РЕФЕРАТ

Пояснювальна записка: ___ с., ___ рис., ___ табл., __ дод., ___ джерел.

Об'єкт розробки: веб-орієнтована інформаційна система для автоматизації створення турнірних сіток для змагань.

Мета кваліфікаційної роботи: створення веб-орієнтованого програмного додатку для автоматизації створення турнірних сіток для змагань з бойових та інших видів спорту.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проведено аналіз предметної галузі, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, обрано вибір платформи для розробки, виконано проектування і розробка програми, наведено опис алгоритму і структури функціонування програми, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження програми, описана робота програми.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення роботи полягає в можливості завдяки розробленому додатку вести облік учасників змагання, де звичайні користувачі могли б тільки переглядати відомості про учасників та турнірні таблиці і сітки, а тренери змогли б редагувати відомості про учасників і переглядати турнірні таблиці і сітки.

Актуальність теми кваліфікаційної роботи визначається великим попитом на подібні розробки, через те, що розроблене програмне забезпечення дозволить організаторам легко та швидко створювати турнірні сітки та редагувати їх. Організаторам змагань потрібно буде лише мати таблицю з переліком всіх учасників і при завдані деяких параметрів, натиснувши на одну кнопку, програма буде формувати турнірні сітки по відповідним параметрам.

Список ключових слів: ЗМАГАННЯ, ТУРНІР, ІНФОРМАЦІЙНА СИСТЕМА, БАЗА ДАНИХ, ВЕБ-САЙТ, ПРОГРАМНИЙ ДОДАТОК, ПРОЕКТУВАННЯ, ПРОГРАМІВАННЯ.

ABSTRACT

Пояснювальна записка: ___ с., ___ рис., ___ табл., __ дод., ___ джерел.

Об'єкт розробки: веб-орієнтована інформаційна система для автоматизації створення турнірних сіток для змагань.

Мета кваліфікаційної роботи: створення веб-орієнтованого програмного додатку для автоматизації створення турнірних сіток для змагань з бойових та інших видів спорту.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проведено аналіз предметної галузі, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, обрано вибір платформи для розробки, виконано проектування і розробка програми, наведено опис алгоритму і структури функціонування програми, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження програми, описана робота програми.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення роботи полягає в можливості завдяки розробленому додатку вести облік учасників змагання, де звичайні користувачі могли б тільки переглядати відомості про учасників та турнірні таблиці і сітки, а тренери змогли б редагувати відомості про учасників і переглядати турнірні таблиці і сітки.

Актуальність теми кваліфікаційної роботи визначається великим попитом на подібні розробки, через те, що розроблене програмне забезпечення дозволить організаторам легко та швидко створювати турнірні сітки та редагувати їх. Організаторам змагань потрібно буде лише мати таблицю з переліком всіх учасників і при завдані деяких параметрів, натиснувши на одну кнопку, програма буде формувати турнірні сітки по відповідним параметрам.

Список ключових слів: ЗМАГАННЯ, ТУРНІР, ІНФОРМАЦІЙНА СИСТЕМА, БАЗА ДАНИХ, ВЕБ-САЙТ, ПРОГРАМНИЙ ДОДАТОК, ПРОЕКТУВАННЯ, ПРОГРАМІВАННЯ.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ІС – інформаційна система;

ОС – операційна система;

ПЗ – програмне забезпечення;

ПК – персональний комп'ютер;

CTS − Common Type System, звичайна система типів.

ЗМІСТ

ВСТУП

У більшості видів спорту, щоб виявити переможця, використовують турнірні сітки. Це може бути як одна турнірна сітка, так і декілька. Одну турнірну сітку використовують тоді, коли є лише одна загальна категорія, наприклад, у футболі на чемпіонаті світу, де учасники - це збірні команди країн. Декілька турнірних сіток використовують тоді, коли учасники змагань розбиваються на категорії: вагові, вікові і т.д. Наприклад, у змаганнях з карате, де існують вагові та вікові категорії, учасниками яких є бійці.

В залежності від кількості учасників турнірна сітка складається з наступних етапів: фінал (1/2), півфінал (1/4), 1/8, 1/16, 1/32 і т.д. На кожному етапі проводяться поєдинки (якщо йдеться про бойові види спорту) чи матчі (якщо йдеться відповідно про командні види спорту), переможець яких переходить у наступний етап.

У XXI столітті важко уявити міжнародне або національне спортивну подію без відповідного комп'ютерного супроводу. Для великих змагань розробляються замовні системи з вузькою спеціалізацією під конкретний вид спорту.

Для того, щоб сформувати турнірну сітку, організаторам турніру приходиться заповнювати дані про учасників в паперовому вигляді, у кращому випадку використовуючи текстовий редактор або табличний процесор. Наприклад, в MS Excel доводиться вводити імена учасників з клавіатури у вже заготовленні шаблони турнірної сітки – це якщо одна турнірна сітка, а якщо їх багато? Тоді необхідно розбивати всіх учасників на категорії, тобто додавати їх прізвища в окремі таблиці (одна таблиця – одна турнірна сітка). А якщо турнірна сітка вже сформована і потрібно внести певні зміни, додати чи видалити прізвище учасника, що призведе до зміни розташування поєдинків по етапах. Це все займає багато часу.

Але є рішення цієї проблеми, яке дозволить організаторам легко та швидко створювати турнірні сітки та редагувати їх. Організаторам змагань

потрібно буде лише мати таблицю з переліком всіх учасників і при завдані деяких параметрів, натиснувши на одну кнопку, програма буде формувати турнірні сітки по відповідним параметрам. Така б програма вела б облік учасників змагання, де звичайні користувачі могли б тільки переглядати відомості про учасників та турнірні таблиці і сітки, а тренери змогли б редагувати відомості про учасників і переглядати турнірні таблиці і сітки.

Метою даної кваліфікаційної роботи є створення веб-орієнтованого програмного додатку для автоматизації створення турнірних сіток для змагань з бойових та інших видів спорту мовою програмування C#.

Для виконання даної роботи необхідно виконати наступні етапи:

1. Проаналізувати особливості мови програмування C# та середовища програмування Microsoft Visual Studio, платформи ASP.NET MVC5, особливості роботи з СКБД SQLite та MS SQL Server.

2. Описати характеристику середи функціонування програми та її функціональні можливості,

3. Сформулювати вимоги до інформаційної та програмної сумісності, складу і параметрів технічних систем.

4. Описати процес проектування програми та її результати реалізації.

Для реалізації вище зазначених функцій необхідно розділити програмний продукт на дві частини, а саме вебсайт, де здійснюється внесення даних про учасників тренерами та програмний додаток для здійснення формування турнірних таблиць та сіток для спортивних змагань та програмний додаток для адміністратора, де він має змогу завантажити дані про всіх зареєстрованих учасників з вебсайту та створити турнірні сітки, при необхідності роздрукувати їх.

Даний розроблений програмний продукт може бути застосований в організаціях, які організовують змагання з бойових видів спорту, наприклад: карате, тхейквондо, бокс і т.д.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

Спортивний турнір - спортивне змагання з ігрових видів спорту з великим числом учасників (гравців або команд).

Турніри можуть проводитися як короткі заходи в певному місці, або як довгострокова система матчів, що триває протягом тривалого часу (сезон) і проводиться з роз'їздами команд.

Система визначення переможця, жеребкування пар для кожного матчу і кваліфікації в наступних турах регламентуються для кожного турніру правилами змагань.

Кругова система - найпопулярніша для змагань, що проводяться тривалий час, протягом сезону, який може тривати цілий рік і навіть більше. При круговій системі кожна команда або гравець зустрічається з кожною іншою, по як мінімум, раз (одноколових система), часто двічі (двоколова система), зазвичай на своєму і чужому полі (в футболі, хокеї і т. Д.), Або граючи білими і чорними (у шахах і шашках).

При великому числі учасників змагання і обмеженому часу на проведення турнірів використовується олімпійська система, яку називають системою з вибуванням або плей-офф. За такою системою завжди проводяться змагання в любительському боксі, майже завжди в тенісі тощо.

Часто застосовуються змішані системи, коли спочатку організовуються кваліфікаційні групи, змагання в яких проходять за коловою системою, а визначення переможця турніру проводиться за системою з вибуванням.

У системи з вибуванням є модифікація, турнірна система з вибуванням після двох поразок, в якій програла команда не вибуває, а випадає з верхньої сітки в нижню. Крім того, в системі з вибуванням іноді проводять втішні поєдинки, особливо матчі за третє місце, що допомагають визначити, кому

вручати бронзові медалі.

Швейцарська система використовується в разі, коли число учасників занадто велике для проведення кругового змагання, але є можливість провести більше турів, ніж в системі з вибуванням. При швейцарською системою всі учасники турніру грають однакову кількість турів, але в кожному турі зустрічаються гравці або команди, які набрали однакову (або, принаймні, близьке) кількість очок [19].

Система Макмагона схожа на швейцарську, але з тією різницею, що кожному гравцеві вже до початку турніру присвоюється певна кількість очок. Така система використовується в тих випадках, коли в турнірі беруть участь гравці дуже різної сили. Вона дозволяє зводити між собою гравців приблизно однакового рівня. Гравець, який виступає успішно, при такій системі піднімається автоматично в наступну за силою групу після виграшу, отримуючи можливість зіграти з більш сильним противником.

