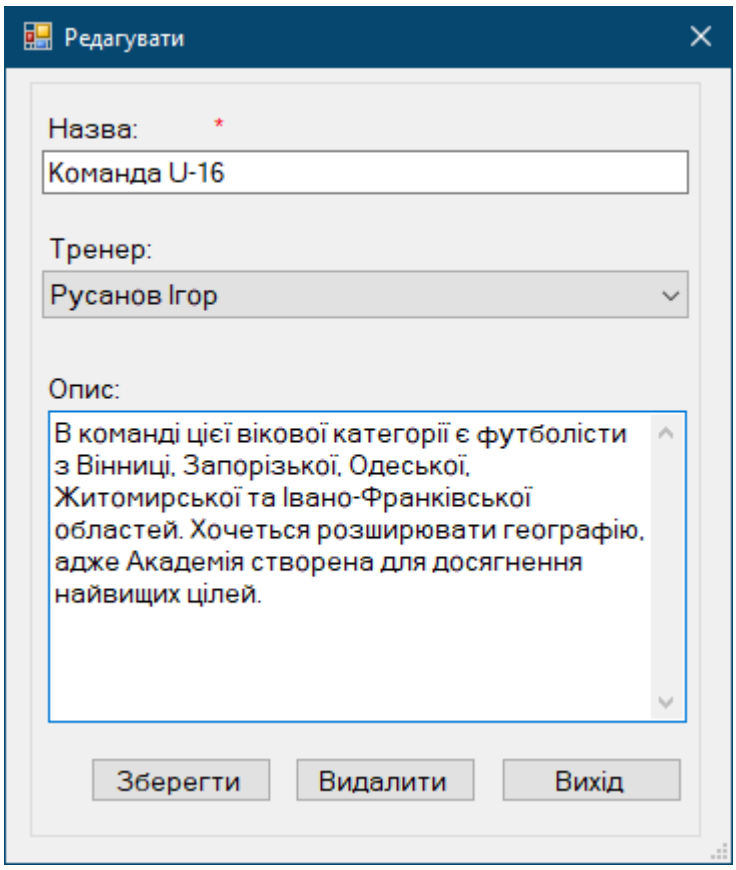

Рисунок 15 – Вікно для редагування даних групи

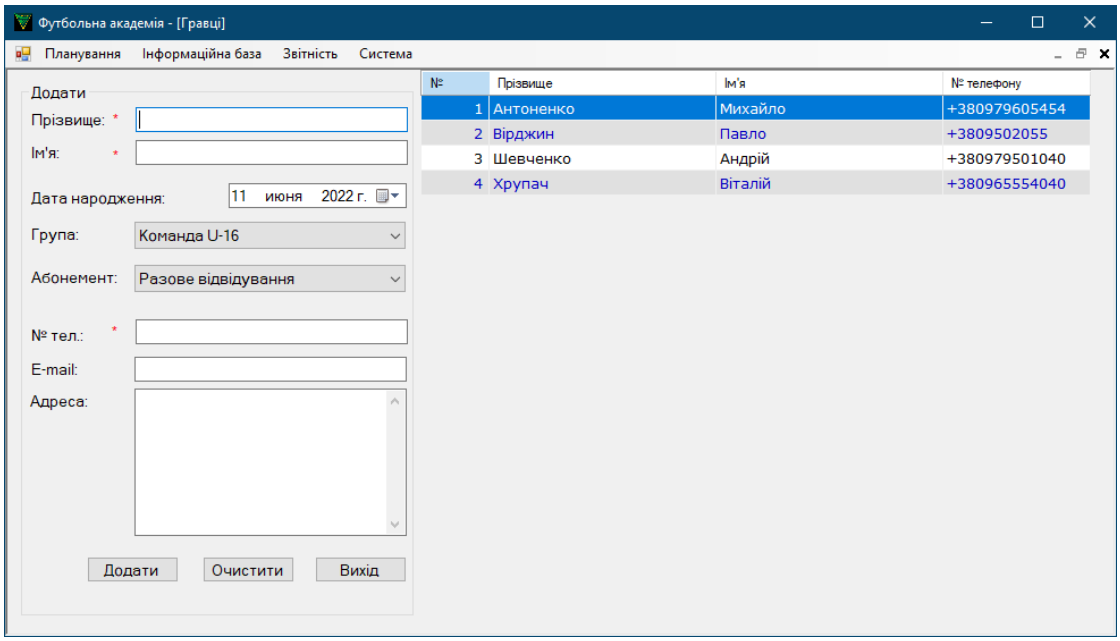

Рисунок 16 – Вікно для опрацювання інформації про гравців

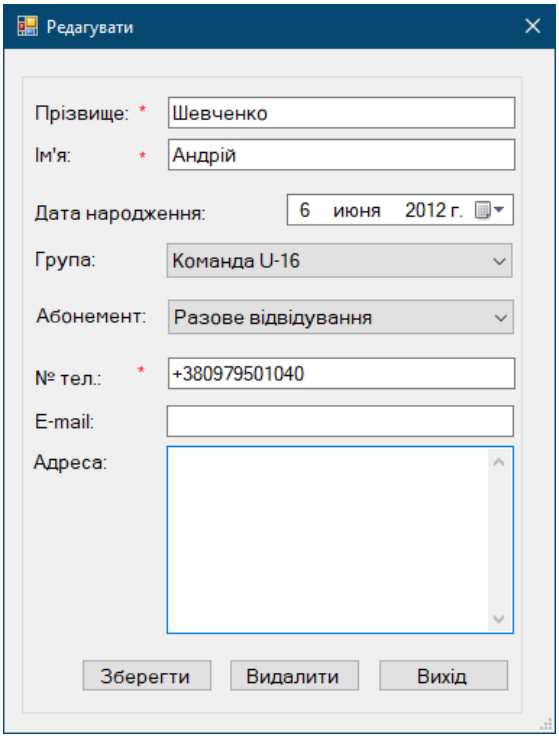

Рисунок 17 – Вікно для редагування інформації про гравця

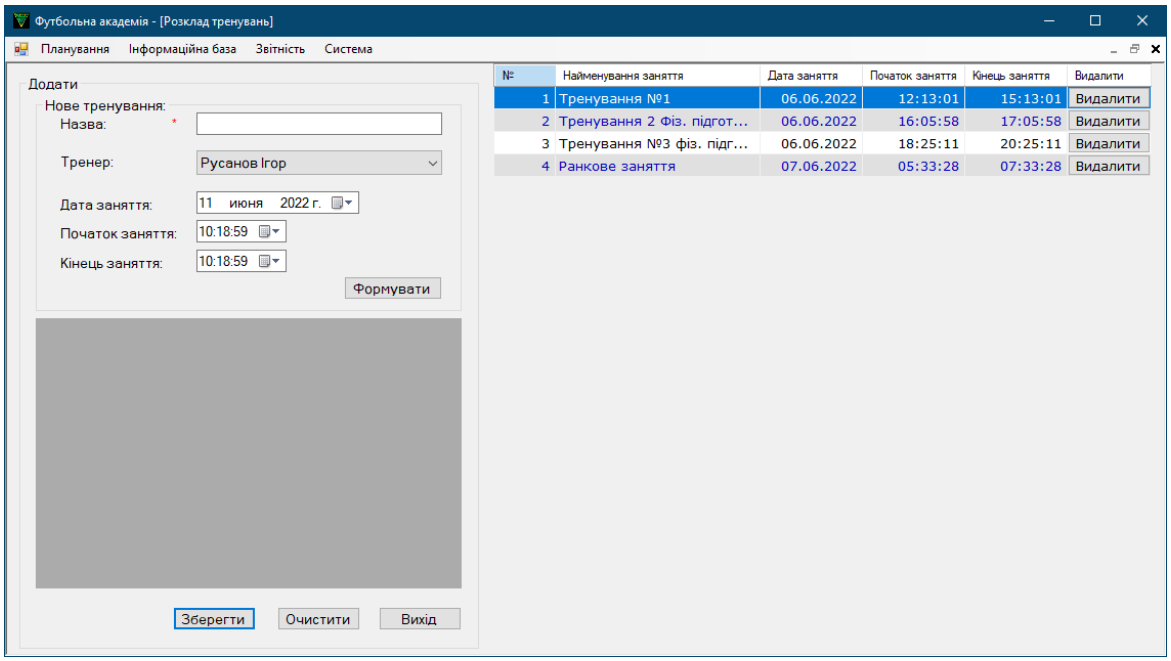

Рисунок 18 – Вікно для планування розкладів занять

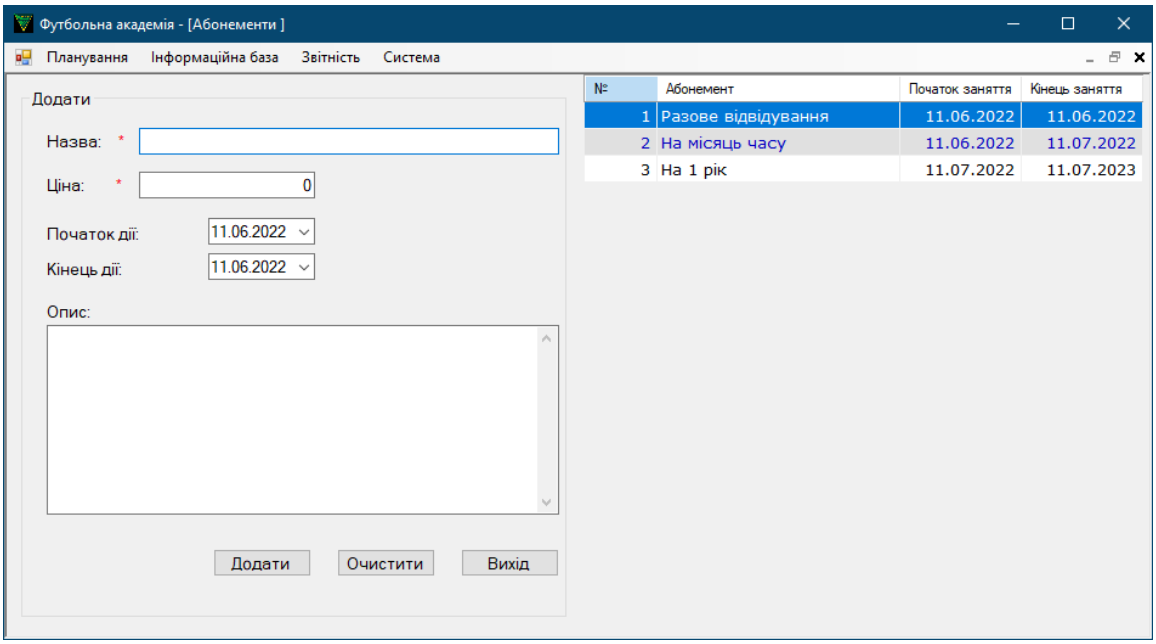

Рисунок 19 – Опрацювання інформації про абонементи

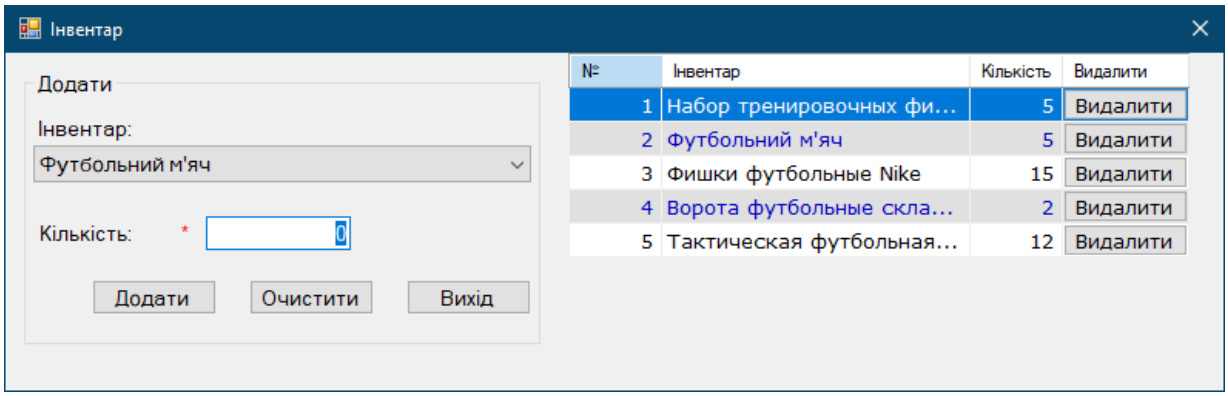

Рисунок 20 – Вікно для опрацювання необхідного інвентаря для заняття

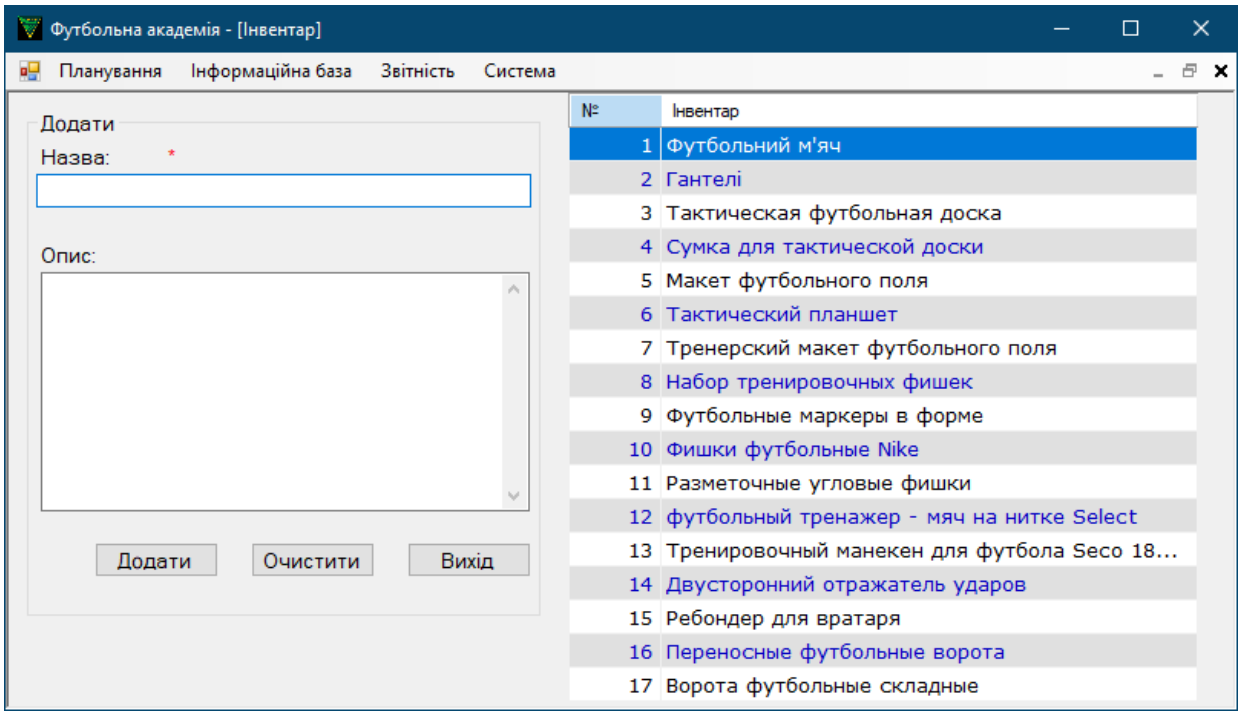

Рисунок 21 – Вікно для опрацювання даних інвентаря

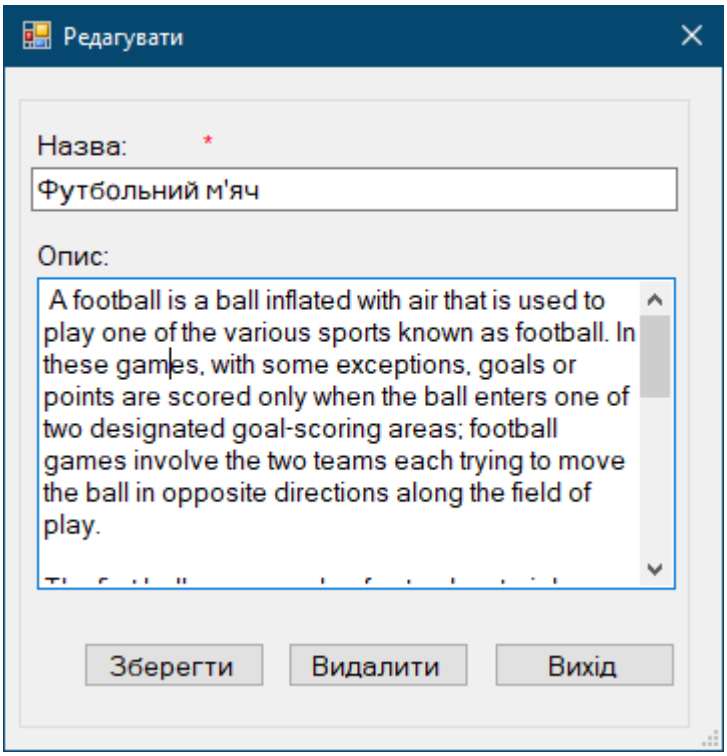