Комп'ютерна система проведення змагань (англ. Computer-Aided Tournament System) - апаратно-програмний інформаційний комплекс для організації спортивних змагань. Програмна частина забезпечує складання турнірних таблиць, планування змагань, введення, обробку і публікацію результатів, автоматизацію інших дій.

Комп'ютерна система актуальна при великому числі учасників в індивідуальних видах спорту, в яких кращий виявляється в результаті серії поєдинків один на один або двоє на двоє (парний розряд). Її застосування починається зі стадії підготовки і прийому заявок. Вона дозволяє краще відстежувати хід змагань, оперативно виводити інформаційні матеріали на стенди і в Інтернет для учасників і глядачів.

У XXI столітті важко уявити міжнародне або національне спортивну подію без відповідного комп'ютерного супроводу. Для великих змагань розробляються замовні системи з вузькою спеціалізацією під конкретний вид спорту.

Регіональні федерації, місцеві об'єднання і окремі спортклуби теж

проводять турніри: масові змагання серед любителів, дітей та юнаків. Кількість лотів в них у багато разів перевищує число дорослих спортсменів професіоналів. Такі організації також мають потребу в програмному забезпеченні змагань і класифікації спортсменів. Їм потрібні доступні системи, що автоматизують рутинні процедури, що полегшують координацію і ведення протоколів[16].

Крім програм існують також онлайн сервіси для проведення змагань, в яких, як правило, гравці самі реєструються на змагання, а потім повідомляють про результати матчу.

1.2. Призначення розробки та галузь застосування

Даний розроблений програмний продукт може бути застосований в організаціях, які організовують змагання з бойових видів спорту, наприклад: карате, тхейквондо, бокс і т.д.

Завдяки розробленому додатку можливо вести облік учасників змагання, де звичайні користувачі могли б тільки переглядати відомості про учасників та турнірні таблиці і сітки, а тренери змогли б редагувати відомості про учасників і переглядати турнірні таблиці і сітки.

Розроблене програмне забезпечення дозволить організаторам легко та швидко створювати турнірні сітки та редагувати їх. Організаторам змагань потрібно буде лише мати таблицю з переліком всіх учасників і при завдані деяких параметрів, натиснувши на одну кнопку, програма буде формувати турнірні сітки по відповідним параметрам.

1.3. Підстава для розробки

Підставою для розробки кваліфікаційної роботи бакалавра на тему «Розробка веб-орієнтованого програмного додатку для автоматизації створення турнірних сіток для змагань засобами мови C#» є наказ по Національному

технічному університету «Дніпровська політехніка» від __.__. 2021р. № ____-_.

1.4. Постановка завдання

Метою даної кваліфікаційної роботи є створення веб-орієнтованого програмного додатку для автоматизації створення турнірних сіток для змагань з бойових та інших видів спорту мовою програмування C#.

Завданням даної роботи є спроектувати та розробити інформаційну систему з багаторівневим доступом для автоматизованого створення турнірних таблиць та сіток для спортивних змагань.

Для виконання даної роботи необхідно виконати наступні етапи:

1. Проаналізувати особливості мови програмування C# та середовища програмування Microsoft Visual Studio, платформи ASP.NET MVC5, особливості роботи з СКБД SQLite та MS SQL Server.
2. Описати характеристику середи функціонування програми та її функціональні можливості,
3. Сформулювати вимоги до інформаційної та програмної сумісності, складу і параметрів технічних систем.
4. Описати процес проектування програми та її результати реалізації.

Інтерфейс додатка повинен бути графічним, багатовіконним. Інформаційна система повинна зберігати інформацію про учасників змагань та користувачів системи. А також надавати можливість:

– не авторизованим користувачам переглядати відомості про учасників змагань, турнірні таблиці та турнірні сітки;

– тренерам додавати, редагувати дані про учасників змагань та переглядати турнірні таблиці та сітки;

– адміністратору формувати та виводити на друк турнірні таблиці та турнірні сітки на підставі даних внесених тренерами про учасників змагань.

Розроблюваний програмний продукт, повинен складається з двох частин,

а саме з веб-сайту для формування переліку учасників змагань та програмного додатку (для адміністратора), призначений для того, щоб організатори турніру з бойових та інших видів спорту мали змогу легко та швидко сформувати турнірні сітки.

Вхідними даними системи будуть виступати особисті дані про учасників змагань, які будуть додаватися до системи тренерами (авторизованими користувачами) на веб-сайті та дані для їхньої авторизації (логін та пароль).

В якості вихідних даних будуть виступати сформовані турнірні таблиці зареєстрованих на змагання учасників, сформовані турнірні таблиці та сітки по кожній категорії, які будуть розташовані на сайті, побудовані турнірні сітки змагань.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Для виконання кваліфікаційної роботи необхідно спроектувати та розробити інформаційну систему для автоматизації формування турнірних таблиць та сіток для змагань з бойових видів спорту.

Під час реалізації даної роботи необхідно дотримуватися наступних вимог до функціоналу інформаційної системи:

1. В програмному продукті необхідно передбачити ведення обліку учасників змагань та багаторівневий доступ.

2. Звичайні користувачі системи повинні мати можливість тільки переглядати відомості про учасників змагань та сформовані турнірні таблиці і сітки на сайті.

3. Тренери повинні мати можливість вносити та редагувати відомості про учасників і переглядати сформовані турнірні сітки на сайті.

4. Адміністратор повинен мати можливість переглядати та видаляти відомості про учасників та формувати з вхідних даних турнірні таблиці і сітки, а також при необхідності виводити їх на друк.

5. При завантаженні веб-сайту повинна з’явитися його головна сторінка, де буде відображатися перелік усіх зареєстрованих учасників. Головна сторінка повинна містити елементи керування та навігації. Після входу в акаунт, користувачу (тренеру) повинна бути доступна функція для додавання, редагування та видалення даних про учасників.

6. Для того, щоб мати можливість сформувати турнірні таблиці та додати турнірні сітки потрібно бути авторизованим користувачем на сайті під спеціальним логіном та паролем адміністратора.

7. При запуску програмного додатка, який безпосередньо формує турнірні сітки, повинна бути передбачена функція авторизації адміністратора. Адміністратор повинен мати можливість завантажити інформацію про учасників з сайту, розбити загальний перелік учасників на таблиці по категоріям, сформувати турнірні сітки по відповідним категоріям виходячи із сформованих таблиць. Також він повинен мати змогу переглянути у програмі створені турнірні сітки та при необхідності роздрукувати їх.

1.5.2. Вимоги до інформаційної безпеки

Проблема захисту інформації є багатоплановою, комплексною і охоплює ряд важливих завдань.

Можна виділити вимоги до інформаційної безпеки:

- цілісність даних;
- захист від неавторизованого редагування;
- конфіденційність інформації;
- доступність інформації для всіх авторизованих користувачів.

Для надійної роботи додатку необхідно:

- використовувати ліцензійне програмне забезпечення на сервері;
- здійснювати захист від вірусів на сервері;
- здійснювати захист від несанкціонованого доступу;

— застосовувати на сервері джерело безперебійного живлення для захисту від перепадів напруги або збоїв у живленні;

— здійснювати контроль даних, що вводяться користувачем.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для стабільної роботи додатка на персональному комп'ютері повинна бути встановлена ОС Windows 7 і вище та мати наступні мінімальні характеристики комп'ютера:

– процесор: 1 ГГц;
– кількість ядер: 2;
– оперативна пам'ять: 2 Гб;
– відеокарта: 256 Мб; тип пам'яті DDR2;
– об'єм пам'яті жорсткого диску: 512Гб.

Рекомендовані наступні характеристики комп'ютера:

– процесор: 2,4ГГц;
– кількість ядер: 2;
– оперативна пам'ять: 8Гб;
– відеокарта: 2Гб; тип пам'яті DDR3;
– об'єм пам'яті жорсткого диску: 1024Гб.

1.5.4. Вимоги до інформаційної та програмної сумісності

Для повноцінної роботи програмного продукту необхідно наступне програмне забезпечення:

– будь-який браузер для завантаження сайту для реєстрації учасників на змагання;

– табличний процесор MS Excel для завантаження даних в таблицю бази даних.

Для забезпечення сталого функціонування ІС потрібна наявність наступних системних установок та ліцензованого програмного забезпечення:

- заінстальоване середовище розробки Visual Studio 2017;
- ОС Windows 7 з архітектурою 64 розряди (x64);

Підтримувані операційні системи:

- Windows XP x86 SP 3 (усі редакції, крім Starter Edition);
- Windows Vista x86 або x64 SP 2 (усі редакції, крім Starter Edition);
- Windows 7 x86 або x64.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Завданням даної роботи є спроектувати та розробити інформаційну систему з багаторівневим доступом для автоматизованого створення турнірних таблиць та сіток для спортивних змагань.

Розроблюваний програмний продукт складається з двох частин, а саме з веб-сайту для формування переліку учасників змагань та програмного додатку (для адміністратора), призначений для того, щоб організатори турніру з бойових та інших видів спорту мали змогу легко та швидко сформувати турнірні сітки.

Інформаційна система зберігає інформацію про учасників змагань та користувачів системи та надає можливість:

– не авторизованим користувачам переглядати відомості про учасників змагань, турнірні таблиці та турнірні сітки;

– тренерам додавати, редагувати дані про учасників змагань та переглядати турнірні таблиці та сітки;

– адміністратору формувати та виводити на друк турнірні таблиці та турнірні сітки на підставі даних внесених тренерами про учасників змагань.