Рисунок 22 – Вікно для редагування інформації вибраного інвентаря

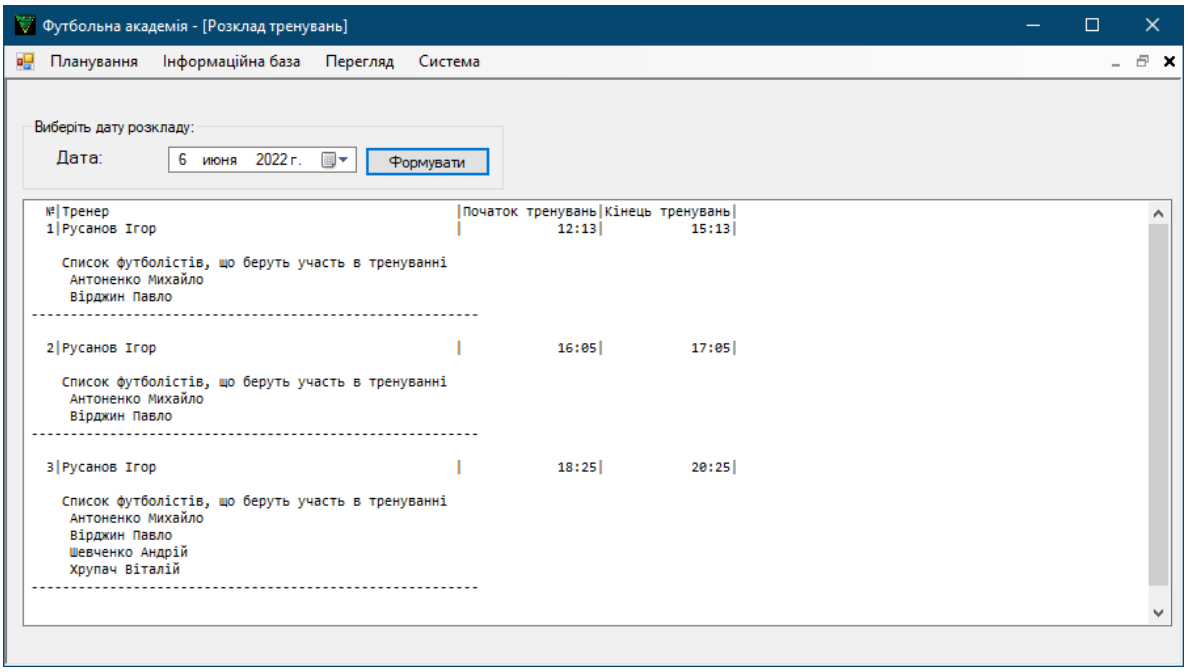

Рисунок 23 – Розклад тренувань

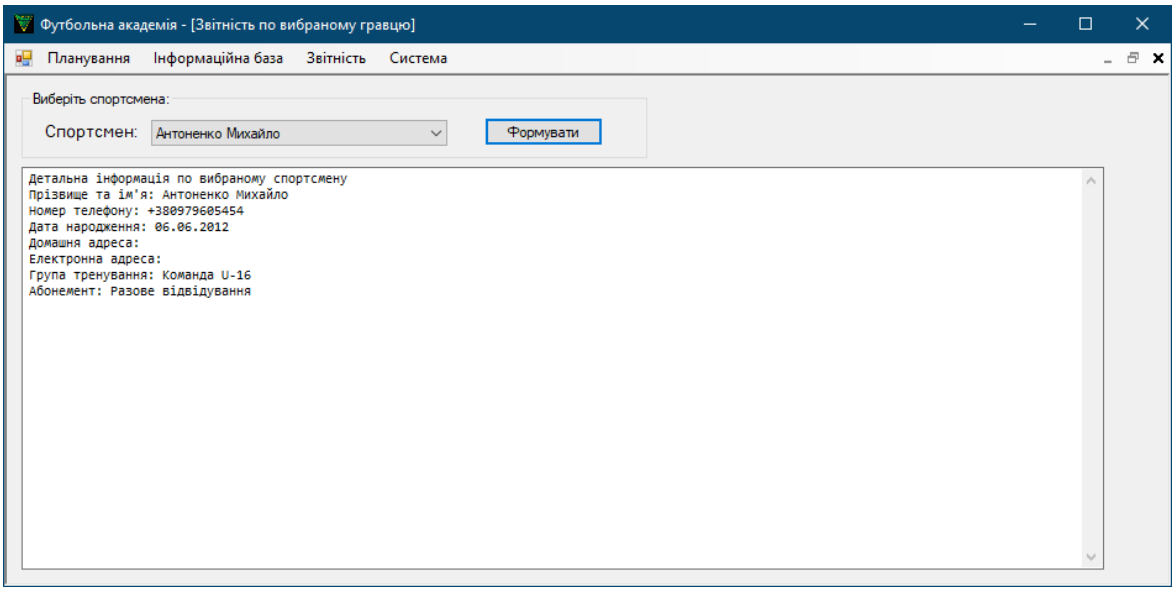

Рисунок 24 – Вікно для формування звітності про спортсмена

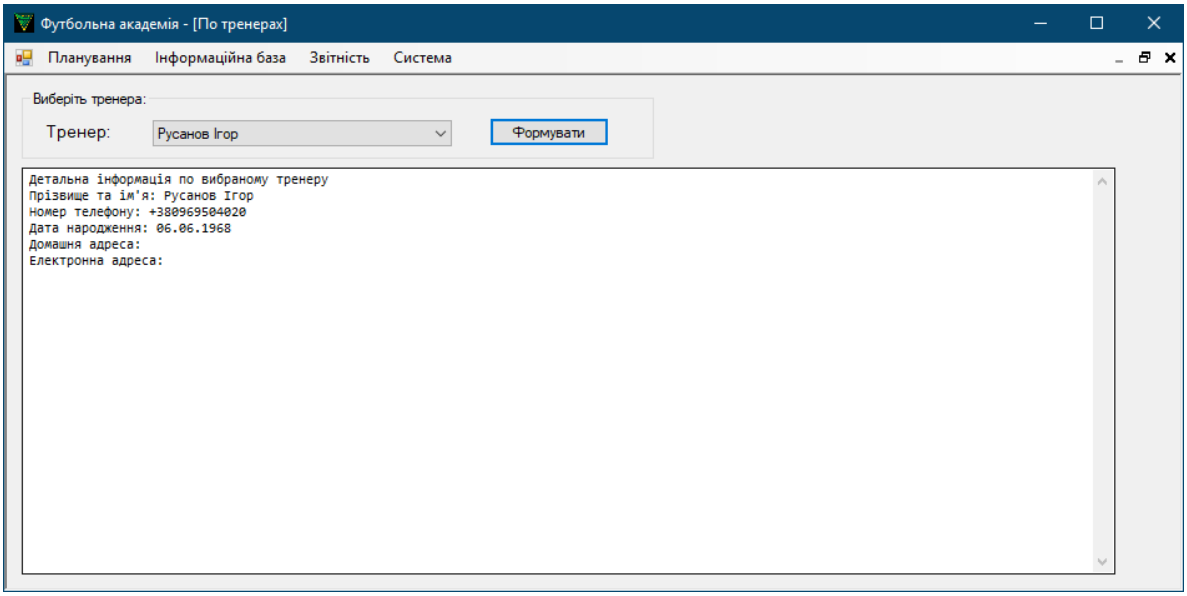

Рисунок 25 – Вікно для формування звітності про тренера

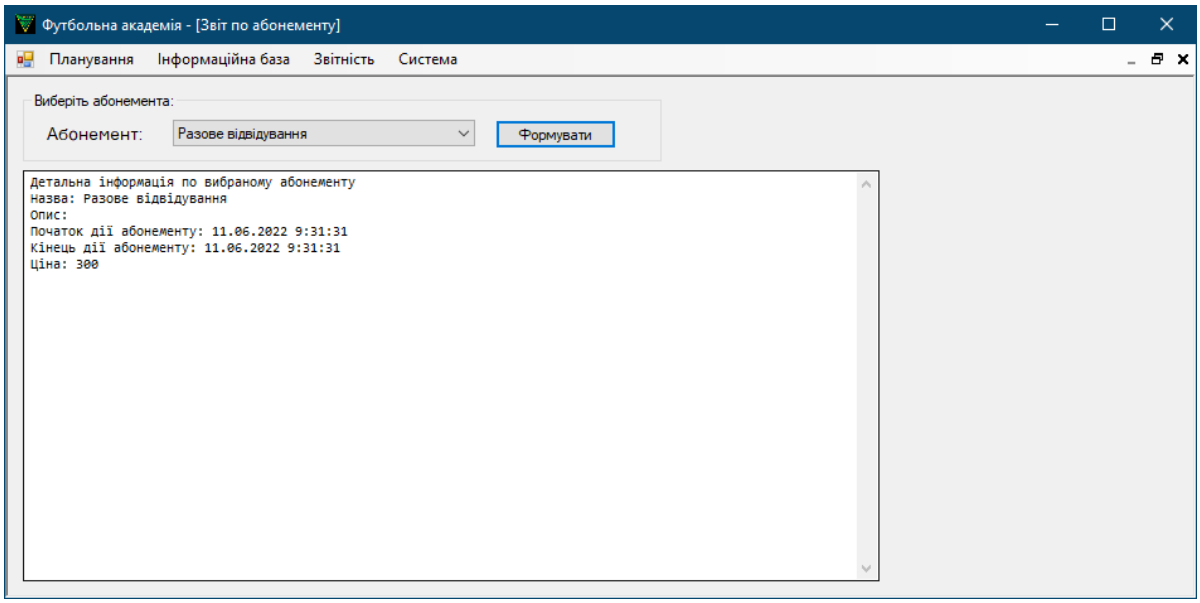

Рисунок 26 – Вікно для формування звітності про абонемент

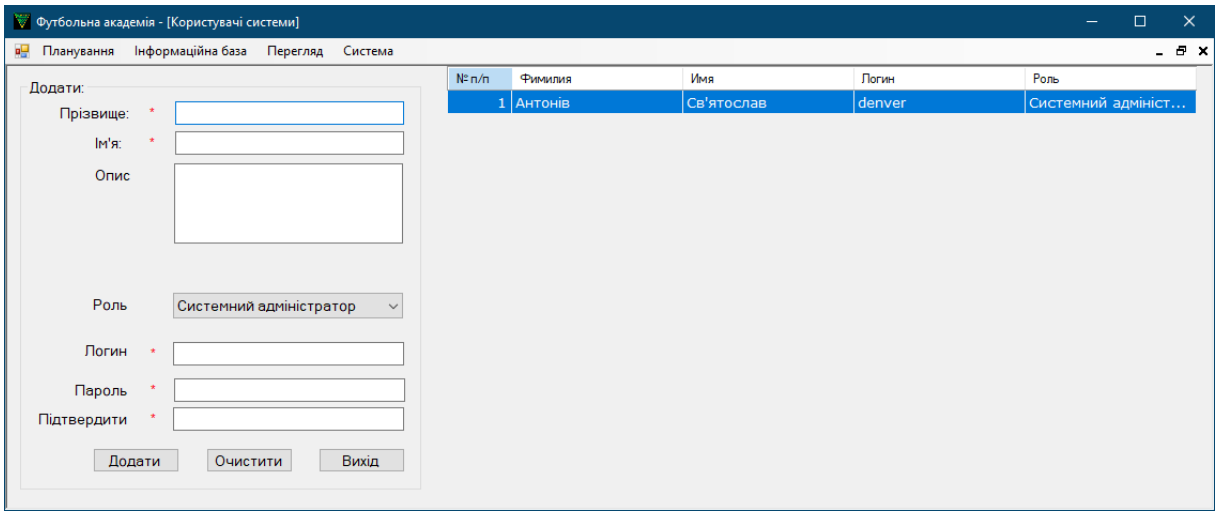

Рисунок 27 – Додавання нового користувача системи

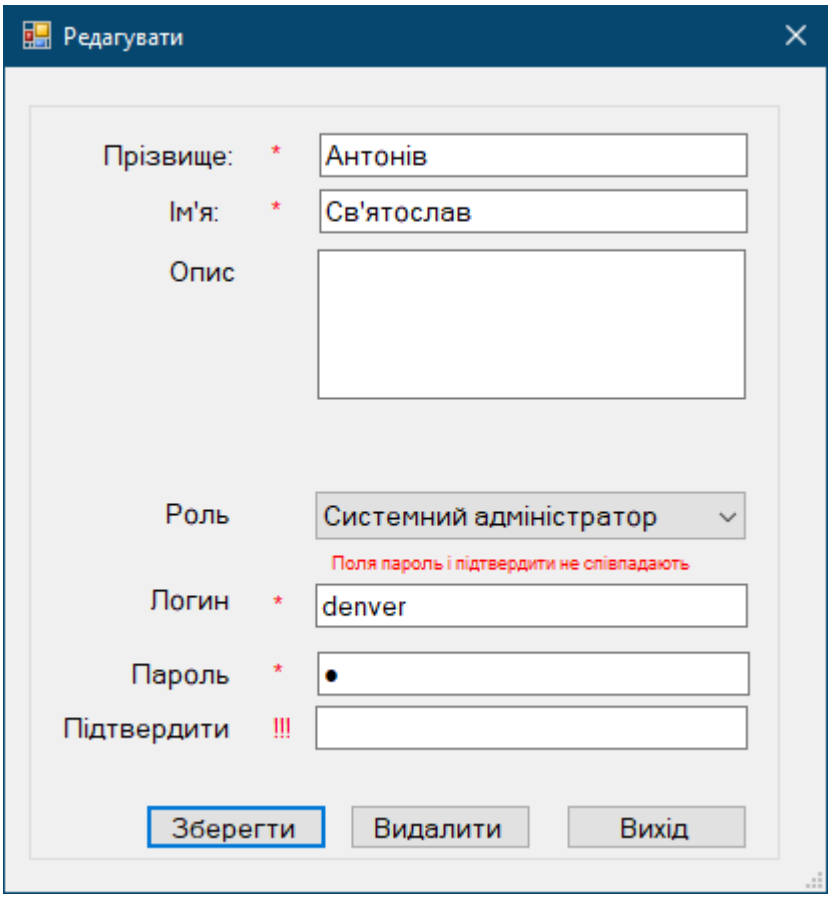

Рисунок 28 – Повідомлення про не співпадіння паролю

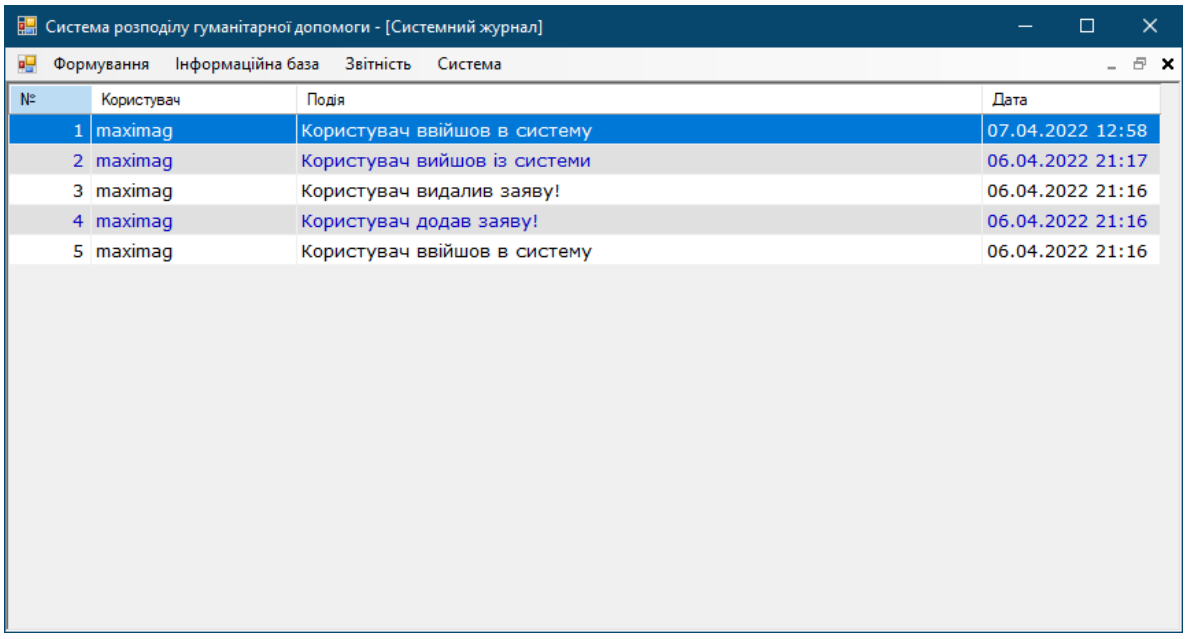

Рисунок 29 – Вікно «Системний журнал».

Додаток Е. Програмний код

Лістинг 1. Код класу «EncryptData»

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Security.Cryptography;
using System.IO;

namespace FootballAcademy.AppCode
{
  class EncryptData
  {
    static byte[] bytes = ASCIIEncoding.ASCII.GetBytes("FixedWWW");

    public string Encrypt(string originalString)
    {
      if (String.IsNullOrEmpty(originalString))
      {
        throw new ArgumentNullException
               ("The string which needs to be encrypted can not be null.");
      }
      DESCryptoServiceProvider cryptoProvider = new DESCryptoServiceProvider();
      MemoryStream memoryStream = new MemoryStream();
      CryptoStream cryptoStream = new CryptoStream(memoryStream,
          cryptoProvider.CreateEncryptor(bytes, bytes), CryptoStreamMode.Write);
      StreamWriter writer = new StreamWriter(cryptoStream);
      writer.Write(originalString);
      writer.Flush();
      cryptoStream.FlushFinalBlock();
      writer.Flush();
      return Convert.ToBase64String(memoryStream.GetBuffer(), 0, (int)memoryStream.Length);
    }

    public string Decrypt(string cryptedString)
    {
      if (String.IsNullOrEmpty(cryptedString))
      {
        throw new ArgumentNullException
           ("The string which needs to be decrypted can not be null.");
      }
      DESCryptoServiceProvider cryptoProvider = new DESCryptoServiceProvider();
      MemoryStream memoryStream = new MemoryStream
              (Convert.FromBase64String(cryptedString));
      CryptoStream cryptoStream = new CryptoStream(memoryStream,
          cryptoProvider.CreateDecryptor(bytes, bytes), CryptoStreamMode.Read);
      StreamReader reader = new StreamReader(cryptoStream);
      return reader.ReadToEnd();
    }
  }
}
```

Лістинг 2. Код класу «RaportBLL»

```
using FootballAcademy.Providers;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace FootballAcademy.BLL {
  class RaportBLL {
    private ScheduleProvider _ScheduleProvider = new ScheduleProvider();
    private GroupsProvider _GroupsProvider = new GroupsProvider();
    private TrenerProvider _TrenerProvider = new TrenerProvider();

    public List<ScheduleRaport> GetScheduleRaportListByDate(DateTime ScheduleDate) {
      List<ScheduleRaport> allScheduleRaport = new List<ScheduleRaport>();
      List<Schedule> ScheduleList = new List<Schedule>();
      List<Groups> GroupsList = new List<Groups>();
      List<Trener> TrenerList = new List<Trener>();
      ScheduleList = _ScheduleProvider.GetAllSchedule();
      GroupsList = _GroupsProvider.GetAllGroups();
      TrenerList = _TrenerProvider.GetAllTrener();



      for (int i = 0; i < ScheduleList.Count; i++) {
        if (ScheduleDate.ToShortDateString() ==
ScheduleList[i].DateOfOccupation.ToShortDateString()) {
          ScheduleRaport oneSchedule = new ScheduleRaport();
          oneSchedule.ScheduleId = ScheduleList[i].ScheduleId;
          oneSchedule.ScheduleName = ScheduleList[i].ScheduleName;
          oneSchedule.DateOfOccupation = ScheduleList[i].DateOfOccupation;
          oneSchedule.StartDate = ScheduleList[i].StartDate;
          oneSchedule.EndDate = ScheduleList[i].EndDate;
          oneSchedule.TrenerId = ScheduleList[i].TrenerId;
          allScheduleRaport.Add(oneSchedule);
        }
      }

      allScheduleRaport.Sort(new ScheduleRaportListComparer("StartDate"));


      for (int i = 0; i < allScheduleRaport.Count; i++) {
        allScheduleRaport[i].Number = i + 1;
        allScheduleRaport[i].FIO = GetFIO(allScheduleRaport[i].TrenerId, TrenerList);
      }

      return allScheduleRaport;
    }

    private string GetFIO(int TrenerId, List<Trener> TrenerList) {
      for (int i = 0; i < TrenerList.Count; i++) {
        if (TrenerId == TrenerList[i].TrenerId) {
          return TrenerList[i].FIO;
        }
      }
      return "";
    }

    public class ScheduleRaportListComparer : IComparer<ScheduleRaport> {
      private string _sortColumn;
      private bool _reverse;

      public int Compare(ScheduleRaport a, ScheduleRaport b) {
        int retVal = 0;
        switch (_sortColumn) {
          case "StartDate":
            retVal = a.StartDate.CompareTo(b.StartDate);
            break;
          case "FIO":
            retVal = a.FIO.CompareTo(b.FIO);
            break;
```

```
        }
        int _reverseInt = 1;
        if ((_reverse)) {
          _reverseInt = -1;
        }
        return (retVal * _reverseInt);
      }

      public ScheduleRaportListComparer(string sortColumn) {
        if (sortColumn.Length == 0) {
          sortColumn = "StartDate desc";
        }
        _reverse = sortColumn.ToLowerInvariant().EndsWith(" desc");

        if (_reverse) {
          _sortColumn = sortColumn.Substring(0, sortColumn.Length - 5);
        } else {
          _sortColumn = sortColumn;
        }
      }
    }


  }
}

public class ScheduleRaport {
  private int _Number;
  private int _ScheduleId;
  private string _ScheduleName;
  private DateTime _StartDate;
  private DateTime _EndDate;
  private int _TrenerId;
  private string _FIO;
  private DateTime _DateOfOccupation;

  public ScheduleRaport() {
    _Number = 0;
    _ScheduleId = 0;
    _ScheduleName = String.Empty;
    _DateOfOccupation = new DateTime();
    _StartDate = new DateTime();
    _EndDate = new DateTime();
    _TrenerId = 0;
    _FIO = String.Empty;
  }
  public int Number {
    set { _Number = value; }
    get { return _Number; }
  }
  public int ScheduleId {
    set { _ScheduleId = value; }
    get { return _ScheduleId; }
  }
  public string ScheduleName {
    set { _ScheduleName = value; }
    get { return _ScheduleName; }
  }
  public DateTime DateOfOccupation {
    set { _DateOfOccupation = value; }
    get { return _DateOfOccupation; }
  }
  public DateTime StartDate {
    set { _StartDate = value; }
```

```
    get { return _StartDate; }
  }
  public DateTime EndDate {
    set { _EndDate = value; }
    get { return _EndDate; }
  }
  public int TrenerId {
    set { _TrenerId = value; }
    get { return _TrenerId; }
  }
  public string FIO {
    set { _FIO = value; }
    get { return _FIO; }
  }

      }
```

Лістинг 3. Код класу «GroupForm»

```
using FootballAcademy.AppCode;
using FootballAcademy.Providers;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace FootballAcademy.Forms.InfoBase {
  public partial class GroupForm : Form {
    private int _selectedRowIndex = 0;
    private ValidationMy _validation = new ValidationMy();
    private GroupsProvider _GroupProvider = new GroupsProvider();
    private List<Groups> _GroupList = new List<Groups>();
    private TrenerProvider _TrenerProvider = new TrenerProvider();
    private List<Trener> _TrenerList = new List<Trener>();

    public GroupForm() {
      InitializeComponent();
      DataLoad();
      LoadAllDate();
    }

    private void AddBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
        _GroupProvider.InsertGroups(GroupNameTBox.Text, DescriptionTBox.Text,
Convert.ToInt32(TrenerCBox.SelectedValue));
        DataLoad();
        ClearAllControls();
      }
    }

    private void ClearBtn_Click(object sender, EventArgs e) {
      ClearAllControls();
    }

    private void ExitBtn_Click(object sender, EventArgs e) {
      this.Close();
    }

    private void LoadAllDate() {
```

```
        _TrenerList = _TrenerProvider.GetAllTrener();
        TrenerCBox.DataSource = _TrenerList;
        TrenerCBox.ValueMember = "TrenerId";
        TrenerCBox.DisplayMember = "FIO";
      }

      private void DataLoad() {
        int firstRowIndex = 0;
        if (GroupGridView.FirstDisplayedScrollingRowIndex > 0) {
          firstRowIndex = GroupGridView.FirstDisplayedScrollingRowIndex;
        }
        try {
          _GroupList = _GroupProvider.GetAllGroups();
          LoadDataInGroupGridView(_GroupList);
          if (_selectedRowIndex == GroupGridView.Rows.Count) {
            _selectedRowIndex = GroupGridView.Rows.Count - 1;
          }
          if (_selectedRowIndex >= 0) {
            GroupGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;
            GroupGridView.Rows[_selectedRowIndex].Selected = true;
          }
        } catch { }
      }

      private void LoadDataInGroupGridView(List<Groups> GroupList) {
        GroupGridView.DataSource = null;
        GroupGridView.Columns.Clear();
        GroupGridView.AutoGenerateColumns = false;
        GroupGridView.RowHeadersVisible = false;

        GroupGridView.DataSource = GroupList;

        if (GroupList.Count > 0) {
          if (GroupList[0].Message== NamesMy.NoDataNames.NoDataInGroup) {
            DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
            messageColumn.DataPropertyName = "Message";
            messageColumn.Width = GroupGridView.Width - NamesMy.SizeOptins.MinusSizePanel;
            GroupGridView.Columns.Add(messageColumn);
          } else {
            DataGridViewColumn DetailIdColumn = new DataGridViewTextBoxColumn();
            DetailIdColumn.DataPropertyName = "GroupsId";
            GroupGridView.Columns.Add(DetailIdColumn);
            GroupGridView.Columns[0].Visible = false;

            DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
            numberColumn.HeaderText = "№ ";
            numberColumn.DataPropertyName = "Number";
            numberColumn.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
            numberColumn.Width = NamesMy.SizeOptins.NumberSize;
            GroupGridView.Columns.Add(numberColumn);

            DataGridViewColumn GroupNameColumn = new DataGridViewTextBoxColumn();
            GroupNameColumn.HeaderText = "Групи";
            GroupNameColumn.DataPropertyName = "GroupsName";
            GroupNameColumn.Width = NamesMy.SizeOptins.NameSize;
            GroupGridView.Columns.Add(GroupNameColumn);


          }
          for (int i = 0; i < GroupGridView.Columns.Count; i++) {
            GroupGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
          }
        }
```

```
    }

    private void ClearAllControls() {
      GroupNameTBox.Text = String.Empty;
      DescriptionTBox.Text = String.Empty;
    }

    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
      if (_validation.IsDataEntering(GroupNameTBox.Text)) {
        GroupNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        GroupNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }

    private void GroupGridView_CellClick(object sender, DataGridViewCellEventArgs e) {
      if (e.RowIndex >= 0 && GroupGridView[0, e.RowIndex].Value.ToString() !=
_GroupList[0].Message) {
        _selectedRowIndex = e.RowIndex;
        UpdateGroupForm updateGroupForm = new
UpdateGroupForm(Convert.ToInt32(GroupGridView[0, e.RowIndex].Value.ToString()));
        updateGroupForm.ShowDialog();
        DataLoad();
      }
    }
  }
}
```

Лістинг 4. Код класу «SoccerPlayerForm»

```
using FootballAcademy.AppCode;
using FootballAcademy.Providers;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace FootballAcademy.Forms.InfoBase {
  public partial class SoccerPlayerForm : Form {
    private int _selectedRowIndex = 0;
    private ValidationMy _validation = new ValidationMy();
    private SoccerPlayerProvider _SoccerPlayerProvider = new SoccerPlayerProvider();
    private List<SoccerPlayer> _SoccerPlayerList = new List<SoccerPlayer>();
    private GroupsProvider _GroupProvider = new GroupsProvider();
    private List<Groups> _GroupList = new List<Groups>();
    public SoccerPlayerForm() {
      InitializeComponent();
      DataLoad();
      LoadAllDate();
    }

    private void AddBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
```

```
        _SoccerPlayerProvider.InsertSoccerPlayer(FirstNameTBox.Text, LastNameTBox.Text,
PhoneTBox.Text, BirthDateDTP.Value,
          AddressTBox.Text, EmailTBox.Text, Convert.ToInt32(GroupCBox.SelectedValue));
        DataLoad();
        ClearAllControls();
      }
    }

    private void ClearBtn_Click(object sender, EventArgs e) {
      ClearAllControls();
    }

    private void ExitBtn_Click(object sender, EventArgs e) {
      this.Close();
    }

    private void LoadAllDate() {
      _GroupList = _GroupProvider.GetAllGroups();
      GroupCBox.DataSource = _GroupList;
      GroupCBox.ValueMember = "GroupsId";
      GroupCBox.DisplayMember = "GroupsName";
    }

    private void DataLoad() {
      int firstRowIndex = 0;
      if (SoccerPlayerGridView.FirstDisplayedScrollingRowIndex > 0) {
        firstRowIndex = SoccerPlayerGridView.FirstDisplayedScrollingRowIndex;
      }
      try {
        _SoccerPlayerList = _SoccerPlayerProvider.GetAllSoccerPlayer();
        LoadDataInSoccerPlayerGridView(_SoccerPlayerList);
        if (_selectedRowIndex == SoccerPlayerGridView.Rows.Count) {
          _selectedRowIndex = SoccerPlayerGridView.Rows.Count - 1;
        }
        if (_selectedRowIndex >= 0) {
          SoccerPlayerGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;
          SoccerPlayerGridView.Rows[_selectedRowIndex].Selected = true;
        }
      } catch { }
    }

    private void LoadDataInSoccerPlayerGridView(List<SoccerPlayer> SoccerPlayerList) {
      SoccerPlayerGridView.DataSource = null;
      SoccerPlayerGridView.Columns.Clear();
      SoccerPlayerGridView.AutoGenerateColumns = false;
      SoccerPlayerGridView.RowHeadersVisible = false;

      SoccerPlayerGridView.DataSource = SoccerPlayerList;

      if (SoccerPlayerList.Count > 0) {
        if (SoccerPlayerList[0].Message == NamesMy.NoDataNames.NoDataInSoccerPlayer) {
          DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
          messageColumn.DataPropertyName = "Message";
          messageColumn.Width = SoccerPlayerGridView.Width -
NamesMy.SizeOptins.MinusSizePanel;
          SoccerPlayerGridView.Columns.Add(messageColumn);
        } else {
          DataGridViewColumn DetailIdColumn = new DataGridViewTextBoxColumn();
          DetailIdColumn.DataPropertyName = "SoccerPlayerId";
          SoccerPlayerGridView.Columns.Add(DetailIdColumn);
          SoccerPlayerGridView.Columns[0].Visible = false;

          DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
          numberColumn.HeaderText = "№ ";
```

```
          numberColumn.DataPropertyName = "Number";
          numberColumn.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
          numberColumn.Width = NamesMy.SizeOptins.NumberSize;
          SoccerPlayerGridView.Columns.Add(numberColumn);

          DataGridViewColumn LastNameColumn = new DataGridViewTextBoxColumn();
          LastNameColumn.HeaderText = "Прізвище";
          LastNameColumn.DataPropertyName = "LastName";
          LastNameColumn.Width = NamesMy.SizeOptins.NameSize;
          SoccerPlayerGridView.Columns.Add(LastNameColumn);

          DataGridViewColumn FirstNameColumn = new DataGridViewTextBoxColumn();
          FirstNameColumn.HeaderText = "Ім'я";
          FirstNameColumn.DataPropertyName = "FirstName";
          FirstNameColumn.Width = NamesMy.SizeOptins.NameSize;
          SoccerPlayerGridView.Columns.Add(FirstNameColumn);

          DataGridViewColumn PhoneColumn = new DataGridViewTextBoxColumn();
          PhoneColumn.HeaderText = "№ телефону";
          PhoneColumn.DataPropertyName = "Phone";
          PhoneColumn.Width = 130;
          SoccerPlayerGridView.Columns.Add(PhoneColumn);
        }
        for (int i = 0; i < SoccerPlayerGridView.Columns.Count; i++) {
          SoccerPlayerGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
        }
      }
    }

    private void ClearAllControls() {
      LastNameTBox.Text = String.Empty;
      FirstNameTBox.Text = String.Empty;
      PhoneTBox.Text = String.Empty;
    }

    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
      if (_validation.IsDataEntering(LastNameTBox.Text)) {
        LastNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        LastNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(FirstNameTBox.Text)) {
        FirstNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        FirstNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(PhoneTBox.Text)) {
        PhoneValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        PhoneValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }

    private void SoccerPlayerGridView_CellClick(object sender, DataGridViewCellEventArgs e)
{
      if (e.RowIndex >= 0 && SoccerPlayerGridView[0, e.RowIndex].Value.ToString() !=
_SoccerPlayerList[0].Message) {
        _selectedRowIndex = e.RowIndex;
```