Розроблене програмне забезпечення дозволяє організаторам легко та швидко створювати турнірні сітки та редагувати їх. Організаторам змагань потрібно буде лише мати таблицю з переліком всіх учасників і при завдані деяких параметрів, натиснувши на одну кнопку, програма буде формувати турнірні сітки по відповідним параметрам.

2.2. Опис застосованих математичних методів

Під час проектування даного програмного продукту ніяких математичних методів моделювання ПЗ не використовувалося.

2.3. Опис використаних технологій та мов програмування

Програма написана за допомогою методів об’єктного-подібного програмування на мові C# та скомпільована з розширенням exe. Вона може працювати у операційних системах Windows та UNIX-подібних ОС. В останній операційній системі exe-файл можна запустити за допомогою безкоштовного емулятора під назвою WINE.

На сьогодні найпопулярніші операційні системи (далі - ОС) є Windows 7 та Windows 10, якщо говорити про ОС Windows.

З кожним днем користувачі персональних комп’ютерів переходять на більш нову ОС Windows 10, якщо дозволяють характеристики. Windows 10 має низку переваг над Windows 7.

Дизайн Windows 10 однозначно виглядає більш сучасним, в оформленні ОС використовуються яскраві насичені кольори, до того ж можливість використання декількох віртуальних робочих столів істотно спрощує роботу.

Нововведенням, яке зацікавить геймерів, є можливість роботи з DirectX 12, що збільшує продуктивність комп'ютера і покращує оптимізацію. Ще однією відмінною рисою стала функція Xbox Live, що дозволяє грати в багатокористувацькому режимі в ігри як з комп’ютером, так і з Xbox відповідно.

Windows 10 в цьому випадку впевнено обходить Windows 7, оскільки десята версія заточена під роботу з безліччю сенсорних пристроїв. Користуватися Windows 10 з планшета набагато зручніше, при цьому не потрібно стилусів та інших предметів, сама навігація реалізована на найвищому рівні.

При виборі двох систем Windows 10 і Windows 7, було зупинено свій вибір на «десятці». Вона випереджає «сімку» по багатьом тестам по продуктивності. Також має багато нових сучасних технологій. Розробники Microsoft використовували всі переваги Windows 7 і Windows 8 і використовували їх в новій десятої версії. Єдиним недоліком вибору Windows 10 є не підтримка драйверів для старих комплектуючих і не підтримка старих програм.

Якщо на комп'ютер встановлена 64-х розрядна операційна система Windows 10, то додаток буде працювати без проблем [23].

Розроблювана інформаційна система складається з двох частин, а саме веб-сайту для формування переліку учасників змагань та додатку, який формує турнірні таблиці та сітки для спортивних змагань.

Для створення програмного додатку було обране середовище розробки Microsoft Visual Studio 2017 та мова програмування C#.

Microsoft Visual Studio - це вбудоване середовище розробки для створення, документування, запуску і налагодження програм, написаних на мовах .NET. Дане середовище розробки вважається відкритим мовним середовищем. Важливою умовою для підключення мов в середовище VisualStudio.Net вважається впровадження єдиного каркасу - платформи Framework.Net.

Платформа Framework. Net дозволяє:

– просто застосувати складові, створені на всіляких мовах;

– розробляти цілий додаток з декількох частин на різних мовах.

Платформа Framework.Net має дві провідні складові:

– Framework Class Library (далі - FCL) – бібліотеку класів каркасу;

– Common Language Runtime (далі - CLR) – загальномовне виконавче середовище.

В рамках наданої платформи застосовується звичайна система типів Common Type System (далі - CTS), яка цілком описує всі типи даних, підтримувані середовищем виконання, визначає взаємодію типів даних та їх

представлення у форматі метаданих .NET.

Комплект правил, що визначають підмножину сукупних типів даних, у відношенні яких гарантується, що вони небезпечні при застосуванні у всіх мовах .NET, описується в рамках специфікації Common Language Specification (далі - CLS). Для того щоб класи, створені на різних мовах, можливо було разом застосувати в рамках одного додатку, вони повинні задовольняти конкретні обмеження, які задаються CLS. Клас, що задовольняє CLS, іменується CLS-сумісним. Він доступний для застосування в інших мовах, класи яких можуть бути клієнтами або ж спадкоємцями сумісного класу.

Платформа .NET дає у розпорядження розробника програмного забезпечення бібліотеку базисних класів, доступну з будь-якої мови програмування .NET. Тому що число класів бібліотек FCL досягає декількох тисяч, тому в цілях структуризації функціонально найближчі класи з'єднуються в групи, іменовані простором імен (Namespace).

Головним простором імен бібліотеки FCL вважається простір System, що містить як класи, так і інші вкладені простори імен. Наприклад, в просторі System.Collections присутні класи і інтерфейси, що підтримують роботу з колекціями об'єктів - переліками, чергами, словниками. Простір System.Windows.Forms має класи, які застосовуються при розробці windows-додатків.

C# генерує код, призначений для виконання лише тільки в середовищі виконання .NET (керований код). Сам бінарний файл, який має керований файл, називається збіркою. Збірка має код на проміжній мові MSIL (Microsoft Intermediate Language) або ж елементарно IL. Подібно байт-коду Java IL-код компілюється в переносні специфічні функції пам'яті при конкретному запиті середовища виконання .NET до блоку IL-інструкцій. Двійкові модулі .NET на додаток містять ще метадані. Метадані описують не тільки типи, що застосовуються в збірці, а й саму збірку. Така частка метаданих називається маніфестом.

У більшості випадків між двійковим файлом .NET і збіркою є ві-

дношення "один-до-одного". Втім збірка може складатися як з одного, так і з кількох двійкових файлів.

Збірка з першого файлу має і маніфест, і метадані, і інструкції IL. Двійкові файли, що утворюють разом спільну збірку іменуються модулями. При цьому один з двійкових файлів зобов'язаний містити маніфест збірки. Інші модулі можуть тримати лише тільки метадані типів і інструкції IL.

Багатофайлові збірки дають можливість середовищу виконання вибірково завантажувати лише тільки ті збірки, які в цей момент роботи програми справді необхідні, що дозволяє зменшувати мережевий трафік і нарощування швидкості роботи програми [12].

Для того щоб перейти до створення програми, потрібно створити проект графічного додатка та обрати тип проекту Windows Forms Application.

Windows Forms дозволяє розробляти інтелектуальні клієнти. Інтелектуальний клієнт - це додаток з повнофункціональним графічним інтерфейсом, нескладне в розгортанні і оновленні, здатне працювати при наявності або недоступності включення до інтернету і застосовує більше безпечний доступ до ресурсів на локальному ПК в порівнянні з класичними додатками Windows.

Windows Forms - це розробка інтелектуальних клієнтів для платформи .NET Framework, комплект керованих бібліотек, що спрощують виконання популярних завдань, таких як читання і запис в файловій системі. При застосуванні середовища розробки, такого як Visual Studio, можливо створювати додатки інтелектуальних клієнтів Windows Forms, які відображають інформацію, здійснюють запит на введення від користувачів і взаємодіють з віддаленими комп'ютерами по мережі.

У Windows Forms конфігурація - це зорова площина, на якій виводиться інформація для користувача. Як правило додаток Windows Forms ґрунтується методом розміщення складових управління на форму і написання коду для реагування на дії користувача, такі як клацання миші або ж натискання кнопок. Елемент керування - це окремий елемент користувацького інтерфейсу,

призначений для відображення або ж введення даних.

При виконанні користувачем будь-якої дії з формою або ж одним з її складових управління формується подія. Додаток відгукується на ці дії з підтримкою коду і обробляє дії при їх появі.

Windows Forms включає великий комплект складових управління, які можна додавати на форми: текстові поля, кнопки, розгортуючі списки, перемикачі та в тому числі і інтернет-сторінки. Перелік всіх складових управління, які можливо застосувати на формі, представлені в розділі Елементи управління для застосування в формах Windows Forms. У разі, якщо існуючий елемент керування не задовольняє потребам, в Windows Forms можливо зробити користувацькі елементи управління з підтримкою класу UserControl.

До складу Windows Forms входять функціональні складові призначені для користувацького інтерфейсу, що дозволяють відтворювати можливості складних додатків, як Microsoft Office. Застосовуючи складові управління ToolStrip і MenuStrip, можливо створювати панелі інструментів і меню, що мають текст і картинки, підменю і інші складові управління, такі як текстові поля і поля зі списками.

З підтримкою перетягування і вставки конструктор Windows Forms в Visual Studio дозволяє легко створювати додатки Windows Forms. Досить виділити елемент управління курсором і помістити його в необхідний простір на формі. Для подолання проблем, пов'язаних з вирівнюванням складових управління, конструктор дає такі засоби, як лінії сітки і лінії прив'язки. З підтримкою Visual Studio або ж компіляції з командного рядка, можливо застосувати FlowLayoutPanel,

TableLayoutPanel і SplitContainer складові управління для створення складних макетів форм за короткий час.

Зрештою, в разі якщо треба зробити власні особисті елементи призначеного для користувацького інтерфейсу, простір імен System.Drawing має великий комплект вміщень, важливих для відтворення рядків, кіл та інших фігур саме на формі [21].