```
        UpdateSoccerPlayerForm updateSoccerPlayerForm = new
UpdateSoccerPlayerForm(Convert.ToInt32(SoccerPlayerGridView[0,
e.RowIndex].Value.ToString()));
        updateSoccerPlayerForm.ShowDialog();
        DataLoad();
      }
    }
  }
}
```

Лістинг 5. Код класу «TrenerForm»

```
using FootballAcademy.AppCode;
using FootballAcademy.Providers;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace FootballAcademy.Forms.InfoBase {
  public partial class TrenerForm : Form {
    private int _selectedRowIndex = 0;
    private ValidationMy _validation = new ValidationMy();
    private TrenerProvider _TrenerProvider = new TrenerProvider();
    private List<Trener> _TrenerList = new List<Trener>();

    public TrenerForm() {
      InitializeComponent();
      DataLoad();

    }

    private void AddBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
        _TrenerProvider.InsertTrener(FirstNameTBox.Text, LastNameTBox.Text, PhoneTBox.Text,
BirthDateDTP.Value,
          AddressTBox.Text, EmailTBox.Text);
        DataLoad();
        ClearAllControls();
      }
    }

    private void ClearBtn_Click(object sender, EventArgs e) {
      ClearAllControls();
    }

    private void ExitBtn_Click(object sender, EventArgs e) {
      this.Close();
    }

    private void DataLoad() {
      int firstRowIndex = 0;
      if (TrenerGridView.FirstDisplayedScrollingRowIndex > 0) {
        firstRowIndex = TrenerGridView.FirstDisplayedScrollingRowIndex;
      }
      try {
        _TrenerList = _TrenerProvider.GetAllTrener();
```

```
            LoadDataInTrenerGridView(_TrenerList);
            if (_selectedRowIndex == TrenerGridView.Rows.Count) {
              _selectedRowIndex = TrenerGridView.Rows.Count - 1;
            }
            if (_selectedRowIndex >= 0) {
              TrenerGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;
              TrenerGridView.Rows[_selectedRowIndex].Selected = true;
            }
          } catch { }
        }

        private void LoadDataInTrenerGridView(List<Trener> TrenerList) {
          TrenerGridView.DataSource = null;
          TrenerGridView.Columns.Clear();
          TrenerGridView.AutoGenerateColumns = false;
          TrenerGridView.RowHeadersVisible = false;

          TrenerGridView.DataSource = TrenerList;

          if (TrenerList.Count > 0) {
            if (TrenerList[0].Message == NamesMy.NoDataNames.NoDataInTrener) {
              DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
              messageColumn.DataPropertyName = "Message";
              messageColumn.Width = TrenerGridView.Width - NamesMy.SizeOptins.MinusSizePanel;
              TrenerGridView.Columns.Add(messageColumn);
            } else {
              DataGridViewColumn DetailIdColumn = new DataGridViewTextBoxColumn();
              DetailIdColumn.DataPropertyName = "TrenerId";
              TrenerGridView.Columns.Add(DetailIdColumn);
              TrenerGridView.Columns[0].Visible = false;

              DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
              numberColumn.HeaderText = "№ ";
              numberColumn.DataPropertyName = "Number";
              numberColumn.DefaultCellStyle.Alignment =
    DataGridViewContentAlignment.MiddleRight;
              numberColumn.Width = NamesMy.SizeOptins.NumberSize;
              TrenerGridView.Columns.Add(numberColumn);

              DataGridViewColumn LastNameColumn = new DataGridViewTextBoxColumn();
              LastNameColumn.HeaderText = "Прізвище";
              LastNameColumn.DataPropertyName = "LastName";
              LastNameColumn.Width = NamesMy.SizeOptins.NameSize;
              TrenerGridView.Columns.Add(LastNameColumn);

              DataGridViewColumn FirstNameColumn = new DataGridViewTextBoxColumn();
              FirstNameColumn.HeaderText = "Ім'я";
              FirstNameColumn.DataPropertyName = "FirstName";
              FirstNameColumn.Width = NamesMy.SizeOptins.NameSize;
              TrenerGridView.Columns.Add(FirstNameColumn);

              DataGridViewColumn PhoneColumn = new DataGridViewTextBoxColumn();
              PhoneColumn.HeaderText = "№ телефону";
              PhoneColumn.DataPropertyName = "Phone";
              PhoneColumn.Width = 130;
              TrenerGridView.Columns.Add(PhoneColumn);
            }
            for (int i = 0; i < TrenerGridView.Columns.Count; i++) {
              TrenerGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
            }
          }
        }

        private void ClearAllControls() {
```

```
      LastNameTBox.Text = String.Empty;
      FirstNameTBox.Text = String.Empty;
      PhoneTBox.Text = String.Empty;
    }

    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
      if (_validation.IsDataEntering(LastNameTBox.Text)) {
        LastNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        LastNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(FirstNameTBox.Text)) {
        FirstNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        FirstNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(PhoneTBox.Text)) {
        PhoneValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        PhoneValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }

    private void TrenerGridView_CellClick(object sender, DataGridViewCellEventArgs e) {
      if (e.RowIndex >= 0 && TrenerGridView[0, e.RowIndex].Value.ToString() !=
_TrenerList[0].Message) {
        _selectedRowIndex = e.RowIndex;
        UpdateTrenerForm updateTrenerForm = new
UpdateTrenerForm(Convert.ToInt32(TrenerGridView[0, e.RowIndex].Value.ToString()));
        updateTrenerForm.ShowDialog();
        DataLoad();
      }
    }

  }
}
```

Лістинг 6. Код класу «UpdateGroupForm»

```
using FootballAcademy.AppCode;
using FootballAcademy.Providers;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace FootballAcademy.Forms.InfoBase {
  public partial class UpdateGroupForm : Form {
    private int _GroupId = 0;
    private Groups _selectedGroup = new Groups();
    private GroupsProvider _GroupPrivider = new GroupsProvider();
    private ValidationMy _Validation = new ValidationMy();
```

```
    private TrenerProvider _TrenerProvider = new TrenerProvider();
    private List<Trener> _TrenerList = new List<Trener>();
    public UpdateGroupForm(int GroupId) {
      InitializeComponent();
      _GroupId = GroupId;
      LoadAllDate();
    }

    private void SaveBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
        _GroupPrivider.UpdateGroups(GroupNameTBox.Text, DescriptionTBox.Text,
Convert.ToInt32(TrenerCBox.SelectedValue), _GroupId);
        this.Close();
      }
    }

    private void DeleteBtn_Click(object sender, EventArgs e) {
      if (MessageBox.Show("Ви дійсно хочете видалити цей елемент?", "Видалити",
MessageBoxButtons.YesNo) == DialogResult.Yes) {
        _GroupPrivider.DeleteGroups(_GroupId);
        this.Close();
      }
    }

    private void ExitBtn_Click(object sender, EventArgs e) {
      this.Close();
    }

    private void LoadAllDate() {
      _TrenerList = _TrenerProvider.GetAllTrener();
      TrenerCBox.DataSource = _TrenerList;
      TrenerCBox.ValueMember = "TrenerId";
      TrenerCBox.DisplayMember = "FIO";

      _selectedGroup = _GroupPrivider.SelectedGroupsByGroupsId(_GroupId);
      GroupNameTBox.Text = _selectedGroup.GroupsName;
      DescriptionTBox.Text = _selectedGroup.Description;
      TrenerCBox.SelectedValue = _selectedGroup.TrenerId;
    }

    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
      if (_Validation.IsDataEntering(GroupNameTBox.Text)) {
        GroupNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        GroupNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }
  }
}
```

Лістинг 7. Код класу «UpdateSoccerPlayerForm»

```
using FootballAcademy.AppCode;
using FootballAcademy.Providers;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace FootballAcademy.Forms.InfoBase {
  public partial class UpdateSoccerPlayerForm : Form {
    private int _SoccerPlayerId = 0;
    private SoccerPlayer _selectedSoccerPlayer = new SoccerPlayer();
    private SoccerPlayerProvider _SoccerPlayerPrivider = new SoccerPlayerProvider();
    private ValidationMy _validation = new ValidationMy();
    private GroupsProvider _GroupProvider = new GroupsProvider();
    private List<Groups> _GroupList = new List<Groups>();
    public UpdateSoccerPlayerForm(int SoccerPlayerId) {
      InitializeComponent();
      _SoccerPlayerId = SoccerPlayerId;
      LoadAllDate();
    }

    private void SaveBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
        _SoccerPlayerPrivider.UpdateSoccerPlayer(FirstNameTBox.Text, LastNameTBox.Text,
PhoneTBox.Text, BirthDateDTP.Value,
          AddressTBox.Text, EmailTBox.Text, Convert.ToInt32(GroupCBox.SelectedValue),
_SoccerPlayerId);
        this.Close();
      }
    }

    private void DeleteBtn_Click(object sender, EventArgs e) {
      if (MessageBox.Show("Ви дійсно хочете видалити цей елемент?", "Видалити",
MessageBoxButtons.YesNo) == DialogResult.Yes) {
        _SoccerPlayerPrivider.DeleteSoccerPlayer(_SoccerPlayerId);
        this.Close();
      }
    }

    private void ExitBtn_Click(object sender, EventArgs e) {
      this.Close();
    }

    private void LoadAllDate() {
      _GroupList = _GroupProvider.GetAllGroups();
      GroupCBox.DataSource = _GroupList;
      GroupCBox.ValueMember = "GroupsId";
      GroupCBox.DisplayMember = "GroupsName";

      _selectedSoccerPlayer =
_SoccerPlayerPrivider.SelectedSoccerPlayerBySoccerPlayerId(_SoccerPlayerId);
      FirstNameTBox.Text = _selectedSoccerPlayer.FirstName;
      LastNameTBox.Text = _selectedSoccerPlayer.LastName;
      BirthDateDTP.Value = _selectedSoccerPlayer.BirthDate;
      GroupCBox.SelectedValue = _selectedSoccerPlayer.GroupId;
      PhoneTBox.Text = _selectedSoccerPlayer.Phone;
      EmailTBox.Text = _selectedSoccerPlayer.Email;
      AddressTBox.Text = _selectedSoccerPlayer.Address;
    }

    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
      if (_validation.IsDataEntering(LastNameTBox.Text)) {
        LastNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        LastNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
```

```
        isCorrect = false;
      }
      if (_validation.IsDataEntering(FirstNameTBox.Text)) {
        FirstNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        FirstNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(PhoneTBox.Text)) {
        PhoneValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        PhoneValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }

  }
}
```

Лістинг 8. Код класу «TrainingScheduleForm»

```
using FootballAcademy.AppCode;
using FootballAcademy.Forms.Systems;
using FootballAcademy.Providers;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace FootballAcademy.Forms.Planing {
  public partial class TrainingScheduleForm : Form {
    private int _selectedRowIndex = 0;
    private ValidationMy _validation = new ValidationMy();
    private ScheduleProvider _ScheduleProvider = new ScheduleProvider();
    private LogsProvider _LogsProvider = new LogsProvider();
    private List<Schedule> _ScheduleList = new List<Schedule>();
    private ScheduleLProvider _ScheduleLProvider = new ScheduleLProvider();
    private List<ScheduleL> _ScheduleLList = new List<ScheduleL>();
    private TrenerProvider _TrenerProvider = new TrenerProvider();
    private List<Trener> _TrenerList = new List<Trener>();
```

```
        private SoccerPlayerProvider _SoccerPlayerProvider = new SoccerPlayerProvider();
        private List<SoccerPlayer> _SoccerPlayerList = new List<SoccerPlayer>();
        private GroupsProvider _GroupsProvider = new GroupsProvider();
        private Groups _SelectedGroup = new Groups();

        public TrainingScheduleForm() {
          InitializeComponent();
          LoadAllDate();
          DataLoad();
        }

        private void FormingBtn_Click(object sender, EventArgs e) {
          if (IsDataEnteringCorrect()) {
            FormingStatement();
          }
        }
        private void AddBtn_Click(object sender, EventArgs e) {
          if (IsDataEnteringCorrect() && _ScheduleLList.Count > 0) {
            _ScheduleProvider.InsertSchedule(ScheduleNameTBox.Text, DateOfCompletionDTP.Value,
              StartDateDTP.Value, EndDateDTP.Value, Convert.ToInt32(TrenerCBox.SelectedValue));
            int lastSchedule = _ScheduleProvider.GetLastRecords();
            for (int i = 0; i < _ScheduleLList.Count; i++) {
              _ScheduleLList[i].ScheduleId = lastSchedule;
            }
            _ScheduleLProvider.InsertBatchScheduleL(_ScheduleLList);
            _LogsProvider.InsertLogs(LoginForm.CurrentUser.UsersId, "Було створене нове тренування: " +
ScheduleNameTBox.Text, DateTime.Now);
            ClearAllControls();
            DataLoad();
          }
        }

        private void ClearBtn_Click(object sender, EventArgs e) {
          ClearAllControls();
        }

        private void ExitBtn_Click(object sender, EventArgs e) {
          this.Close();
        }

        private void LoadAllDate() {
```

```
  _TrenerList = _TrenerProvider.GetAllTrener();
  TrenerCBox.DataSource = _TrenerList;
  TrenerCBox.ValueMember = "TrenerId";
  TrenerCBox.DisplayMember = "FIO";
 }

 private void DataLoad() {
  int firstRowIndex = 0;
  if (ScheduleGridView.FirstDisplayedScrollingRowIndex > 0) {
   firstRowIndex = ScheduleGridView.FirstDisplayedScrollingRowIndex;
  }
  try {
   _ScheduleList = _ScheduleProvider.GetAllSchedule();
   LoadDataInScheduleGridView(_ScheduleList);
   if (_selectedRowIndex == ScheduleGridView.Rows.Count) {
    _selectedRowIndex = ScheduleGridView.Rows.Count - 1;
   }
   if (_selectedRowIndex >= 0) {
    ScheduleGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;
    ScheduleGridView.Rows[_selectedRowIndex].Selected = true;
   }
  } catch { }
 }

 private void LoadDataInScheduleGridView(List<Schedule> ScheduleList) {
  ScheduleGridView.DataSource = null;
  ScheduleGridView.Columns.Clear();
  ScheduleGridView.AutoGenerateColumns = false;
  ScheduleGridView.RowHeadersVisible = false;

  ScheduleGridView.DataSource = ScheduleList;

  if (ScheduleList.Count > 0) {
   if (ScheduleList[0].Message == NamesMy.NoDataNames.NoDataInSchedule) {
    DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
    messageColumn.DataPropertyName = "Message";
    messageColumn.Width = ScheduleGridView.Width - NamesMy.SizeOptins.MinusSizePanel;
    ScheduleGridView.Columns.Add(messageColumn);
   } else {
    DataGridViewColumn DetailIdColumn = new DataGridViewTextBoxColumn();
    DetailIdColumn.DataPropertyName = "ScheduleId";
```

```
            ScheduleGridView.Columns.Add(DetailIdColumn);
            ScheduleGridView.Columns[0].Visible = false;

            DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
            numberColumn.HeaderText = "№ ";
            numberColumn.DataPropertyName = "Number";
            numberColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
            numberColumn.Width = NamesMy.SizeOptins.NumberSize;
            ScheduleGridView.Columns.Add(numberColumn);

            DataGridViewColumn ScheduleNameColumn = new DataGridViewTextBoxColumn();
            ScheduleNameColumn.HeaderText = "Найменування заняття";
            ScheduleNameColumn.DataPropertyName = "ScheduleName";
            ScheduleNameColumn.Width = NamesMy.SizeOptins.NameSize;
            ScheduleGridView.Columns.Add(ScheduleNameColumn);

            DataGridViewColumn DateOfCompletionColumn = new DataGridViewTextBoxColumn();
            DateOfCompletionColumn.HeaderText = "Дата заняття";
            DateOfCompletionColumn.DataPropertyName = "DateOfOccupation";
            DateOfCompletionColumn.DefaultCellStyle.Format = "dd/MM/yyyy";
            DateOfCompletionColumn.DefaultCellStyle.Alignment                                    =
DataGridViewContentAlignment.MiddleRight;
            DateOfCompletionColumn.Width = 100;
            ScheduleGridView.Columns.Add(DateOfCompletionColumn);

            DataGridViewColumn StartDateColumn = new DataGridViewTextBoxColumn();
            StartDateColumn.HeaderText = "Початок заняття";
            StartDateColumn.DataPropertyName = "StartDate";
            StartDateColumn.DefaultCellStyle.Format = "HH:mm:ss";
            StartDateColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
            StartDateColumn.Width = 100;
            ScheduleGridView.Columns.Add(StartDateColumn);

            DataGridViewColumn EndDateColumn = new DataGridViewTextBoxColumn();
            EndDateColumn.HeaderText = "Кінець заняття";
            EndDateColumn.DataPropertyName = "EndDate";
            EndDateColumn.DefaultCellStyle.Format = "HH:mm:ss";
            EndDateColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
            EndDateColumn.Width = 100;
            ScheduleGridView.Columns.Add(EndDateColumn);
```

```
            DataGridViewButtonColumn deleteBtn = new DataGridViewButtonColumn();
            deleteBtn.HeaderText = NamesMy.ProgramButtons.DeleteBtn;
            deleteBtn.Text = NamesMy.ProgramButtons.DeleteBtn;
            deleteBtn.UseColumnTextForButtonValue = true;
            deleteBtn.ToolTipText = NamesMy.ProgramButtons.DeleteBtn;
            deleteBtn.Width = NamesMy.SizeOptins.DeleteBtnSize;
            ScheduleGridView.Columns.Add(deleteBtn);
          }
          for (int i = 0; i < ScheduleGridView.Columns.Count; i++) {
            ScheduleGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
          }
        }
      }

      private void ClearAllControls() {
        ScheduleNameTBox.Text = String.Empty;
        _ScheduleLList.Clear();
        LoadDataInFormingScheduleDGV(_ScheduleLList);
      }

      private bool IsDataEnteringCorrect() {
        bool isCorrect = true;
        if (_validation.IsDataEntering(ScheduleNameTBox.Text)) {
          ScheduleNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
        } else {
          ScheduleNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
          isCorrect = false;
        }
        return isCorrect;
      }

      private void FormingStatement() {
        _ScheduleLList.Clear();
        _SelectedGroup                                                                    =
_GroupsProvider.SelectedGroupsByTrenerId(Convert.ToInt32(TrenerCBox.SelectedValue));
        _SoccerPlayerList = _SoccerPlayerProvider.GetAllSoccerPlayerByGroupId(_SelectedGroup.GroupsId);
        for (int i = 0; i < _SoccerPlayerList.Count; i++) {
          ScheduleL oneScheduleL = new ScheduleL();
          oneScheduleL.SoccerPlayerId = _SoccerPlayerList[i].SoccerPlayerId;
          oneScheduleL.FIO = _SoccerPlayerList[i].FIO;
          oneScheduleL.TrenerId = Convert.ToInt32(TrenerCBox.SelectedValue);
```

```
            oneScheduleL.GroupId = _SelectedGroup.GroupsId;
            _ScheduleLList.Add(oneScheduleL);
          }
          LoadDataInFormingScheduleDGV(_ScheduleLList);
        }

        private void LoadDataInFormingScheduleDGV(List<ScheduleL> ScheduleLList) {
          FormingScheduleDGV.DataSource = null;
          FormingScheduleDGV.Columns.Clear();
          FormingScheduleDGV.AutoGenerateColumns = false;
          FormingScheduleDGV.RowHeadersVisible = false;

          FormingScheduleDGV.DataSource = ScheduleLList;

          if (ScheduleLList.Count > 0) {
            if (ScheduleLList[0].Message == NamesMy.NoDataNames.NoDataInScheduleL) {
              DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
              messageColumn.DataPropertyName = "Message";
              messageColumn.Width = ScheduleGridView.Width - NamesMy.SizeOptins.MinusSizePanel;
              FormingScheduleDGV.Columns.Add(messageColumn);
            } else {

              DataGridViewColumn ScheduleNameColumn = new DataGridViewTextBoxColumn();
              ScheduleNameColumn.HeaderText = "П.І.Б.";
              ScheduleNameColumn.DataPropertyName = "FIO";
              ScheduleNameColumn.Width = 320;
              FormingScheduleDGV.Columns.Add(ScheduleNameColumn);
            }
            for (int i = 0; i < FormingScheduleDGV.Columns.Count; i++) {
              FormingScheduleDGV.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
            }
          }
        }

        private void ScheduleGridView_CellClick(object sender, DataGridViewCellEventArgs e) {
          if (e.ColumnIndex == 6) {
            if (MessageBox.Show("Ви дійсно бажаєте видалити це тренування?", "Видалити",
MessageBoxButtons.YesNo) == DialogResult.Yes) {
              int selectedScheduleId = Convert.ToInt32(ScheduleGridView[0, e.RowIndex].Value.ToString());
              _ScheduleProvider.DeleteSchedule(selectedScheduleId);
              _ScheduleLProvider.DeleteScheduleL(selectedScheduleId);
```

```
            DataLoad();
          }
        }
      }
    }
  }
```

Лістинг 9. Код класу «ScheduleRaportForm»

```
using FootballAcademy.BLL;
using FootballAcademy.Providers;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace FootballAcademy.Forms.Raport {
  public partial class ScheduleRaportForm : Form {
    ScheduleLProvider _ScheduleLProvider = new ScheduleLProvider();
    List<ScheduleL> _ScheduleLList = new List<ScheduleL>();
    RaportBLL _allRaportBLL = new RaportBLL();
    List<ScheduleRaport> _ScheduleRaportList = new List<ScheduleRaport>();


    public ScheduleRaportForm() {
      InitializeComponent();
    }

    private void FormingBtn_Click(object sender, EventArgs e) {
      _ScheduleRaportList =
_allRaportBLL.GetScheduleRaportListByDate(DateOfOccupationDTP.Value);
      GetRaport(_ScheduleRaportList);

    }

    public void GetRaport(List<ScheduleRaport> ScheduleRaportList) {

      RaportTBox.Text = String.Format("{0,3}|{1, -50}|{2, 17}|{3, 16}|\r\n", "№", "Тренер",
"Початок тренувань", "Кінець тренувань");
      for (int i = 0; i < ScheduleRaportList.Count(); i++) {
        string raportString = String.Format("{0,3}|{1, -50}|{2, 17}|{3, 16}|\r\n",
        ScheduleRaportList[i].Number, ScheduleRaportList[i].FIO,
ScheduleRaportList[i].StartDate.ToShortTimeString(),
ScheduleRaportList[i].EndDate.ToShortTimeString());
        RaportTBox.Text += raportString;
        _ScheduleLList =
_ScheduleLProvider.GetAllScheduleLByScheduleId(ScheduleRaportList[i].ScheduleId);
        RaportTBox.Text += "\r\n";
        RaportTBox.Text += "    Список футболістів, що беруть участь в тренуванні\r\n";
        for (int j = 0; j < _ScheduleLList.Count; j++) {
          RaportTBox.Text += "     " +  _ScheduleLList[j].FIO + "\r\n";
        }
        RaportTBox.Text += "------------------------------------------------------------\r\n";
        RaportTBox.Text += "\r\n";
      }
    }
```

```
  }
}
```

Лістинг 10. Код класу «LoginForm»

```
using FootballAcademy.AppCode;
using FootballAcademy.Providers;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace FootballAcademy.Forms.Systems {
  public partial class LoginForm : Form {
    public static Users CurrentUser = new Users();

    private UsersProvider _UserProvider = new UsersProvider();
    private ValidationMy _validation = new ValidationMy();
    private LogsProvider _LogsProvider = new LogsProvider();
    private List<Users> _UserList = new List<Users>();
    public LoginForm() {
      InitializeComponent();
      LoadAllDate();
    }

    private void SubmitBtn_Click(object sender, EventArgs e) {
      GetSubmitData();
    }

    private void DataLoad() {
      _LogsProvider.InsertLogs(CurrentUser.UsersId, "Користувач ввійшов в систему",
DateTime.Now);
      this.Visible = false;
      (new FootballAcademyMDI()).ShowDialog();
      _LogsProvider.InsertLogs(CurrentUser.UsersId, "Користувач вийшов із системи",
DateTime.Now);
      this.Close();
    }

    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
      if (_validation.IsDataEntering(UserNameCBox.Text)) {
        UserNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        UserNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(UserPasswordTbx.Text)) {
        UserPasswordValidation.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        UserPasswordValidation.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }

    private void LoadAllDate() {
```

```
      _UserList = _UserProvider.GetAllUsers();
      UserNameCBox.DataSource = _UserList;
      UserNameCBox.AutoCompleteMode = AutoCompleteMode.SuggestAppend;
      UserNameCBox.AutoCompleteSource = AutoCompleteSource.ListItems;
      UserNameCBox.ValueMember = "UsersId";
      UserNameCBox.DisplayMember = "UsersName";
    }

    private void GetSubmitData() {
      try {
        if (IsDataEnteringCorrect()) {
          List<Users> listUsers = new List<Users>();
          listUsers =
_UserProvider.SelectedUsersByUsersNameAndUsersPassword(UserNameCBox.Text,
UserPasswordTbx.Text);
          if (listUsers.Count > 0) {
            CurrentUser = listUsers[0];
            DataLoad();
          } else {

MessageBox.Show(NamesMy.MessageBoxExaption.ThisUserLoginAndUserPasswordNotExistInSystem,
NamesMy.MessageBoxExaption.CaptionMessage);
          }
        }
      } catch (Exception ex) {
        MessageBox.Show(ex.ToString());
      }
    }
  }
}
```

Лістинг 11. Код класу «SystemLogForm»

```
using FootballAcademy.AppCode;
using FootballAcademy.Providers;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace FootballAcademy.Forms.Systems {
  public partial class SystemLogForm : Form {
    private int _selectedRowIndex = 0;
    private LogsProvider _LogsProvider = new LogsProvider();
    private List<Logs> _LogsList = new List<Logs>();
    public SystemLogForm() {
      InitializeComponent();
      DataLoad();
    }

    private void DataLoad() {
      int firstRowIndex = 0;
      if (LogsGridView.FirstDisplayedScrollingRowIndex > 0) {
        firstRowIndex = LogsGridView.FirstDisplayedScrollingRowIndex;
      }
      try {
        _LogsList = _LogsProvider.GetAllLogs();
```

```
        LoadDataInLogsGridView(_LogsList);
        if (_selectedRowIndex == LogsGridView.Rows.Count) {
          _selectedRowIndex = LogsGridView.Rows.Count - 1;
        }
        if (_selectedRowIndex >= 0) {
          LogsGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;
          LogsGridView.Rows[_selectedRowIndex].Selected = true;
        }
      } catch { }
    }

    private void LoadDataInLogsGridView(List<Logs> LogsList) {
      LogsGridView.DataSource = null;
      LogsGridView.Columns.Clear();
      LogsGridView.AutoGenerateColumns = false;
      LogsGridView.RowHeadersVisible = false;

      LogsGridView.DataSource = LogsList;

      if (LogsList.Count > 0) {
        if (LogsList[0].Message == NamesMy.NoDataNames.NoDataInLogs) {
          DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
          messageColumn.DataPropertyName = "Message";
          messageColumn.Width = LogsGridView.Width - NamesMy.SizeOptins.MinusSizePanel;
          LogsGridView.Columns.Add(messageColumn);
        } else {
          DataGridViewColumn DetailIdColumn = new DataGridViewTextBoxColumn();
          DetailIdColumn.DataPropertyName = "LogsId";
          LogsGridView.Columns.Add(DetailIdColumn);
          LogsGridView.Columns[0].Visible = false;

          DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
          numberColumn.HeaderText = "№ ";
          numberColumn.DataPropertyName = "Number";
          numberColumn.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
          numberColumn.Width = NamesMy.SizeOptins.NumberSize;
          LogsGridView.Columns.Add(numberColumn);

          DataGridViewColumn UsersNameColumn = new DataGridViewTextBoxColumn();
          UsersNameColumn.HeaderText = "Користувач";
          UsersNameColumn.DataPropertyName = "UsersName";
          UsersNameColumn.Width = 150;
          LogsGridView.Columns.Add(UsersNameColumn);

          DataGridViewColumn EventNameShowColumn = new DataGridViewTextBoxColumn();
          EventNameShowColumn.HeaderText = "Подія";
          EventNameShowColumn.DataPropertyName = "EventNameShow";
          EventNameShowColumn.Width = 500;
          LogsGridView.Columns.Add(EventNameShowColumn);

          DataGridViewColumn EvendDateColumn = new DataGridViewTextBoxColumn();
          EvendDateColumn.HeaderText = "Дата";
          EvendDateColumn.DataPropertyName = "EventDate";
          EvendDateColumn.Width = NamesMy.SizeOptins.Date;
          LogsGridView.Columns.Add(EvendDateColumn);

        }
        for (int i = 0; i < LogsGridView.Columns.Count; i++) {
          LogsGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
        }
      }
    }
```

```
  }
}
```

Лістинг 12. Код класу «UpdateUsersForm»

```
using FootballAcademy.AppCode;
using FootballAcademy.Providers;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace FootballAcademy.Forms.Systems {
  public partial class UpdateUsersForm : Form {
    private int _UserId = 0;
    private Users _selectedUser = new Users();
    private UsersProvider _UserProvider = new UsersProvider();
    private ValidationMy _validation = new ValidationMy();
    private RoleApp _RoleApp = new RoleApp();
    private List<Role> _RoleList = new List<Role>();

    public UpdateUsersForm(int UserId) {
      InitializeComponent();
      _UserId = UserId;
      LoadAllDate();
    }

    private void SaveBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
        _UserProvider.UpdateUsers(FirstNameTBox.Text, LastNameTBox.Text, UserLoginTbx.Text,
PasswordTbx.Text,
          Convert.ToInt32(RolesCBox.SelectedValue), DescriptionTbx.Text, _UserId);
        this.Close();
      }
    }

    private void DeleteBtn_Click(object sender, EventArgs e) {
      if (MessageBox.Show("Ви дійсно хочете видалити цей елемент?", "Видалити",
MessageBoxButtons.YesNo) == DialogResult.Yes) {
        _UserProvider.DeleteUsers(_UserId);
        this.Close();
      }
    }

    private void ExitBtn_Click(object sender, EventArgs e) {
      this.Close();
    }

    private void LoadAllDate() {
      _RoleList = _RoleApp.GetRoleList();
      RolesCBox.DataSource = _RoleList;
      RolesCBox.ValueMember = "RoleId";
      RolesCBox.DisplayMember = "RoleName";

      _selectedUser = _UserProvider.SelectedUsersByUsersId(_UserId);
      FirstNameTBox.Text = _selectedUser.FirstName;
      LastNameTBox.Text = _selectedUser.LastName;
```

```
      UserLoginTbx.Text = _selectedUser.UsersName;
      RolesCBox.SelectedValue = _selectedUser.RoleId;
    }

    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
      if (_validation.IsDataEntering(FirstNameTBox.Text)) {
        FirstNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        FirstNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(LastNameTBox.Text)) {
        LastNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        LastNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsPasswordMatch(PasswordTbx.Text, RePasswordTbx.Text)) {
        PasswordAndRePasswordDontMatchLbl.Visible = false;
      } else {
        PasswordAndRePasswordDontMatchLbl.Visible = true;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(UserLoginTbx.Text)) {
        UserLoginValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        UserLoginValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(PasswordTbx.Text)) {
        PasswordValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        PasswordValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(RePasswordTbx.Text)) {
        RePasswordValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        RePasswordValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }
  }
}
```

Лістинг 13. Код класу «UsersForm»

```
using FootballAcademy.AppCode;
using FootballAcademy.Providers;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace FootballAcademy.Forms.Systems {
```

```
  public partial class UsersForm : Form {
    private int _selectedRowIndex = 0;
    ValidationMy _validation = new ValidationMy();
    private UsersProvider _UserProvider = new UsersProvider();
    private List<Users> _UserList = new List<Users>();
    private RoleApp _RoleApp = new RoleApp();
    private List<Role> _RoleList = new List<Role>();
    public UsersForm() {
      InitializeComponent();
      LoadAllDate();
      DataLoad();
    }

    private void AddBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
        _UserProvider.InsertUsers(FirstNameTBox.Text, LastNameTBox.Text, UserLoginTbx.Text,
PasswordTbx.Text,
          Convert.ToInt32(RolesCBox.SelectedValue), DescriptionTbx.Text);
        DataLoad();
        ClearAllControls();
      }
    }

    private void ClearBtn_Click(object sender, EventArgs e) {
      ClearAllControls();
    }

    private void ExitBtn_Click(object sender, EventArgs e) {
      this.Close();
    }

    private void DataLoad() {
      int firstRowIndex = 0;
      if (UsersGridView.FirstDisplayedScrollingRowIndex > 0) {
        firstRowIndex = UsersGridView.FirstDisplayedScrollingRowIndex;
      }
      try {
        _UserList = _UserProvider.GetAllUsers();
        LoadDataInKlientGridView(_UserList);
        if (_selectedRowIndex == UsersGridView.Rows.Count) {
          _selectedRowIndex = UsersGridView.Rows.Count - 1;
        }
        if (_selectedRowIndex >= 0) {
          UsersGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;
          UsersGridView.Rows[_selectedRowIndex].Selected = true;
        }
      } catch { }
    }

    private void LoadDataInKlientGridView(List<Users> UserList) {
      UsersGridView.DataSource = null;
      UsersGridView.Columns.Clear();
      UsersGridView.AutoGenerateColumns = false;
      UsersGridView.RowHeadersVisible = false;