Для програмування всіх подій на формі була обрана мова програмування C#. Вона, серед більшості мов програмування, займає почесне місце. C# унаслідувала практично усе найкраще від своїх попередників. Ця мова набуває все більшої популярності. Її освоюють не лише початківці, а й ті програмісти, що мають навички використання інших мов програмування. Ця мова програмування повністю підтримує три основних принципи, що лежать в основі об'єктно-орієнтованого програмування: інкапсуляцію, поліморфізм та успадкування – і широко використовується для навчання об'єктно-орієнтованого програмування.

Поряд із цим, мова програмування C# має цілий ряд інших переваг, що роблять її зручною у використанні для навчання програмування:

– мова програмування C# - мультипарадигмальна, окрім об'єкт-но-орієнтованої вона підтримує імперативну та функціональну парадигми;

– мова C# надає можливість створювати програми як для консольного, так і для віконного виконання;

– «C-подібний» синтаксис мови C#, як найбільш поширений, дозволяє фахівцю відносно легко перейти до використання багатьох інших мов програмування;

– існує велика кількість доступних для використання візуальних середовищ програмування мовою C#, (Microsoft Visual Studio Professional, Microsoft Visual C# Express Edition, Sharp Develop, Mono Develop, Borland C# Editor);

– програми, написані мовою C#, можуть виконуватися під управлінням багатьох операційних систем, що підтримують програмні платформи Microsoft .Net Framework чи Mono;

– фірма Microsoft (розробник мови програмування C# та середовища програмування Visual Studio Professional), а також інші розробники середовищ програмування мовою C# надають потужну і різносторонню підтримку своїх впроваджень [10].

C# розроблялась як мова програмування прикладного рівня для CLR і

тому вона залежить, перш за все, від можливостей самої CLR. Це стосується, перш за все, системи типів C#. Присутність або відсутність тих або інших виразних особливостей мови диктується тим, чи може конкретна мовна особливість бути трансльована у відповідні конструкції CLR. Так, з розвитком CLR від версії 1.1 до 2.0 знач-но збагатилася і сама C#. CLR надає C#, як і всім іншим .NET-орієнтованим мовам, багато можливостей, яких позбавлені «класичні» мови програмування.

C# підтримує старі типізовані неявні оголошення змінних з ключовим словом var і неявно типізовані масиви з ключовим словом new[], за яким слідує ініціалізатор колекції.

C# підтримує суворий тип даних Boolean, bool. Вирази, які приймають умови, такі як while та if, вимагають умови, що реалізує оператор true або false. Хоча C++ також має тип Boolean, він може бути вільно перетворений в цілі числа та з них, а вирази, такі як if(a), вимагають тільки того, щоб a був конвертований в bool, що дозволяє бу-ти a int-типу або вказівником.

C# безпечніший в порівнянні з C++. Єдиними неявними перетвореннями за замовчуванням є ті, які вважаються безпечними, наприклад, розширення цілих чисел. Це застосовується під час компіляції, під час JIT і, в деяких випадках, під час виконання. Не відбувається неявних перетворень між булевими і цілими числами, а також між членами перерахування і цілими числами (крім літерала 0, який може бути неявно перетворений в будь-який нумерований тип). Будь-яке призначене для користувача перетворення повинно бути явно позначене як явне або неявне, на відміну від конструкторів копіювання C++ і операторів перетворення, які за умовчанням є неявними.

Мова C# не допускає глобальних змінних або функцій. Всі методи і члени повинні бути оголошені всередині класів. Статичні члени відкритих класів можуть замінювати глобальні змінні та функції.

Методи в мові програмування є членами класу в проекті, деякі методи мають підписи, а деякі не мають підпису. Методи можуть бути недійсними або можуть повертати щось на зразок рядка, цілого, подвійного, десяткового, float і

bool. Якщо метод недійсний, це означає, що метод не повертає жодного типу даних.

Подібно C++, і на відміну від Java, програмісти на C# повинні використовувати ключове слово virtual, щоб дозволити перевизначати методи підкласами.

Методи розширення в C# дозволяють програмістам використовувати статичні методи, як якщо б вони були методами з таблиці методів класу, дозволяючи програмістам додавати методи до об'єкта, який, на їхню думку, повинен існувати на цьому об'єкті і його похідних [27].

Для зберігання даних про учасників та адміністаратора в додатку використовується база даних, яка створена за допомогою системи управління базами даних (далі – СУБД) SQLite, яка має ряд наступних переваг:

– не вимагає окремого процесу сервера або системи для роботи (без сервера);

– поставляється з нульовою конфігурацією, що означає відсутність необхідності в налаштуванні або адмініструванні;

– повна база даних SQLite зберігається в одному кросплатформеному диску;

– є автономною, що означає відсутність зовнішніх залежностей;

– SQLite-транзакції повністю сумісні з ACID, забезпечуючи безпечний доступ до декількох процесів або потоків;

– підтримує більшість функцій мови запитів;

– SQLite написана на ANSI-C і забезпечує легкий і простий у використанні API [8].

Щоб здійснювати зв'язок між базою даних і додатком на C# необхідний посередник. І саме таким посередником є технологія ADO.NET.

ADO.NET являє собою технологію роботи з даними, яка заснована на платформі .NET Framework. Ця технологія має набір класів, через які є можливість відправляти запити до баз даних, встановлювати підключення, отримувати відповідь від бази даних і виробляти ряд інших операцій.

Причому важливо зазначити, що систем управління баз даних може бути безліч. У своїй сутності вони можуть відрізнятися. MS SQL Server, наприклад, для створення запитів використовує мову T-SQL. Різні системи баз даних можуть мати різні типи даних. Також можуть відрізнятися якісь інші моменти. Однак функціонал ADO.NET побудований таким чином, щоб надати розробникам уніфікований інтерфейс для роботи з самими різними СУБД.

Основу інтерфейсу взаємодії з базами даних в ADO.NET представляє обмежене коло об'єктів: Connection, Command, DataReader, DataSet і DataAdapter. За допомогою об'єкта Connection відбувається установка підключення до джерела даних. Об'єкт Command дозволяє виконувати операції з даними в базі даних (далі – БД). Об'єкт DataReader зчитує отримані в результаті запиту дані. Об'єкт DataSet призначений для зберігання даних в БД і дозволяє працювати з ними незалежно від БД. І об'єкт DataAdapter є посередником між DataSet і джерелом даних. Головним чином, через ці об'єкти і буде йти робота з базою даних.

Однак щоб використовувати один і той же набір об'єктів для різних джерел даних, необхідний відповідний провайдер даних. Власне через провайдер даних в ADO.NET і здійснюється взаємодія з базою даних. Причому для кожного джерела даних в ADO.NET може бути свій провайдер, який власне і визначає конкретну реалізацію вищевказаних класів.

Функціонально класи ADO.NET можна розбити на два рівня: підключений і відключений. Кожен провайдер даних .NET реалізує свої версії об'єктів Connection, Command, DataReader, DataAdapter і ряду інших, який складають підключений рівень. Тобто за допомогою них встановлюється підключення до БД і виконується з нею взаємодія. Як правило, реалізації цих об'єктів для кожного конкретного провайдера в своїй назві мають префікс, який вказує на провайдер: SqlConection, SqlCommad, SqlDataReader, SqlDataAdapter.

Інші класи, такі як DataSet, DataTable, DataRow, DataColumn і ряд інших складають відключений рівень, так як після отримання даних в DataSet ми можемо працювати з цими даними незалежно від того, чи встановлено

підключення чи ні. Тобто після отримання даних з БД додаток може бути відключено від джерела даних [9].

Сформовані додатком турнірні таблиці будуть зберігатися у зовнішньому файлі.

Файл - це набір даних, який зберігається на зовнішньому пристрої пам'яті (наприклад на жорсткому диску). Файл має ім'я і розширення. Розширення дозволяє ідентифікувати, які дані і в якому форматі зберігаються у файлі.

Під роботою з файлами мається на увазі:

– створення файлів; видалення файлів; читання даних; запис даних;
– зміна параметрів файлу (ім'я, розширення).

У C# є простір імен System.IO, в якому реалізовані всі необхідні нам класи для роботи з файлами. Щоб підключити цей простір імен, необхідно на самому початку програми додати рядок using System.IO. Сумісно можна використовувати простір імен System.Text, що допоможе використовувати різні кодування тексту.

Для створення порожнього файлу, в класі File є метод Create(). Він приймає один аргумент - шлях. Нижче наведений приклад створення порожнього текстового файлу new_file.txt на диску D. Якщо файл з таким ім'ям вже існує, він буде переписаний на новий порожній файл.

Метод WriteAllText() створює новий файл (якщо такого немає), або відкриває існуючий і записує текст, замінюючи все, що було у файлі.

Метод Delete() видаляє файл за вказаним шляхом.

Крім того, щоб читати/записувати дані у файл у C# можна використовувати потоки.

Потік - це абстрактне уявлення даних (у байтах), яке полегшує роботу з ними. В якості джерела даних може бути файл, пристрій введення-виведення, принтер.

Клас Stream є абстрактним базовим класом для всіх потокових класів у C#. Для роботи з файлами використовується клас FileStream (файловий потік). FileStream - представляє потік, який дозволяє виконувати операції

читання/запису у файл.

Режими відкриття FileMode:

– Append - відкриває файл (якщо той існує) і переводить покажчик у кінець файлу (дані будуть дописуватися до кінця), або створює новий файл. Даний режим можливий тільки при режимі доступу FileAccess.Write;

– Create - створює новий файл (якщо той існує - замінює його);

– CreateNew - створює новий файл (якщо той існує - генеруються "винятки");

– Open - відкриває файл (якщо той не існує - генеруються "винятки");

– OpenOrCreate - відкриває файл, або створює новий, якщо його не існує;

– Truncate - відкриває файл, але всі дані всередині файлу затирає (якщо файлу не існує - генеруються "винятки").

Режими доступу FileAccess:

– Read - відкриття файлу тільки на читання, при спробі запису генерується виключення;

– Write - відкриття файлу тільки на запис, при спробі читання генерується виключення;

– ReadWrite - відкриття файлу на читання і запис.

Для читання даних з потоку використовується клас StreamReader. У ньому реалізовано безліч методів для зручного зчитування даних.

Метод ReadToEnd() зчитує всі дані з файлу. ReadLine() - зчитує один рядок.

Властивість EndOfStream вказує, чи знаходиться поточна позиція в потоці в кінці потоку (чи досягнуто кінець файлу). Повертає true або false.

Для запису даних у потік використовується клас StreamWriter. Метод WriteLine() записує у файл порядково (те ж саме, що і простий запис за допомогою Write(), тільки в кінці додається новий рядок).

Потрібно завжди пам'ятати, що після роботи з потоком, його треба закрити (звільнити ресурси), використавши метод Close() [25].

Для розробки веб-сайту була обрана платформа ASP.NET MVC, яка являє собою фреймворк для створення сайтів і веб-додатків за допомогою реалізації паттерна MVC.

Концепція паттерна (шаблону) MVC (model - view - controller) передбачає поділ додатка на три компоненти.

Контролер (controller) представляє клас, що забезпечує зв'язок між користувачем і системою, поданням і сховищем даних. Він отримує введені користувачем дані і обробляє їх. І в залежності від результатів обробки відправляє користувачеві певний висновок, наприклад, у вигляді представлення.

Представлення (view) - це власне візуальна частина або призначений для користувача інтерфейс програми. Як правило, html-сторінка, яку користувач бачить, зайшовши на сайт.

Модель (model) представляє клас, що описує логіку використовуваних даних.

Загальну схему взаємодії цих компонентів можна представити наступним чином (рис. 2.1).

У цій схемі модель є незалежним компонентом - будь-які зміни контролера або подання не зачіпають модель. Контролер і представлення є відносно незалежними компонентами, і нерідко їх можна змінювати незалежно один від одного.

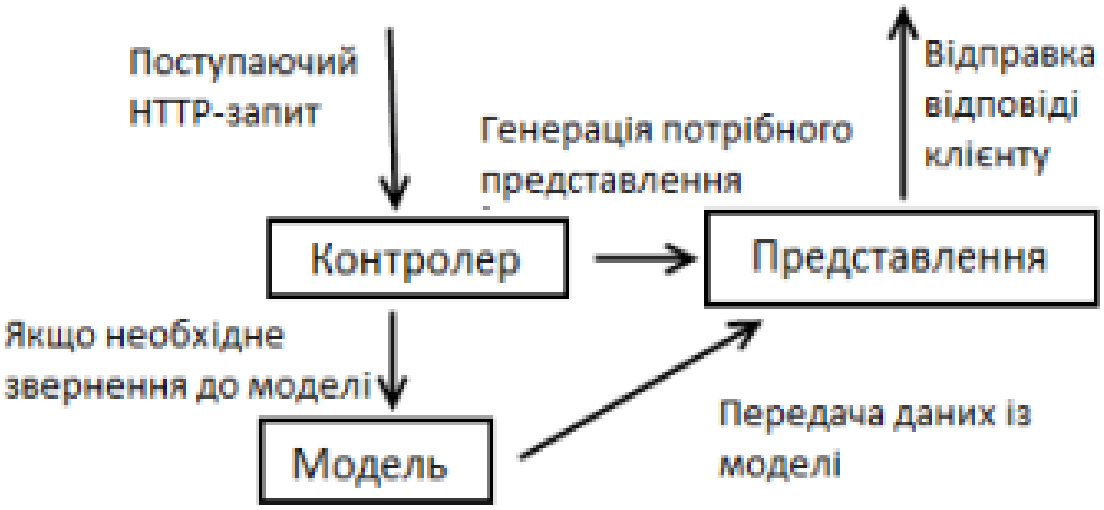

Рис. 2.1. Схема взаємодії компонентів ASP.NET MVC

Завдяки цьому реалізується концепція поділ відповідальності, в зв'язку з чим легше побудувати роботу над окремими компонентами. Крім того, внаслідок цього додаток має кращу тестованість. Тобто є можливість тестувати окреме представлення незалежно від контролера. Або є можливість зосередитися на бекенді (програмно-апаратна частина сервісу) і тестувати контролер.

Конкретні реалізації та визначення даного патерну можуть відрізнятися, але в силу своєї гнучкості і простоти він став дуже популярним останнім часом, особливо в сфері веб-розробки.

Свою реалізацію паттерна представляє платформа ASP.NET MVC. 2013 рік ознаменувався виходом нової версії ASP.NET MVC - MVC 5, а також релізом Visual Studio 2013, яка надає інструментарій для роботи з MVC5.

Хоча в багатьох аспектах MVC 5 не дуже сильно відрізнятиметься від MVC 4, багато з однієї версії цілком можна застосувати до іншої, але в той же час є і суттєві відмінності:

– у MVC 5 змінилася концепція аутентифікації і авторизації. Замість SimpleMembershipProvider була впроваджена система ASP.NET Identity, яка використовує компоненти OWIN і Katana;

– для створення адаптивного і розширюваного інтерфейсу в MVC 5

використовується css-фреймворк Bootstrap;

– додані фільтри аутентифікації, а також з'явилася функціональність перевизначення фільтрів;

– у MVC 5 також додані атрибути маршрутизації.

Це найбільш важливі нововведення в MVC 5. Крім того, є ще ряд менш значимих, наприклад, використання за замовчуванням Entity Framework 6, деякі зміни при створенні проекту (концепція One ASP.NET), додаткові компоненти і т.д.

У будь-якому випадку всі отримані при роботі з MVC 4 навички можна успішно застосовувати при використанні MVC 5, враховуючи, звичайно, нововведення [17].

Збереження даних, які вводяться до системи засобами веб-сайту (дані про учасників, тренерів та адміністратора) здійснюється з використанням бази даних, яка реалізована за допомогою СУБД MS SQL Server.

SQL Server є однією з найбільш популярних систем управління базами даних в світі. Дана СУБД підходить для самих різних проектів: від невеликих додатків до великих високонавантажених проектів.

Розглянемо особливості SQL Server:

– продуктивність, SQL Server працює дуже швидко;

– надійність і безпека, SQL Server надає шифрування даних;

– простота, з даної СУБД відносно легко працювати і вести адміністрування.

Центральним аспектом в MS SQL Server, як і в будь-якій СУБД, є база даних. База даних являє собою сховище даних, організованих певним способом. Нерідко фізично база даних представляє файл на жорсткому диску, хоча така відповідність необов'язкова. Для зберігання і адміністрування баз даних застосовуються системи управління базами даних (database management system) або СУБД (DBMS). І якраз MS SQL Server є однією з такою СУБД.

Для організації баз даних MS SQL Server використовує реляційну модель. Ця модель баз даних була розроблена ще в 1970 році Едгаром Коддом. А на

сьогоднішній день вона фактично є стандартом для організації баз даних.

Реляційна модель передбачає зберігання даних у вигляді таблиць, кожна з яких складається з рядків і стовпців. Кожен рядок зберігає окремий об'єкт, а в стовпчиках розміщуються атрибути цього об'єкта.

Для ідентифікації кожного рядка в рамках таблиці застосовується первинний ключ (primary key). В якості первинного ключа може виступати один або декілька стовпців. Використовуючи первинний ключ, ми можемо посилатися на певний рядок в таблиці. Відповідно два рядки не можуть мати один і той же первинний ключ.

Через ключі одна таблиця може бути пов'язана з іншого, тобто між двома таблицями можуть бути організовані зв'язки. А сама таблиця може бути представлена у вигляді відношення ("relation").

Для взаємодії з базою даних застосовується мова SQL (Structured Query Language). Клієнт (наприклад, зовнішня програма) відправляє запит на мові SQL за допомогою спеціального API. СУБД належним чином інтерпретує і виконує запит, а потім посилає клієнту результат виконання [11].

Переваги мови SQL:

– незалежність від конкретної СУБД – незважаючи на наявність діалектів і відмінностей в синтаксисі, в більшості своїй тексти SQL-запитів, що містять, DDL і DML, можуть бути досить легко перенесені з однієї СУБД в іншу; наявність стандартів – наявність стандартів і набору тестів для виявлення сумісності і відповідності конкретній реалізації SQL загальноприйнятому стандарту тільки сприяє «стабілізації» мови;

– декларативність – за допомогою SQL програміст описує тільки те, які дані потрібно витягнути, або модифікувати а те, яким чином це зробити, вирішує СУБД безпосередньо при обробці SQL-запиту.