      UsersGridView.DataSource = UserList;

      if (UserList.Count > 0) {
        if (UserList[0].Message == NamesMy.NoDataNames.NoDataInUsers) {
          DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
          messageColumn.DataPropertyName = "Message";
          messageColumn.Width = UsersGridView.Width - NamesMy.SizeOptins.MinusSizePanel;
          UsersGridView.Columns.Add(messageColumn);
        } else {
```

```
            DataGridViewColumn deviseIdColumn = new DataGridViewTextBoxColumn();
            deviseIdColumn.DataPropertyName = "UsersId";
            UsersGridView.Columns.Add(deviseIdColumn);
            UsersGridView.Columns[0].Visible = false;

            DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
            numberColumn.HeaderText = "№ п/п";
            numberColumn.DataPropertyName = "Number";
            numberColumn.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
            numberColumn.Width = NamesMy.SizeOptins.NumberSize;
            UsersGridView.Columns.Add(numberColumn);

            DataGridViewColumn lastNameColumn = new DataGridViewTextBoxColumn();
            lastNameColumn.HeaderText = "Фимилия";
            lastNameColumn.DataPropertyName = "LastName";
            lastNameColumn.Width = NamesMy.SizeOptins.Name;
            UsersGridView.Columns.Add(lastNameColumn);

            DataGridViewColumn firstNameColumn = new DataGridViewTextBoxColumn();
            firstNameColumn.HeaderText = "Имя";
            firstNameColumn.DataPropertyName = "FirstName";
            firstNameColumn.Width = NamesMy.SizeOptins.Name;
            UsersGridView.Columns.Add(firstNameColumn);

            DataGridViewColumn UserNameColumn = new DataGridViewTextBoxColumn();
            UserNameColumn.HeaderText = "Логин";
            UserNameColumn.DataPropertyName = "UsersName";
            UserNameColumn.Width = NamesMy.SizeOptins.Name;
            UsersGridView.Columns.Add(UserNameColumn);

            DataGridViewColumn roleNameColumn = new DataGridViewTextBoxColumn();
            roleNameColumn.HeaderText = "Роль";
            roleNameColumn.DataPropertyName = "RoleName";
            roleNameColumn.Width = NamesMy.SizeOptins.Name;
            UsersGridView.Columns.Add(roleNameColumn);
          }
          for (int i = 0; i < UsersGridView.Columns.Count; i++) {
            UsersGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
          }
        }
      }

      private void ClearAllControls() {
        FirstNameTBox.Text = String.Empty;
        LastNameTBox.Text = String.Empty;
        DescriptionTbx.Text = String.Empty;
        UserLoginTbx.Text = String.Empty;
        PasswordTbx.Text = String.Empty;
        RePasswordTbx.Text = String.Empty;
      }

      private bool IsDataEnteringCorrect() {
        bool isCorrect = true;
        if (_validation.IsDataEntering(FirstNameTBox.Text)) {
          FirstNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
        } else {
          FirstNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
          isCorrect = false;
        }
        if (_validation.IsDataEntering(LastNameTBox.Text)) {
          LastNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
        } else {
          LastNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
```

```
        isCorrect = false;
      }
      if (_validation.IsPasswordMatch(PasswordTbx.Text, RePasswordTbx.Text)) {
        PasswordAndRePasswordDontMatchLbl.Visible = false;
        PasswordAndRePasswordDontMatchLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        PasswordAndRePasswordDontMatchLbl.Visible = true;
        PasswordAndRePasswordDontMatchLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(UserLoginTbx.Text)) {
        UserLoginValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        UserLoginValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(PasswordTbx.Text)) {
        PasswordValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        PasswordValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(RePasswordTbx.Text)) {
        RePasswordValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        RePasswordValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }

    private void LoadAllDate() {
      _RoleList = _RoleApp.GetRoleList();
      RolesCBox.DataSource = _RoleList;
      RolesCBox.ValueMember = "RoleId";
      RolesCBox.DisplayMember = "RoleName";
    }

    private void UsersGridView_CellClick(object sender, DataGridViewCellEventArgs e) {
      if (e.RowIndex >= 0 && UsersGridView[0, e.RowIndex].Value.ToString() !=
_UserList[0].Message) {
        _selectedRowIndex = e.RowIndex;
        UpdateUsersForm updateUsersForm = new
UpdateUsersForm(Convert.ToInt32(UsersGridView[0, e.RowIndex].Value.ToString()));
        updateUsersForm.ShowDialog();
        DataLoad();
      }
    }
  }
}
```

Лістинг 14. Код класу «GroupsProvider»

```
using FootballAcademy.AppCode;
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FootballAcademy.Providers {
```

```
  class GroupsProvider {
    private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings["CONNECTSQL"];

    public void InsertGroups(string GroupsName, string Description, int TrenerId) {
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = "INSERT INTO Clusters (GroupsName, Description, TrenerId)
VALUES(@GroupsName, @Description, @TrenerId)";
      MySqlCommand cmd = new MySqlCommand(query, connection);
      cmd.Parameters.AddWithValue("@GroupsName", GroupsName);
      cmd.Parameters.AddWithValue("@Description", Description);
      cmd.Parameters.AddWithValue("@TrenerId", TrenerId);
      connection.Open();
      cmd.ExecuteNonQuery();
      connection.Close();
      connection.Dispose();
      connection = null;
    }

    public Groups SelectedGroupsByGroupsId(int GroupsId) {
      Groups selectedGroups = new Groups();
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = "SELECT * FROM Clusters WHERE GroupsId=" + GroupsId.ToString();
      MySqlCommand command = new MySqlCommand(query, connection);
      connection.Open();
      MySqlDataReader reader = command.ExecuteReader();
      while (reader.Read()) {
        selectedGroups.GroupsId = Convert.ToInt32(reader["GroupsId"]);
        selectedGroups.GroupsName = reader["GroupsName"].ToString();
        selectedGroups.Description = reader["Description"].ToString();
        selectedGroups.TrenerId = Convert.ToInt32(reader["TrenerId"]);
      }
      reader.Close();
      connection.Close();
      return selectedGroups;
    }

    public Groups SelectedGroupsByTrenerId(int TrenerId) {
      Groups selectedGroups = new Groups();
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = "SELECT * FROM Clusters WHERE TrenerId=" + TrenerId.ToString();
      MySqlCommand command = new MySqlCommand(query, connection);
      connection.Open();
      MySqlDataReader reader = command.ExecuteReader();
      while (reader.Read()) {
        selectedGroups.GroupsId = Convert.ToInt32(reader["GroupsId"]);
        selectedGroups.GroupsName = reader["GroupsName"].ToString();
        selectedGroups.Description = reader["Description"].ToString();
        selectedGroups.TrenerId = Convert.ToInt32(reader["TrenerId"]);
      }
      reader.Close();
      connection.Close();
      return selectedGroups;
    }


    public List<Groups> GetAllGroups() {
      int i = 0;
      List<Groups> GroupsList = new List<Groups>();
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = "SELECT * FROM Clusters ";
      MySqlCommand command = new MySqlCommand(query, connection);
      connection.Open();
      MySqlDataReader reader = command.ExecuteReader();
```

```
      while (reader.Read()) {
        Groups selectedGroups = new Groups();
        selectedGroups.Number = ++i;
        selectedGroups.GroupsId = Convert.ToInt32(reader["GroupsId"]);
        selectedGroups.GroupsName = reader["GroupsName"].ToString();
        selectedGroups.Description = reader["Description"].ToString();
        selectedGroups.TrenerId = Convert.ToInt32(reader["TrenerId"]);
        GroupsList.Add(selectedGroups);
      }
      reader.Close();
      connection.Close();

      if (GroupsList.Count == 0) {
        Groups noGroups = new Groups();
        noGroups.GroupsId = 0;
        noGroups.Message = NamesMy.NoDataNames.NoDataInGroup;
        GroupsList.Add(noGroups);
      }
      return GroupsList;
    }

    public void UpdateGroups(string GroupsName, string Description, int TrenerId, int
GroupsId) {
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = "UPDATE Clusters SET GroupsName = @GroupsName, Description =
@Description, TrenerId=@TrenerId " +
          " WHERE GroupsId = @GroupsId";
      MySqlCommand cmd = new MySqlCommand(query, connection);
      cmd.Parameters.AddWithValue("@GroupsName", GroupsName);
      cmd.Parameters.AddWithValue("@Description", Description);
      cmd.Parameters.AddWithValue("@TrenerId", TrenerId);
      cmd.Parameters.AddWithValue("@GroupsId", GroupsId);
      connection.Open();
      cmd.ExecuteNonQuery();
      connection.Close();
      connection.Dispose();
      connection = null;
    }

    public void DeleteGroups(int GroupsId) {
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = "DELETE FROM Clusters WHERE GroupsId = @GroupsId";
      MySqlCommand cmd = new MySqlCommand(query, connection);
      cmd.Parameters.AddWithValue("@GroupsId", GroupsId);
      connection.Open();
      cmd.ExecuteNonQuery();
      connection.Close();
      connection.Dispose();
      connection = null;
    }

  }
}


public class Groups {
  private int _Number;
  private int _GroupsId;
  private string _GroupsName;
  private string _Description;
  private int _TrenerId;
  private string _Message;

  public Groups() {
```

```
    _Number = 0;
    _GroupsId = 0;
    _GroupsName = String.Empty;
    _Description = String.Empty;
    _TrenerId = 0;
    _Message = String.Empty;
  }

  public int Number {
    set { _Number = value; }
    get { return _Number; }
  }
  public int GroupsId {
    set { _GroupsId = value; }
    get { return _GroupsId; }
  }
  public string GroupsName {
    set { _GroupsName = value; }
    get { return _GroupsName; }
  }
  public string Description {
    set { _Description = value; }
    get { return _Description; }
  }
  public int TrenerId {
    set { _TrenerId = value; }
    get { return _TrenerId; }
  }
  public string Message {
    set { _Message = value; }
    get { return _Message; }
  }
}
```

Лістинг 15. Код класу «LogsProvider»

```
using FootballAcademy.AppCode;
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FootballAcademy.Providers {
  class LogsProvider {
    private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings["CONNECTSQL"];

    public void InsertLogs(int UsersId, string EventNameShow, DateTime EventDate) {
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = "INSERT INTO Logs (UsersId, EventNameShow, EventDate) VALUES(@UsersId,
@EventNameShow, @EventDate)";
      MySqlCommand cmd = new MySqlCommand(query, connection);
      cmd.Parameters.AddWithValue("UsersId", UsersId);
      cmd.Parameters.AddWithValue("EventNameShow", EventNameShow);
      cmd.Parameters.AddWithValue("EventDate", EventDate.ToString("yyyy-MM-dd HH:mm:ss"));
      connection.Open();
      cmd.ExecuteNonQuery();
      connection.Close();
    }
```

```
    public List<Logs> GetAllLogs() {
      int i = 0;
      List<Logs> LogsList = new List<Logs>();
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = "SELECT Logs.LogsId, Logs.UsersId, Logs.EventNameShow, Logs.EventDate,
Users.UsersName " +
      "FROM Logs INNER JOIN Users ON Users.UsersId = Logs.UsersId  ORDER BY Logs.EventDate
DESC";
      MySqlCommand command = new MySqlCommand(query, connection);
      connection.Open();
      MySqlDataReader reader = command.ExecuteReader();
      while (reader.Read()) {
        Logs selectedLogs = new Logs();
        selectedLogs.Number = ++i;
        selectedLogs.LogsId = Convert.ToInt32(reader["LogsId"]);
        selectedLogs.UsersId = Convert.ToInt32(reader["UsersId"]);
        selectedLogs.EventNameShow = reader["EventNameShow"].ToString();
        selectedLogs.EventDate = Convert.ToDateTime(reader["EventDate"]);
        selectedLogs.UsersName = reader["UsersName"].ToString();
        LogsList.Add(selectedLogs);
      }
      reader.Close();
      connection.Close();

      if (LogsList.Count == 0) {
        Logs noLogs = new Logs();
        noLogs.LogsId = 0;
        noLogs.Message = NamesMy.NoDataNames.NoDataInLogs;
        LogsList.Add(noLogs);
      }
      return LogsList;
    }

  }
}


public class Logs {
  private int _Number;
  private int _LogsId;
  private int _UsersId;
  private string _UsersName;
  private string _EventNameShow;
  private DateTime _EventDate;
  private string _Message;

  public Logs() {
    _Number = 0;
    _LogsId = 0;
    _UsersId = 0;
    _UsersName = String.Empty;
    _EventNameShow = String.Empty;
    _EventDate = new DateTime();
    _Message = String.Empty;
  }

  public int Number {
    set { _Number = value; }
    get { return _Number; }
  }
  public int LogsId {
    set { _LogsId = value; }
    get { return _LogsId; }
```

```
  }
  public int UsersId {
    set { _UsersId = value; }
    get { return _UsersId; }
  }
  public string UsersName {
    set { _UsersName = value; }
    get { return _UsersName; }
  }
  public string EventNameShow {
    set { _EventNameShow = value; }
    get { return _EventNameShow; }
  }
  public DateTime EventDate {
    set { _EventDate = value; }
    get { return _EventDate; }
  }
  public string Message {
    set { _Message = value; }
    get { return _Message; }
  }

      }
```

Лістинг 16. Код класу «ScheduleLProvider»

```
using FootballAcademy.AppCode;
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FootballAcademy.Providers {
  class ScheduleLProvider {
    private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings["CONNECTSQL"];

    public void InsertBatchScheduleL(List<ScheduleL> ScheduleL) {
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = "INSERT INTO ScheduleL (SoccerPlayerId, TrenerId, GroupId, ScheduleId)
VALUES(@SoccerPlayerId, @TrenerId, @GroupId, @ScheduleId)";
      MySqlCommand cmd = new MySqlCommand(query, connection);
      connection.Open();
      for (int i = 0; i < ScheduleL.Count; i++) {
        cmd.Parameters.AddWithValue("SoccerPlayerId", ScheduleL[i].SoccerPlayerId);
        cmd.Parameters.AddWithValue("TrenerId", ScheduleL[i].TrenerId);
        cmd.Parameters.AddWithValue("GroupId", ScheduleL[i].GroupId);
        cmd.Parameters.AddWithValue("ScheduleId", ScheduleL[i].ScheduleId);
        cmd.ExecuteNonQuery();
        while (cmd.Parameters.Count > 0) {
          cmd.Parameters.RemoveAt(0);
        }
      }
      connection.Close();
    }


    public List<ScheduleL> GetAllScheduleLByScheduleId(int ScheduleId) {
      SoccerPlayerProvider _SoccerPlayerProvider = new SoccerPlayerProvider();
      List<SoccerPlayer> SoccerPlayerList = new List<SoccerPlayer>();
      SoccerPlayerList = _SoccerPlayerProvider.GetAllSoccerPlayer();
```

```
      int i = 0;
      List<ScheduleL> ScheduleLList = new List<ScheduleL>();
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = "SELECT * FROM ScheduleL WHERE ScheduleId=" + ScheduleId.ToString();
      MySqlCommand command = new MySqlCommand(query, connection);
      connection.Open();
      MySqlDataReader reader = command.ExecuteReader();
      while (reader.Read()) {
        ScheduleL selectedScheduleL = new ScheduleL();
        selectedScheduleL.Number = ++i;
        selectedScheduleL.ScheduleLId = Convert.ToInt32(reader["ScheduleLId"]);
        selectedScheduleL.SoccerPlayerId = Convert.ToInt32(reader["SoccerPlayerId"]);
        selectedScheduleL.TrenerId = Convert.ToInt32(reader["TrenerId"]);
        selectedScheduleL.GroupId = Convert.ToInt32(reader["GroupId"]);
        selectedScheduleL.ScheduleId = Convert.ToInt32(reader["ScheduleId"]);
        ScheduleLList.Add(selectedScheduleL);
      }
      reader.Close();
      connection.Close();

      for (int j = 0; j < ScheduleLList.Count; j++) {
        ScheduleLList[j].FIO = GetSoccerPlayerFIO(ScheduleLList[j].SoccerPlayerId,
SoccerPlayerList);
      }

      if (ScheduleLList.Count == 0) {
        ScheduleL noScheduleL = new ScheduleL();
        noScheduleL.ScheduleLId = 0;
        noScheduleL.Message = NamesMy.NoDataNames.NoDataInScheduleL;
        ScheduleLList.Add(noScheduleL);
      }
      return ScheduleLList;
    }

    private string GetSoccerPlayerFIO(int SoccerPlayerId, List<SoccerPlayer>
SoccerPlayerList) {
      for (int i = 0; i < SoccerPlayerList.Count; i++) {
        if (SoccerPlayerId == SoccerPlayerList[i].SoccerPlayerId) {
          return SoccerPlayerList[i].FIO;
        }
      }
      return "";
    }

    public void DeleteScheduleL(int ScheduleId) {
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = "DELETE FROM ScheduleL  WHERE ScheduleId = @ScheduleId";
      MySqlCommand cmd = new MySqlCommand(query, connection);
      cmd.Parameters.AddWithValue("@ScheduleId", ScheduleId);
      connection.Open();
      cmd.ExecuteNonQuery();
      connection.Close();
      connection.Dispose();
      connection = null;
    }