Поряд з перевагами мова SQL має ряд недоліків:

– складність – хоча SQL і замислювалася, як засіб роботи кінцевого користувача, врешті-решт вона стала настільки складною, що перетворилася на інструмент програміста;

– відступи від стандартів – незважаючи на наявність міжнародного стандарту ANSI SQL-92, багато компаній, СУБД (наприклад, Oracle, Sybase, Microsoft, MySQL), що займаються розробкою, вносять зміни до мови SQL, вживаної в розроблених ними СУБД, тим самим відступаючи від стандарту. Таким чином з'являються специфічні для кожної конкретної СУБД діалекти мови SQL [22].

Для повноцінної роботи програмного продукту використовується табличний процесор MS Excel для завантаження даних в таблицю бази даних.

MS Excel - це широко поширена комп'ютерна програма для проведення розрахунків, складання таблиць і діаграм, обчислення простих і складних функцій. Вона входить до складу пакета Microsoft Office.

Вона являє собою велику таблицю, в яку можна вносити дані, тобто друкувати слова і цифри. Також, використовуючи функції цієї програми, можна виробляти з цифрами різні маніпуляції: складати, віднімати, множити, ділити і багато іншого.

Популярність та ефективність табличного процесора MS Excel визначається його надзвичайно широкими можливостями, основними з яких вважаються:

– створення таблиць даних, між комірками яких можуть бути встановлені зв’язки за допомогою формул. При цьому одні дані вважаються початковими і вводяться до таблиці із зовні (наприклад, з клавіатури), а інші дані вважаються розрахунковими і утворюються в таблиці автоматично за допомогою формул;

– подання даних у наочній графічній формі за допомогою діаграм, в тому числі за допомогою графіків та гістограм різної форми;

– проведення розрахунків за допомогою простих формул, що вводяться вручну, з використанням функцій із спеціальних математичних, економічних, планових, статистичних та інших бібліотек, а також шляхом використання мови програмування VBA (Visual Basic for Application);

– розв’язування задач, які мають оптимізаційний характер, тобто дозволяють знаходити найкращі планові показники, які забезпечують найвищий ефект управлінської діяльності;

– створення баз даних, тобто наборів даних великого обсягу, в яких існують засоби автоматизації пошуку, впорядкування та фільтрації;

– обмін даними з іншими складовими пакету Microsoft Office, а також іншими прикладними програмами персонального комп’ютера за допомогою системного буфера обміну;

– створення макросів, тобто спеціальних команд, за допомогою яких вдається автоматично виконувати послідовності однотипних операцій [15].

2.4. Опис структури системи та алгоритмів її функціонування

2.4.1. Функціональне проектування системи

Програмний продукт для автоматизації формування турнірних сіток для змагань з бойових видів спорту складається з двох частин: програмного додатку, де адміністратор формує турнірні таблиці та сітки, та вебсайту, де тренери можуть додавати дані про учасників, а неавторизовані користувачі переглядати дані про турнірні таблиці або сітки.

Щоб описати функціонал програмного продукту доцільно буде використати діаграму прецедентів (Use Case diagram).

Діаграма прецедентів – в UML, діаграма, де зображено відношення між акторами (користувачами) та прецедентами (функціями) в системі. Також, перекладається, як діаграма варіантів використання [5].

Діаграма прецедентів даного програмного продукту наведена на рис. 2.2.

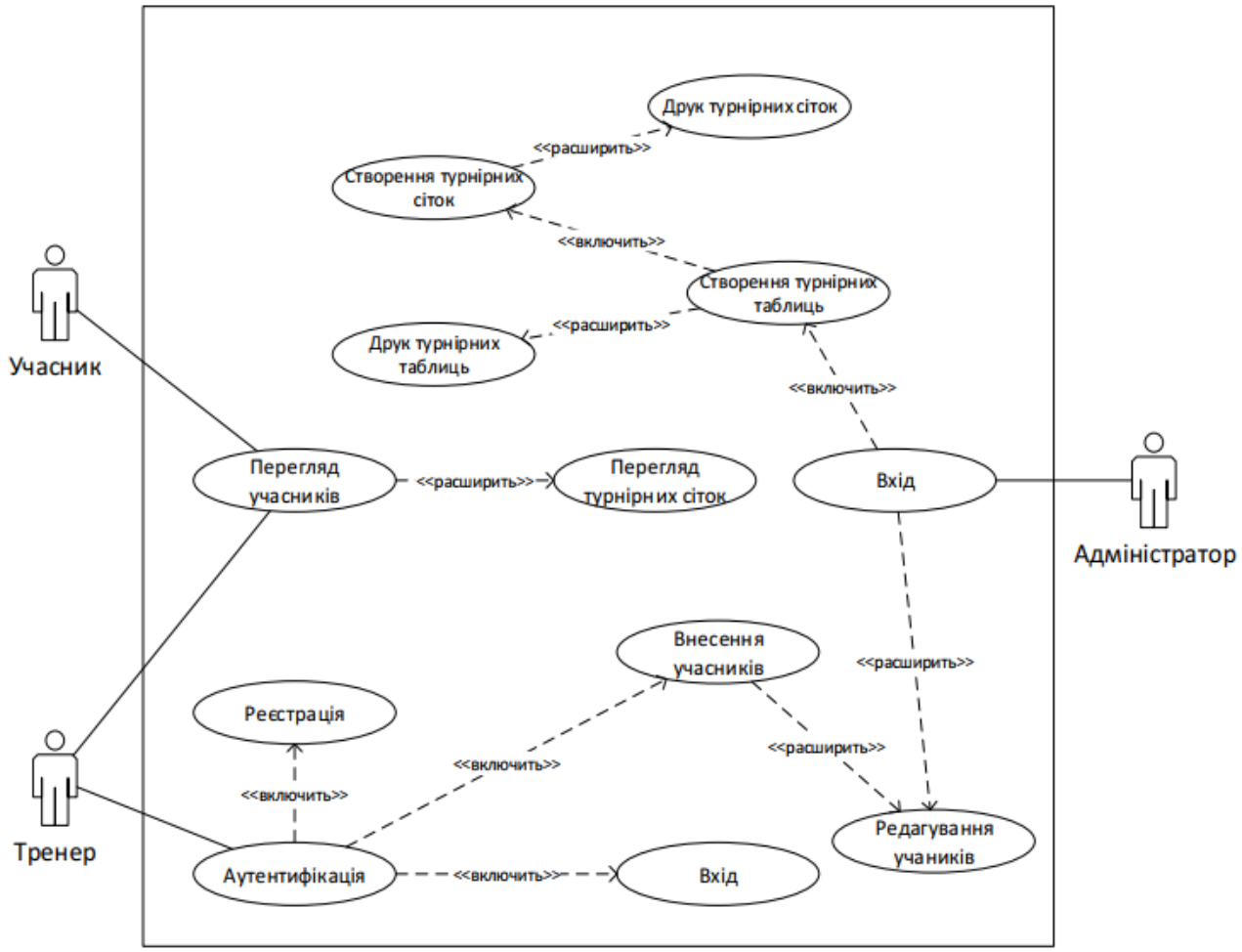

Рис. 2.2. Use Case діаграма програмного продукту

Метою даної інформаційної системи є автоматизоване створення турнірних таблиць та сіток. Саме цей процес зображений на діаграмі послідовностей (Sequence Diagram), яка зображена на рис. 2.3 та 2.4.

Діаграма послідовності – різновид діаграми в UML. Діаграма послідовності відображає взаємодії об'єктів впорядкованих за часом. Зокрема, такі діаграми відображають задіяні об'єкти та послідовність відправлених повідомлень [6].