  }
}


public class ScheduleL {
  private int _Number;
  private int _ScheduleLId;
  private int _SoccerPlayerId;
```

```
  private string _FIO;
  private int _GroupId;
  private int _TrenerId;
  private int _ScheduleId;
  private string _Message;

  public ScheduleL() {
    _Number = 0;
    _ScheduleLId = 0;
    _SoccerPlayerId = 0;
    _FIO = String.Empty;
    _GroupId = 0;
    _TrenerId = 0;
    ScheduleId = 0;
    _Message = String.Empty;
  }

  public int Number {
    set { _Number = value; }
    get { return _Number; }
  }
  public int ScheduleLId {
    set { _ScheduleLId = value; }
    get { return _ScheduleLId; }
  }
  public int SoccerPlayerId {
    set { _SoccerPlayerId = value; }
    get { return _SoccerPlayerId; }
  }
  public string FIO {
    set { _FIO = value; }
    get { return _FIO; }
  }
  public int GroupId {
    set { _GroupId = value; }
    get { return _GroupId; }
  }
  public int TrenerId {
    set { _TrenerId = value; }
    get { return _TrenerId; }
  }
  public int ScheduleId {
    set { _ScheduleId = value; }
    get { return _ScheduleId; }
  }
 public string Message {
    set { _Message = value; }
    get { return _Message; }
  }
}
```

Лістинг 17. Код класу «ScheduleProvider»

```
using FootballAcademy.AppCode;
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Data.OleDb;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace FootballAcademy.Providers {
  class ScheduleProvider {
    private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings["CONNECTSQL"];

    public void InsertSchedule(string ScheduleName, DateTime DateOfOccupation, DateTime
StartDate, DateTime EndDate, int TrenerId) {
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = "INSERT INTO Schedule (ScheduleName, DateOfOccupation, StartDate,
EndDate, TrenerId) VALUES(@ScheduleName, @DateOfOccupation, @StartDate, @EndDate,
@TrenerId)";
      MySqlCommand cmd = new MySqlCommand(query, connection);
      cmd.Parameters.AddWithValue("@ScheduleName", ScheduleName);
      cmd.Parameters.AddWithValue("@DateOfOccupation", DateOfOccupation.ToString("yyyy-MM-dd
HH:mm:ss"));
      cmd.Parameters.AddWithValue("@StartDate", StartDate.ToString("yyyy-MM-dd HH:mm:ss"));
      cmd.Parameters.AddWithValue("@EndDate", EndDate.ToString("yyyy-MM-dd HH:mm:ss"));
      cmd.Parameters.AddWithValue("@TrenerId", TrenerId);
      connection.Open();
      cmd.ExecuteNonQuery();
      connection.Close();
      connection.Dispose();
      connection = null;
    }

    public Schedule SelectedScheduleByScheduleId(int ScheduleId) {
      Schedule selectedSchedule = new Schedule();
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = "SELECT * FROM Schedule WHERE ScheduleId=" + ScheduleId.ToString();
      MySqlCommand command = new MySqlCommand(query, connection);
      connection.Open();
      MySqlDataReader reader = command.ExecuteReader();
      while (reader.Read()) {
        selectedSchedule.ScheduleId = Convert.ToInt32(reader["ScheduleId"]);
        selectedSchedule.ScheduleName = reader["ScheduleName"].ToString();
        selectedSchedule.DateOfOccupation = Convert.ToDateTime(reader["DateOfOccupation"]);
        selectedSchedule.StartDate = Convert.ToDateTime(reader["StartDate"]);
        selectedSchedule.EndDate = Convert.ToDateTime(reader["EndDate"]);
        selectedSchedule.TrenerId = Convert.ToInt32(reader["TrenerId"]);
      }
      reader.Close();
      connection.Close();
      return selectedSchedule;
    }


    public List<Schedule> GetAllSchedule() {
      int i = 0;
      List<Schedule> ScheduleList = new List<Schedule>();
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = "SELECT * FROM Schedule ";
      MySqlCommand command = new MySqlCommand(query, connection);
      connection.Open();
      MySqlDataReader reader = command.ExecuteReader();
      while (reader.Read()) {
        Schedule selectedSchedule = new Schedule();
        selectedSchedule.Number = ++i;
        selectedSchedule.ScheduleId = Convert.ToInt32(reader["ScheduleId"]);
        selectedSchedule.ScheduleName = reader["ScheduleName"].ToString();
        selectedSchedule.DateOfOccupation = Convert.ToDateTime(reader["DateOfOccupation"]);
        selectedSchedule.StartDate = Convert.ToDateTime(reader["StartDate"]);
        selectedSchedule.EndDate = Convert.ToDateTime(reader["EndDate"]);
        selectedSchedule.TrenerId = Convert.ToInt32(reader["TrenerId"]);
```

```
        ScheduleList.Add(selectedSchedule);
      }
      reader.Close();
      connection.Close();

      if (ScheduleList.Count == 0) {
        Schedule noSchedule = new Schedule();
        noSchedule.ScheduleId = 0;
        noSchedule.Message = NamesMy.NoDataNames.NoDataInSchedule;
        ScheduleList.Add(noSchedule);
      }
      return ScheduleList;
    }

    public void UpdateSchedule(string ScheduleName, DateTime DateOfOccupation, DateTime
StartDate, DateTime EndDate, int TrenerId, int ScheduleId) {
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = "UPDATE Schedule SET ScheduleName = @ScheduleName, DateOfOccupation =
@DateOfOccupation " +
          " WHERE ScheduleId = @ScheduleId";
      MySqlCommand cmd = new MySqlCommand(query, connection);
      cmd.Parameters.AddWithValue("@ScheduleName", ScheduleName);
      cmd.Parameters.AddWithValue("@DateOfOccupation", DateOfOccupation.ToString("yyyy-MM-dd
HH:mm:ss"));
      cmd.Parameters.AddWithValue("@StartDate", StartDate.ToString("yyyy-MM-dd HH:mm:ss"));
      cmd.Parameters.AddWithValue("@EndDate", EndDate.ToString("yyyy-MM-dd HH:mm:ss"));
      cmd.Parameters.AddWithValue("@TrenerId", TrenerId);
cmd.Parameters.AddWithValue("@ScheduleId", ScheduleId);
      connection.Open();
      cmd.ExecuteNonQuery();
      connection.Close();
      connection.Dispose();
      connection = null;
    }

    public void DeleteSchedule(int ScheduleId) {
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = "DELETE FROM Schedule WHERE ScheduleId = @ScheduleId";
      MySqlCommand cmd = new MySqlCommand(query, connection);
      cmd.Parameters.AddWithValue("@ScheduleId", ScheduleId);
      connection.Open();
      cmd.ExecuteNonQuery();
      connection.Close();
      connection.Dispose();
      connection = null;
    }

    public int GetLastRecords() {
      Schedule selectedSchedule = new Schedule();
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = "SELECT ScheduleId FROM Schedule WHERE ScheduleId=(SELECT
MAX(ScheduleId) FROM Schedule)";
      MySqlCommand command = new MySqlCommand(query, connection);
      connection.Open();
      MySqlDataReader reader = command.ExecuteReader();
      while (reader.Read()) {
        selectedSchedule.ScheduleId= Convert.ToInt32(reader["ScheduleId"]);
      }
      reader.Close();
      connection.Close();
      return selectedSchedule.ScheduleId;
    }

  }
```

```
}


public class Schedule {
  private int _Number;
  private int _ScheduleId;
  private string _ScheduleName;
  private DateTime _DateOfOccupation;
  private DateTime _StartDate;
  private DateTime _EndDate;
  private int _TrenerId;
  private string _Message;

  public Schedule() {
    _Number = 0;
    _ScheduleId = 0;
    _ScheduleName = String.Empty;
    _DateOfOccupation = new DateTime();
    _StartDate = new DateTime();
    _EndDate = new DateTime();
    _TrenerId = 0;
    _Message = String.Empty;
  }

  public int Number {
    set { _Number = value; }
    get { return _Number; }
  }
  public int ScheduleId {
    set { _ScheduleId = value; }
    get { return _ScheduleId; }
  }
  public string ScheduleName {
    set { _ScheduleName = value; }
    get { return _ScheduleName; }
  }
  public DateTime DateOfOccupation {
    set { _DateOfOccupation = value; }
    get { return _DateOfOccupation; }
  }
  public DateTime StartDate {
    set { _StartDate = value; }
    get { return _StartDate; }
  }
  public DateTime EndDate {
    set { _EndDate = value; }
    get { return _EndDate; }
  }
  public int TrenerId {
    set { _TrenerId = value; }
    get { return _TrenerId; }
  }
  public string Message {
    set { _Message = value; }
    get { return _Message; }
  }
}
```

Лістинг 18. Код класу «SoccerPlayerProvider»

```
using FootballAcademy.AppCode;
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FootballAcademy.Providers {
  class SoccerPlayerProvider {
    private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings["CONNECTSQL"];

    public void InsertSoccerPlayer(string FirstName, string LastName, string Phone, DateTime
BirthDate, string Address, string Email,int GroupId) {
      MySqlConnection connection = new MySqlConnection(_ConnString);

      string query = "INSERT INTO SoccerPlayer (FirstName, LastName, Phone, BirthDate,
Address, Email, GroupId)" +
        " VALUES(@FirstName, @LastName, @Phone, @BirthDate, @Address, @Email, @GroupId)";
      MySqlCommand cmd = new MySqlCommand(query, connection);
      cmd.Parameters.AddWithValue("@FirstName", FirstName);
      cmd.Parameters.AddWithValue("@LastName", LastName);
      cmd.Parameters.AddWithValue("@Phone", Phone);
      cmd.Parameters.AddWithValue("@BirthDate", BirthDate.ToString("yyyy-MM-dd HH:mm:ss"));
      cmd.Parameters.AddWithValue("@Address", Address);
      cmd.Parameters.AddWithValue("@Email", Email);
      cmd.Parameters.AddWithValue("@GroupId", GroupId);
      connection.Open();
      cmd.ExecuteNonQuery();
      connection.Close();
      connection.Dispose();
      connection = null;
    }

    public SoccerPlayer SelectedSoccerPlayerBySoccerPlayerId(int SoccerPlayerId) {
      SoccerPlayer selectedSoccerPlayer = new SoccerPlayer();
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = "SELECT * FROM SoccerPlayer WHERE SoccerPlayerId=" +
SoccerPlayerId.ToString();
      MySqlCommand command = new MySqlCommand(query, connection);
      connection.Open();
      MySqlDataReader reader = command.ExecuteReader();
      while (reader.Read()) {
        selectedSoccerPlayer.SoccerPlayerId = Convert.ToInt32(reader["SoccerPlayerId"]);
        selectedSoccerPlayer.FirstName = reader["FirstName"].ToString();
        selectedSoccerPlayer.LastName = reader["LastName"].ToString();
        selectedSoccerPlayer.FIO = reader["LastName"].ToString() + " " +
reader["FirstName"].ToString();
        selectedSoccerPlayer.Phone = reader["Phone"].ToString();
        selectedSoccerPlayer.BirthDate = Convert.ToDateTime(reader["BirthDate"]);
        selectedSoccerPlayer.Address = reader["Address"].ToString();
        selectedSoccerPlayer.Email = reader["Email"].ToString();
        selectedSoccerPlayer.GroupId = Convert.ToInt32(reader["GroupId"]);
      }
      reader.Close();
      connection.Close();
      return selectedSoccerPlayer;
    }


    public List<SoccerPlayer> GetAllSoccerPlayer() {
      int i = 0;
      List<SoccerPlayer> SoccerPlayerList = new List<SoccerPlayer>();
```

```
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = "SELECT * FROM SoccerPlayer ";
      MySqlCommand command = new MySqlCommand(query, connection);
      connection.Open();
      MySqlDataReader reader = command.ExecuteReader();
      while (reader.Read()) {
        SoccerPlayer selectedSoccerPlayer = new SoccerPlayer();
        selectedSoccerPlayer.Number = ++i;
        selectedSoccerPlayer.SoccerPlayerId = Convert.ToInt32(reader["SoccerPlayerId"]);
        selectedSoccerPlayer.FirstName = reader["FirstName"].ToString();
        selectedSoccerPlayer.LastName = reader["LastName"].ToString();
        selectedSoccerPlayer.FIO = reader["LastName"].ToString() + " " +
reader["FirstName"].ToString();
        selectedSoccerPlayer.Phone = reader["Phone"].ToString();
        selectedSoccerPlayer.BirthDate = Convert.ToDateTime(reader["BirthDate"]);
        selectedSoccerPlayer.Address = reader["Address"].ToString();
        selectedSoccerPlayer.Email = reader["Email"].ToString();
        selectedSoccerPlayer.GroupId = Convert.ToInt32(reader["GroupId"]);
        SoccerPlayerList.Add(selectedSoccerPlayer);
      }
      reader.Close();
      connection.Close();

      if (SoccerPlayerList.Count == 0) {
        SoccerPlayer noSoccerPlayer = new SoccerPlayer();
        noSoccerPlayer.SoccerPlayerId = 0;
        noSoccerPlayer.Message = NamesMy.NoDataNames.NoDataInSoccerPlayer;
        SoccerPlayerList.Add(noSoccerPlayer);
      }
      return SoccerPlayerList;
    }

    public List<SoccerPlayer> GetAllSoccerPlayerByGroupId(int GroupId) {
      int i = 0;
      List<SoccerPlayer> SoccerPlayerList = new List<SoccerPlayer>();
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = "SELECT * FROM SoccerPlayer WHERE GroupId=" + GroupId.ToString();
      MySqlCommand command = new MySqlCommand(query, connection);
      connection.Open();
      MySqlDataReader reader = command.ExecuteReader();
      while (reader.Read()) {
        SoccerPlayer selectedSoccerPlayer = new SoccerPlayer();
        selectedSoccerPlayer.Number = ++i;
        selectedSoccerPlayer.SoccerPlayerId = Convert.ToInt32(reader["SoccerPlayerId"]);
        selectedSoccerPlayer.FirstName = reader["FirstName"].ToString();
        selectedSoccerPlayer.LastName = reader["LastName"].ToString();
        selectedSoccerPlayer.FIO = reader["LastName"].ToString() + " " +
reader["FirstName"].ToString();
        selectedSoccerPlayer.Phone = reader["Phone"].ToString();
        selectedSoccerPlayer.BirthDate = Convert.ToDateTime(reader["BirthDate"]);
        selectedSoccerPlayer.Address = reader["Address"].ToString();
        selectedSoccerPlayer.Email = reader["Email"].ToString();
        selectedSoccerPlayer.GroupId = Convert.ToInt32(reader["GroupId"]);
        SoccerPlayerList.Add(selectedSoccerPlayer);
      }
      reader.Close();
      connection.Close();

      if (SoccerPlayerList.Count == 0) {
        SoccerPlayer noSoccerPlayer = new SoccerPlayer();
        noSoccerPlayer.SoccerPlayerId = 0;
        noSoccerPlayer.Message = NamesMy.NoDataNames.NoDataInSoccerPlayer;
        SoccerPlayerList.Add(noSoccerPlayer);
      }
```

```
      return SoccerPlayerList;
    }

    public void UpdateSoccerPlayer(string FirstName, string LastName, string Phone, DateTime
BirthDate, string Address, string Email, int GroupId, int SoccerPlayerId) {
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = "UPDATE SoccerPlayer SET FirstName=@FirstName, LastName=@LastName,
Phone=@Phone, BirthDate=@BirthDate, Address=@Address, Email=@Email, GroupId=@GroupId" +
          " WHERE SoccerPlayerId = @SoccerPlayerId";
      MySqlCommand cmd = new MySqlCommand(query, connection);
      cmd.Parameters.AddWithValue("@FirstName", FirstName);
      cmd.Parameters.AddWithValue("@LastName", LastName);
      cmd.Parameters.AddWithValue("@Phone", Phone);
      cmd.Parameters.AddWithValue("@BirthDate", BirthDate.ToString("yyyy-MM-dd HH:mm:ss"));
      cmd.Parameters.AddWithValue("@Address", Address);
      cmd.Parameters.AddWithValue("@Email", Email);
      cmd.Parameters.AddWithValue("@GroupId", GroupId);
      cmd.Parameters.AddWithValue("@SoccerPlayerId", SoccerPlayerId);
      connection.Open();
      cmd.ExecuteNonQuery();
      connection.Close();
      connection.Dispose();
      connection = null;
    }

    public void DeleteSoccerPlayer(int SoccerPlayerId) {
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = "DELETE FROM SoccerPlayer WHERE SoccerPlayerId = @SoccerPlayerId";
      MySqlCommand cmd = new MySqlCommand(query, connection);
      cmd.Parameters.AddWithValue("@SoccerPlayerId", SoccerPlayerId);
      connection.Open();
      cmd.ExecuteNonQuery();
      connection.Close();
      connection.Dispose();
      connection = null;
    }