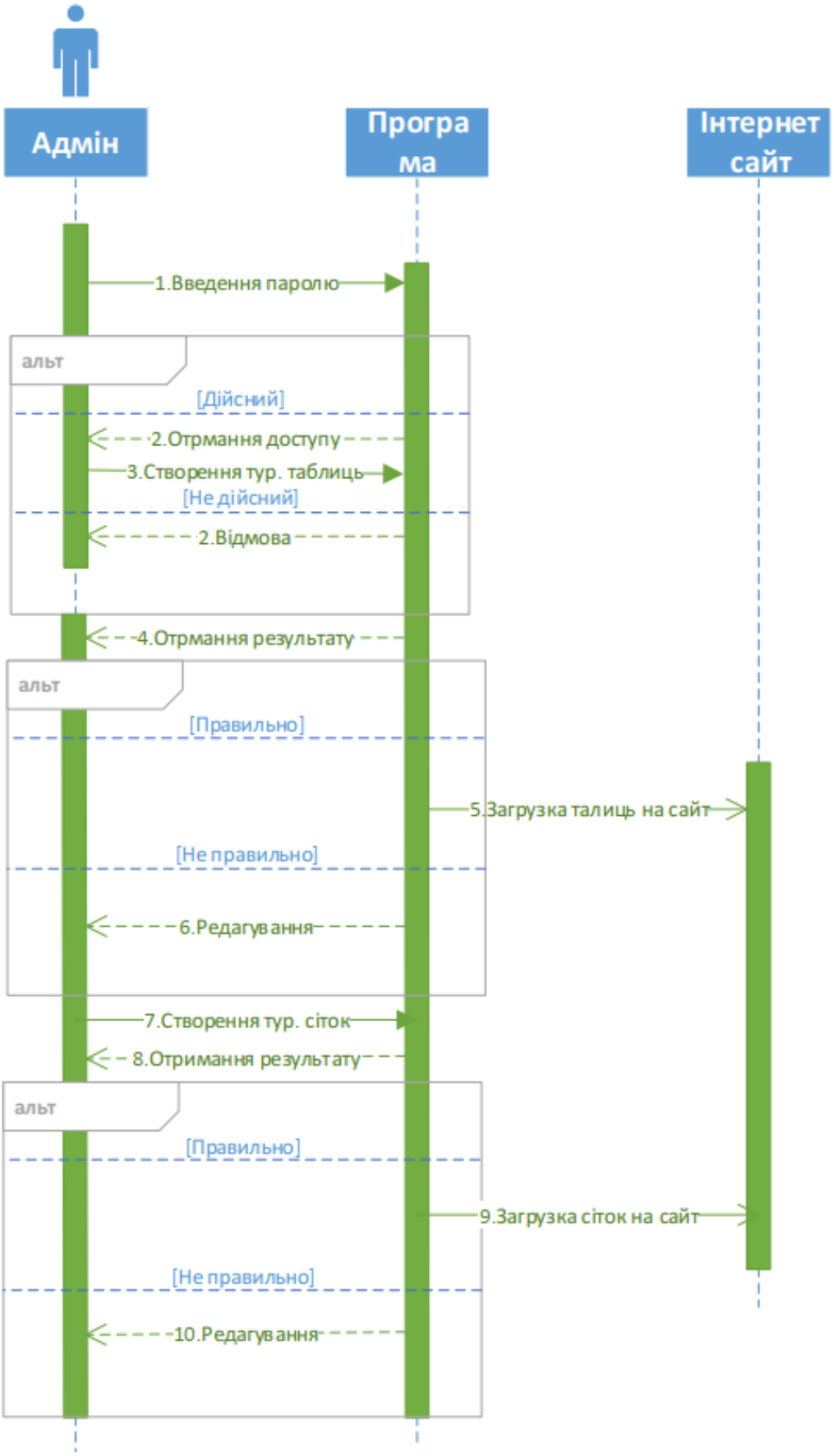

Рис. 2.3. Діаграма послідовності для користувача в ролі адміністратора

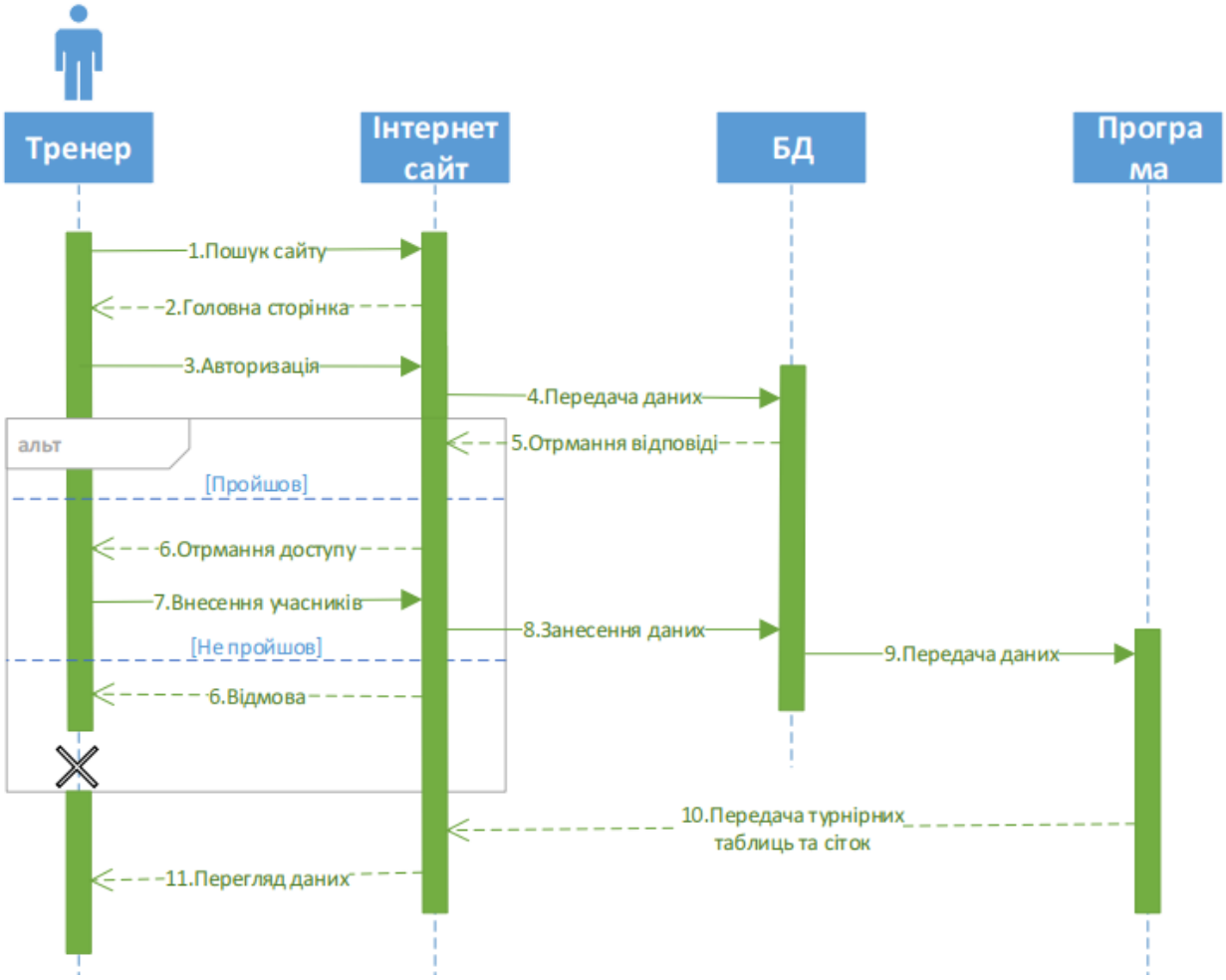

Рис. 2.4. Діаграма послідовності для користувача в ролі тренера

2.4.2. Опис структури програмного додатку ІС

2.4.2.1. Файлова структура програмного додатку

Розглянемо проектування програмного додатку, який обробляє дані про учасників, чим формує турнірні таблиці та сітки для спортивних змагань. Проект складається з файлів, які перелічені нижче.

При створенні віконного додатку типу Windows Forms відразу створюється початкова форма, файл з розширенням .cs (за замовчуванням Program.cs), AssemblyInfo.cs, в якому містяться загальні відомості про зборку проекту, Resources.resx та Settings.settings, код яких створюється самою програмою за допомогою такого засобу, як ResGen.

Файлова структура зображена на рис. 2.5.

Для забезпечення автоматизації створення турнірних таблиць та сіток були створені наступні форми:

– FormAutorization – для авторизації адміністратора;

– FormMain – для занесення даних про учасників у базу даних;

– FormGrid – для формування турнірних таблиць та сіток, а також для відображення турнірних таблиць;

– FormImages – для перегляду та друку створених графічних зображень турнірних сіток.

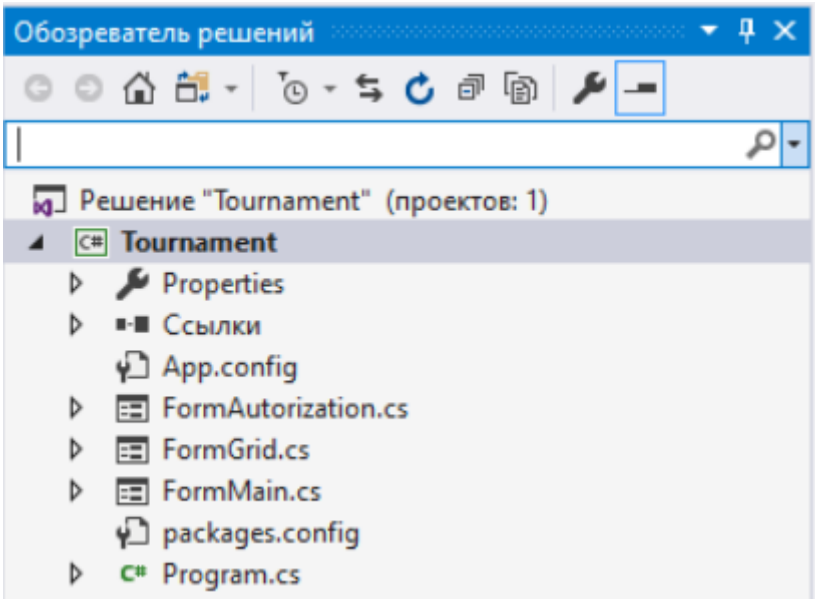

Рис. 2.5. Файлова структура проекту програмного додатку

У всіх вище перелічених формах застосовуються наступні простори імен:

– using system.collections.generi – означає, що універсальну колекцію можна створити, використовуючи один із класів у просторі імен system;

– using System.ComponentModel – надає класи, використовувані для реалізації поведінки компонентів і елементів управління під час розробки та виконання. Це простір імен містить базові класи і інтерфейси для реалізації атрибутів і перетворювачів типів, прив'язки до джерел даних і ліцензування компонентів;

– using System.Data - забезпечує доступ до класів, які представляють архітектуру ADO.NET. ADO.NET дозволяє створювати компоненти, ефективно управляють даними з декількох джерел даних;

– using System.Drawing – надає доступ до GDI, графіки основних функціональних можливостей;

– using System.Linq - містить класи та інтерфейси , які підтримують запити, що використовують Language-Integrated Query (LINQ);

– using System.Text - містить класи, що представляють ASCII і кодування символів Unicode; абстрактні базові класи для перетворення блоків символів до та з байтів блоків; і хелперний клас, який маніпулює та форматує об'єкти String, не створюючи проміжні екземпляри String;

– using System.Threading.Tasks - надає типи , які спрощують роботу написання паралельного і асинхронного коду. Основними типами завдання, яке представляє асинхронну операцію, яку можна чекати та скасовувати, і завдання <TResult> , яке є завданням, яке може повернути значення. Клас TaskFactory надає статичні методи створення та запуску завдань;

– using System.Windows.Forms - містить класи для створення додатків Windows, які дозволяють найбільш ефективно використовувати розширені можливості призначеного для користувача інтерфейсу, які надаються операційною системою Microsoft Windows;

– using System.Data.SQLite - є одним з найбільш популярних програмних движків для вбудованих реляційних баз даних, широко використовуваний розробниками для організації локального зберігання даних і ефективної роботи з ними в розроблюваних програмних продуктах [6].