  }
}



public class SoccerPlayer {
  private int _Number;
  private int _SoccerPlayerId;
  private string _FirstName;
  private string _LastName;
  private string _FIO;
  private string _Phone;
  private DateTime _BirthDate;
  private string _Address;
  private string _Email;
  private int _GroupId;

  private string _Message;

  public SoccerPlayer() {
    _Number = 0;
    _SoccerPlayerId = 0;
    _FirstName = String.Empty;
    _LastName = String.Empty;
    _FIO = String.Empty;
    _Phone = String.Empty;
    _BirthDate = new DateTime();
```

```
    _Address = String.Empty;
    _Email = String.Empty;
    _GroupId = 0;
    _Message = String.Empty;
  }

  public int Number {
    set { _Number = value; }
    get { return _Number; }
  }
  public int SoccerPlayerId {
    set { _SoccerPlayerId = value; }
    get { return _SoccerPlayerId; }
  }
  public string FirstName {
    set { _FirstName = value; }
    get { return _FirstName; }
  }
  public string LastName {
    set { _LastName = value; }
    get { return _LastName; }
  }
  public string FIO {
    set { _FIO = value; }
    get { return _FIO; }
  }
  public string Phone {
    set { _Phone = value; }
    get { return _Phone; }
  }
  public DateTime BirthDate {
    set { _BirthDate = value; }
    get { return _BirthDate; }
  }
  public string Address {
    set { _Address = value; }
    get { return _Address; }
  }
  public string Email {
    set { _Email = value; }
    get { return _Email; }
  }
  public int GroupId {
    set { _GroupId = value; }
    get { return _GroupId; }
  }
  public string Message {
    set { _Message = value; }
    get { return _Message; }
  }

      }
```

Лістинг 19. Код класу «TrenerProvider»

```
using FootballAcademy.AppCode;
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FootballAcademy.Providers {
  class TrenerProvider {
```

```
    private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings["CONNECTSQL"];

    public void InsertTrener(string FirstName, string LastName, string Phone, DateTime
BirthDate, string Address, string Email) {
      MySqlConnection connection = new MySqlConnection(_ConnString);

      string query = "INSERT INTO Trener (FirstName, LastName, Phone, BirthDate, Address,
Email)" +
        " VALUES(@FirstName, @LastName, @Phone, @BirthDate, @Address, @Email)";
      MySqlCommand cmd = new MySqlCommand(query, connection);
      cmd.Parameters.AddWithValue("@FirstName", FirstName);
      cmd.Parameters.AddWithValue("@LastName", LastName);
      cmd.Parameters.AddWithValue("@Phone", Phone);
      cmd.Parameters.AddWithValue("@BirthDate", BirthDate.ToString("yyyy-MM-dd HH:mm:ss"));
      cmd.Parameters.AddWithValue("@Address", Address);
      cmd.Parameters.AddWithValue("@Email", Email);
      connection.Open();
      cmd.ExecuteNonQuery();
      connection.Close();
      connection.Dispose();
      connection = null;
    }

    public Trener SelectedTrenerByTrenerId(int TrenerId) {
      Trener selectedTrener = new Trener();
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = "SELECT * FROM Trener WHERE TrenerId=" + TrenerId.ToString();
      MySqlCommand command = new MySqlCommand(query, connection);
      connection.Open();
      MySqlDataReader reader = command.ExecuteReader();
      while (reader.Read()) {
        selectedTrener.TrenerId = Convert.ToInt32(reader["TrenerId"]);
        selectedTrener.FirstName = reader["FirstName"].ToString();
        selectedTrener.LastName = reader["LastName"].ToString();
        selectedTrener.FIO = reader["LastName"].ToString() + " " +
reader["FirstName"].ToString();
        selectedTrener.Phone = reader["Phone"].ToString();
        selectedTrener.BirthDate = Convert.ToDateTime(reader["BirthDate"]);
        selectedTrener.Address = reader["Address"].ToString();
        selectedTrener.Email = reader["Email"].ToString();
      }
      reader.Close();
      connection.Close();
      return selectedTrener;
    }


    public List<Trener> GetAllTrener() {
      int i = 0;
      List<Trener> TrenerList = new List<Trener>();
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = "SELECT * FROM Trener ";
      MySqlCommand command = new MySqlCommand(query, connection);
      connection.Open();
      MySqlDataReader reader = command.ExecuteReader();
      while (reader.Read()) {
        Trener selectedTrener = new Trener();
        selectedTrener.Number = ++i;
        selectedTrener.TrenerId = Convert.ToInt32(reader["TrenerId"]);
        selectedTrener.FirstName = reader["FirstName"].ToString();
        selectedTrener.LastName = reader["LastName"].ToString();
        selectedTrener.FIO = reader["LastName"].ToString() + " " +
reader["FirstName"].ToString();
```

```
        selectedTrener.Phone = reader["Phone"].ToString();
        selectedTrener.BirthDate = Convert.ToDateTime(reader["BirthDate"]);
        selectedTrener.Address = reader["Address"].ToString();
        selectedTrener.Email = reader["Email"].ToString();
        TrenerList.Add(selectedTrener);
      }
      reader.Close();
      connection.Close();

      if (TrenerList.Count == 0) {
        Trener noTrener = new Trener();
        noTrener.TrenerId = 0;
        noTrener.Message = NamesMy.NoDataNames.NoDataInTrener;
        TrenerList.Add(noTrener);
      }
      return TrenerList;
    }

    public void UpdateTrener(string FirstName, string LastName, string Phone, DateTime
BirthDate, string Address, string Email, int TrenerId) {
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = "UPDATE Trener SET FirstName=@FirstName, LastName=@LastName,
Phone=@Phone, BirthDate=@BirthDate, Address=@Address, Email=@Email" +
          " WHERE TrenerId = @TrenerId";
      MySqlCommand cmd = new MySqlCommand(query, connection);
      cmd.Parameters.AddWithValue("@FirstName", FirstName);
      cmd.Parameters.AddWithValue("@LastName", LastName);
      cmd.Parameters.AddWithValue("@Phone", Phone);
      cmd.Parameters.AddWithValue("@BirthDate", BirthDate.ToString("yyyy-MM-dd HH:mm:ss"));
      cmd.Parameters.AddWithValue("@Address", Address);
      cmd.Parameters.AddWithValue("@Email", Email);
      cmd.Parameters.AddWithValue("@TrenerId", TrenerId);
      connection.Open();
      cmd.ExecuteNonQuery();
      connection.Close();
      connection.Dispose();
      connection = null;
    }

    public void DeleteTrener(int TrenerId) {
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = "DELETE FROM Trener WHERE TrenerId = @TrenerId";
      MySqlCommand cmd = new MySqlCommand(query, connection);
      cmd.Parameters.AddWithValue("@TrenerId", TrenerId);
      connection.Open();
      cmd.ExecuteNonQuery();
      connection.Close();
      connection.Dispose();
      connection = null;
    }

  }
}


public class Trener {
  private int _Number;
  private int _TrenerId;
  private string _FirstName;
  private string _LastName;
  private string _FIO;
  private string _Phone;
  private DateTime _BirthDate;
  private string _Address;
```

```
  private string _Email;

  private string _Message;

  public Trener() {
    _Number = 0;
    _TrenerId = 0;
    _FirstName = String.Empty;
    _LastName = String.Empty;
    _FIO = String.Empty;
    _Phone = String.Empty;
    _BirthDate = new DateTime();
    _Address = String.Empty;
    _Email = String.Empty;
    _Message = String.Empty;
  }

  public int Number {
    set { _Number = value; }
    get { return _Number; }
  }
  public int TrenerId {
    set { _TrenerId = value; }
    get { return _TrenerId; }
  }
  public string FirstName {
    set { _FirstName = value; }
    get { return _FirstName; }
  }
  public string LastName {
    set { _LastName = value; }
    get { return _LastName; }
  }
  public string FIO {
    set { _FIO = value; }
    get { return _FIO; }
  }
  public string Phone {
    set { _Phone = value; }
    get { return _Phone; }
  }
  public DateTime BirthDate {
    set { _BirthDate = value; }
    get { return _BirthDate; }
  }
  public string Address {
    set { _Address = value; }
    get { return _Address; }
  }
  public string Email {
    set { _Email = value; }
    get { return _Email; }
  }
  public string Message {
    set { _Message = value; }
    get { return _Message; }
  }
}
```

Лістинг 20. Код класу «UsersProvider»

```
using FootballAcademy.AppCode;
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FootballAcademy.Providers {
  class UsersProvider {
    private EncryptData _encryptData = new EncryptData();
    private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings["CONNECTSQL"];

    public void InsertUsers(string FirstName, string LastName, string UsersName, string
UsersPassword,
   int RoleId, string Description) {
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = "INSERT INTO Users (FirstName, LastName, UsersName, UsersPassword,
RoleId, Description) VALUES(@FirstName, @LastName, @UsersName, @UsersPassword, @RoleId,
@Description)";
      MySqlCommand cmd = new MySqlCommand(query, connection);
      cmd.Parameters.AddWithValue("FirstName", FirstName);
      cmd.Parameters.AddWithValue("LastName", LastName);
      cmd.Parameters.AddWithValue("UsersName", UsersName);
      cmd.Parameters.AddWithValue("UsersPassword", _encryptData.Encrypt(UsersPassword));
      cmd.Parameters.AddWithValue("RoleId", RoleId);
      cmd.Parameters.AddWithValue("Description", Description);
      connection.Open();
      cmd.ExecuteNonQuery();
      connection.Close();
      connection.Dispose();
      connection = null;
    }

    public Users SelectedUsersByUsersId(int UsersId) {
      Users selectedUsers = new Users();
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = "SELECT * FROM Users WHERE UsersId=" + UsersId.ToString();
      MySqlCommand command = new MySqlCommand(query, connection);
      connection.Open();
      MySqlDataReader reader = command.ExecuteReader();
      while (reader.Read()) {
        selectedUsers.UsersId = Convert.ToInt32(reader["UsersId"]);
        selectedUsers.FirstName = reader["FirstName"].ToString();
        selectedUsers.LastName = reader["LastName"].ToString();
        selectedUsers.UsersName = reader["UsersName"].ToString();
        selectedUsers.FIO = selectedUsers.LastName + " " + selectedUsers.FirstName;
        selectedUsers.UsersPassword =
_encryptData.Decrypt(reader["UsersPassword"].ToString());
        selectedUsers.RoleId = Convert.ToInt32(reader["RoleId"]);
        selectedUsers.Description = reader["Description"].ToString();
      }
      reader.Close();
      connection.Close();
      return selectedUsers;
    }


    public List<Users> SelectedUsersByUsersNameAndUsersPassword(string UsersName, string
UsersPassword) {
      int i = 0;
      List<Users> UsersList = new List<Users>();
      MySqlConnection connection = new MySqlConnection(_ConnString);
```

```
      string query = "SELECT * FROM Users WHERE UsersName='" + UsersName + "' AND
UsersPassword='" + _encryptData.Encrypt(UsersPassword) + "'";
      MySqlCommand command = new MySqlCommand(query, connection);
      connection.Open();
      MySqlDataReader reader = command.ExecuteReader();
      while (reader.Read()) {
        Users selectedUsers = new Users();
        selectedUsers.Number = ++i;
        selectedUsers.UsersId = Convert.ToInt32(reader["UsersId"]);
        selectedUsers.FirstName = reader["FirstName"].ToString();
        selectedUsers.LastName = reader["LastName"].ToString();
        selectedUsers.FIO = selectedUsers.LastName + " " + selectedUsers.FirstName;
        selectedUsers.UsersName = reader["UsersName"].ToString();
        selectedUsers.UsersPassword =
_encryptData.Decrypt(reader["UsersPassword"].ToString());
        selectedUsers.RoleId = Convert.ToInt32(reader["RoleId"]);
        selectedUsers.RoleName = GetRoleName(selectedUsers.RoleId);
        selectedUsers.Description = reader["Description"].ToString();
        UsersList.Add(selectedUsers);
      }
      reader.Close();
      connection.Close();
      return UsersList;
    }

    public List<Users> GetAllUsers() {
      int i = 0;
      List<Users> UsersList = new List<Users>();
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = "SELECT * FROM Users ORDER BY LastName ASC";
      MySqlCommand command = new MySqlCommand(query, connection);
      connection.Open();
      MySqlDataReader reader = command.ExecuteReader();
      while (reader.Read()) {
        Users selectedUsers = new Users();
        selectedUsers.Number = ++i;
        selectedUsers.UsersId = Convert.ToInt32(reader["UsersId"]);
        selectedUsers.FirstName = reader["FirstName"].ToString();
        selectedUsers.LastName = reader["LastName"].ToString();
        selectedUsers.FIO = selectedUsers.LastName + " " + selectedUsers.FirstName;
        selectedUsers.UsersName = reader["UsersName"].ToString();
        selectedUsers.UsersPassword =
_encryptData.Decrypt(reader["UsersPassword"].ToString());
        selectedUsers.RoleId = Convert.ToInt32(reader["RoleId"]);
        selectedUsers.RoleName = GetRoleName(selectedUsers.RoleId);
        selectedUsers.Description = reader["Description"].ToString();
        UsersList.Add(selectedUsers);
      }
      reader.Close();
      connection.Close();

      if (UsersList.Count == 0) {
        Users noUsers = new Users();
        noUsers.UsersId = 0;
        noUsers.Message = NamesMy.NoDataNames.NoDataInUsers;
        UsersList.Add(noUsers);
      }
      return UsersList;
    }

    private string GetRoleName(int RoleId) {
      RoleApp roleApp = new RoleApp();
      for (int i = 0; i < roleApp.GetRoleList().Count(); i++) {
        if (RoleId == roleApp.GetRoleList()[i].RoleId) {
```

```
          return roleApp.GetRoleList()[i].RoleName;
        }
      }
      return "";
    }

    public void UpdateUsers(string FirstName, string LastName, string UsersName, string
UsersPassword,
   int RoleId, string Description, int UsersId) {
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = "UPDATE Users SET FirstName = @FirstName, LastName = @LastName,
UsersName=@UsersName, UsersPassword=@UsersPassword, RoleId=@RoleId,
Description=@Description" +
          " WHERE UsersId = @UsersId";
      MySqlCommand cmd = new MySqlCommand(query, connection);
      cmd.Parameters.AddWithValue("FirstName", FirstName);
      cmd.Parameters.AddWithValue("LastName", LastName);
      cmd.Parameters.AddWithValue("UsersName", UsersName);
      cmd.Parameters.AddWithValue("UsersPassword", _encryptData.Encrypt(UsersPassword));
      cmd.Parameters.AddWithValue("RoleId", RoleId);
      cmd.Parameters.AddWithValue("Description", Description);
      cmd.Parameters.AddWithValue("UsersId", UsersId);
      connection.Open();
      cmd.ExecuteNonQuery();
      connection.Close();
      connection.Dispose();
      connection = null;
    }

    public void DeleteUsers(int UsersId) {
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = "DELETE FROM Users WHERE UsersId = @UsersId";
      MySqlCommand cmd = new MySqlCommand(query, connection);
      cmd.Parameters.AddWithValue("@UsersId", UsersId);
      connection.Open();
      cmd.ExecuteNonQuery();
      connection.Close();
      connection.Dispose();
      connection = null;
    }

  }
}



public class Users {
  private int _Number;
  private int _UsersId;
  private string _FirstName;
  private string _LastName;
  private string _UsersName;
  private string _FIO;
  private string _UsersPassword;
  private int _RoleId;
  private string _RoleName;
  private string _Description;
  private string _Message;

  public Users() {
    _UsersId = 0;
    _FirstName = String.Empty;
    _LastName = String.Empty;
    _UsersName = String.Empty;
```

```
      _FIO = String.Empty;
      _UsersPassword = String.Empty;
      _RoleId = 0;
      _Description = String.Empty;
    }

    public int Number {
      set { _Number = value; }
      get { return _Number; }
    }
    public int UsersId {
      set { _UsersId = value; }
      get { return _UsersId; }
    }
    public string FirstName {
      set { _FirstName = value; }
      get { return _FirstName; }
    }
    public string LastName {
      set { _LastName = value; }
      get { return _LastName; }
    }
    public string FIO {
      set { _FIO = value; }
      get { return _FIO; }
    }
    public string UsersName {
      set { _UsersName = value; }
      get { return _UsersName; }
    }
    public string UsersPassword {
      set { _UsersPassword = value; }
      get { return _UsersPassword; }
    }
    public int RoleId {
      set { _RoleId = value; }
      get { return _RoleId; }
    }
    public string RoleName {
      set { _RoleName = value; }
      get { return _RoleName; }
    }
    public string Description {
      set { _Description = value; }
      get { return _Description; }
    }
    public string Message {
      set { _Message = value; }
      get { return _Message; }
    }
  }
```