2.4.2.2. Створення бази даних проєкту

Для повноцінної роботи програмного додатку була створена база даних, яка використовується для зберігання даних про учасників та адміністратора. Вона створена за допомогою СКБД SQLite.

Перейдемо до проектування бази даних (далі - БД). Виділимо базові сутності з атрибутами:

– Адміністратор. Атрибути: ідентифікатор, логін, пароль;

– Учасники. Атрибути: ідентифікатор, ім’я, дата народження, вік, вага, ступінь, тренер, клуб.

Програмний додаток розроблюється для адміністратора.

Функціональні можливості програмного додатку:

– ведення БД (читання, редагування, видалення записів);

– забезпечення логічної несуперечності БД;

– забезпечення захисту даних від несанкціонованого або випадкового доступу;

Готові запити до бази даних:

– отримання списку всіх учасників;

– отримання списків учасників по категоріям.

Для відображення інформації про сутності та їх відносини застосовують діаграм «сутність – зв’язок».

Схема «сутність-зв'язок» (також ERD або ER-діаграма) - це різновид блок-схеми, де показано, як різні «сутності» (люди, об'єкти, концепції і так далі) пов'язані між собою всередині системи.

ER-діаграма розроблюваної БД наведена на рис. 2.6.

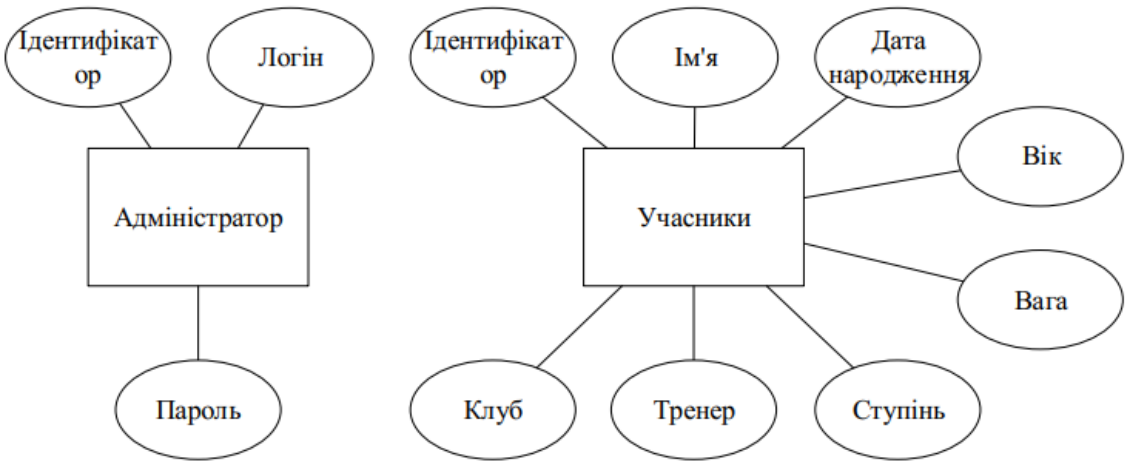

Рис. 2.6. ER-діаграма бази даних додатку

Відношення приведені в таблицях 2.1 та 2.2. Для кожного відношення вказані атрибути з їх внутрішньою назвою, типом та довжиною. Типи даних позначаються наступним чином: N – числовий, C – символьний.

Таблиця 2.1

Таблиця «Адміністратор» (Admins)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Ідентифікатор	id	N	PK
Логин	surname	C(30)	
Пароль	login	C(30)	

Таблиця 2.2

Таблиця «Учасники» (Participants)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Ідентифікатор	id	N	PK
Учасник	name	C(50)	
Дата народження	birthday	C(10)	
Вік	Age	N	
Вага	Weight	N	
Ступінь	Ku	C(5)	
Тренер	Instructor	C(50)	
Клуб	Club	C(20)	

Для зберігання даних про учасників змагань, які додаються у систему тренером засобами вебсайту, використовується база даних, для реалізації якої була обрана СУБД MS SQL Server.

MS SQL Server має вбудовану підтримку .NET Framework. Завдяки цьому, процедури бази даних, що зберігаються, можуть бути написані на будь-якій мові платформи .NET з використанням повного набору бібліотек, доступних для .NET Framework. На відміну від інших процесів, .NET Framework виділяє додаткову пам'ять і будує засоби керування SQL Server, не

використовуючи вбудовані засоби Windows. Це підвищує продуктивність порівняно із загальними алгоритмами Windows, оскільки алгоритми розподілу ресурсів спеціально налагоджені для використання у структурах SQL Server [11].

БД повинна містити інформацію про: учасників змагань; тренерів (авторизованих користувачів), які додають дані про учасників до БД; адміністратора, який завантажує дані щодо турнірних таблиць та сіток.

У відповідності з предметною областю система будується з урахуванням наступних особливостей:

– у кожного користувача повинна бути своя роль;

– якщо користувач зареєстрований в системі, то він має змогу додавати учасників на змагання;

– користувач може додати декількох учасників на змагання.

Виділимо базові сутності з атрибутами:

Користувачі. Атрибути: ідентифікатор, електронна пошта, підтвердження електронної пошти, пароль, штамп безпеки, телефонний номер, підтверджений телефонний номер, два фактори включені, дата закінчення блокування, блокування включене, без доступу, ім’я користувача, роль, тип вимог, значення вимог, логін провайдера, ключ провайдера;

Учасники. Атрибути: ідентифікатор, ідентифікатор користувача, прізвище ім’я, дата народження, країна, регіон, клуб, організація, ступінь, категорія, картинка;

Міграції. Атрибути: ідентифікатор міграції, контекстний ключ, модель, версія продукту.

Турнірні сітки. Атрибути: назва, з років, по років, з ваги, по вагу, статус, шлях до зображення турнірної сітки.

ER-діаграма розроблюваної бази даних приведена на рис. 2.7.

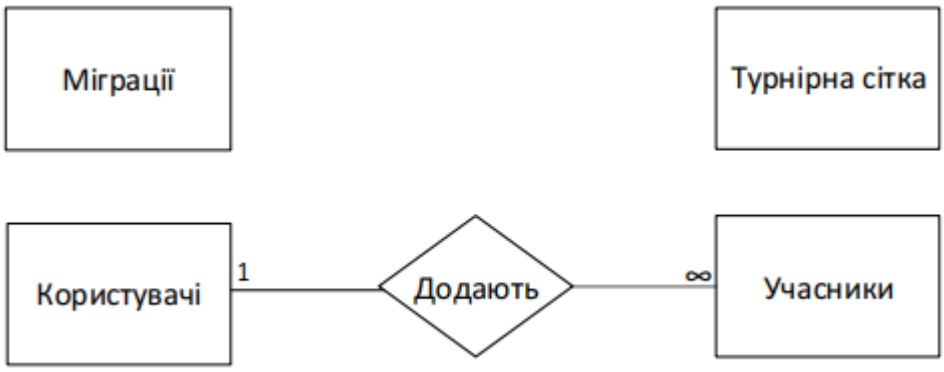

Рис .2.7. ER-діаграма розроблюваної бази даних

Відношення «Міграції» на має зв’язку з іншими відношеннями, так як воно потрібно виключно для оновлення бази даних у випадку зміни моделі даних. У цьому відношенні зберігається серіалізована в двійковий формат модель даних.

Відношення «Турнірні сітки» також не має зв’язку з іншими відношеннями, так як воно потрібно для зміни статусу та додавання зображень турнірних сіток, які були створені раніше.

База даних створюється на основі схеми бази даних. Для перетворення ER-діаграма в схему БД наведемо уточнену ER-діаграму, що містить атрибути сутностей (рис. 2.8).

Так як дані в атрибуті Роль будуть повторюватися, то для коректнішої роботи бази даних необхідно створити нове відношення «Ролі», де будуть зберігатися ідентифікатори користувачів та назви їхніх ролей. Також необхідно додати відношення «Вимоги користувача», для зберігання додаткової інформації користувачів, де будуть зберігатися наступні дані: ідентифікатор відповідного користувача, тип вимог і значення вимог. Аналогічно потрібно додати відношення «Логіни користувачей» для зберігання інформації про сторонні/зовнішні входи, наприклад користувачів, які входять на даний сайт через Google, Facebook, Twitter тощо. У цьому відношенні будуть зберігатися дані про логін та ключ провайдеру до кожного користувача.

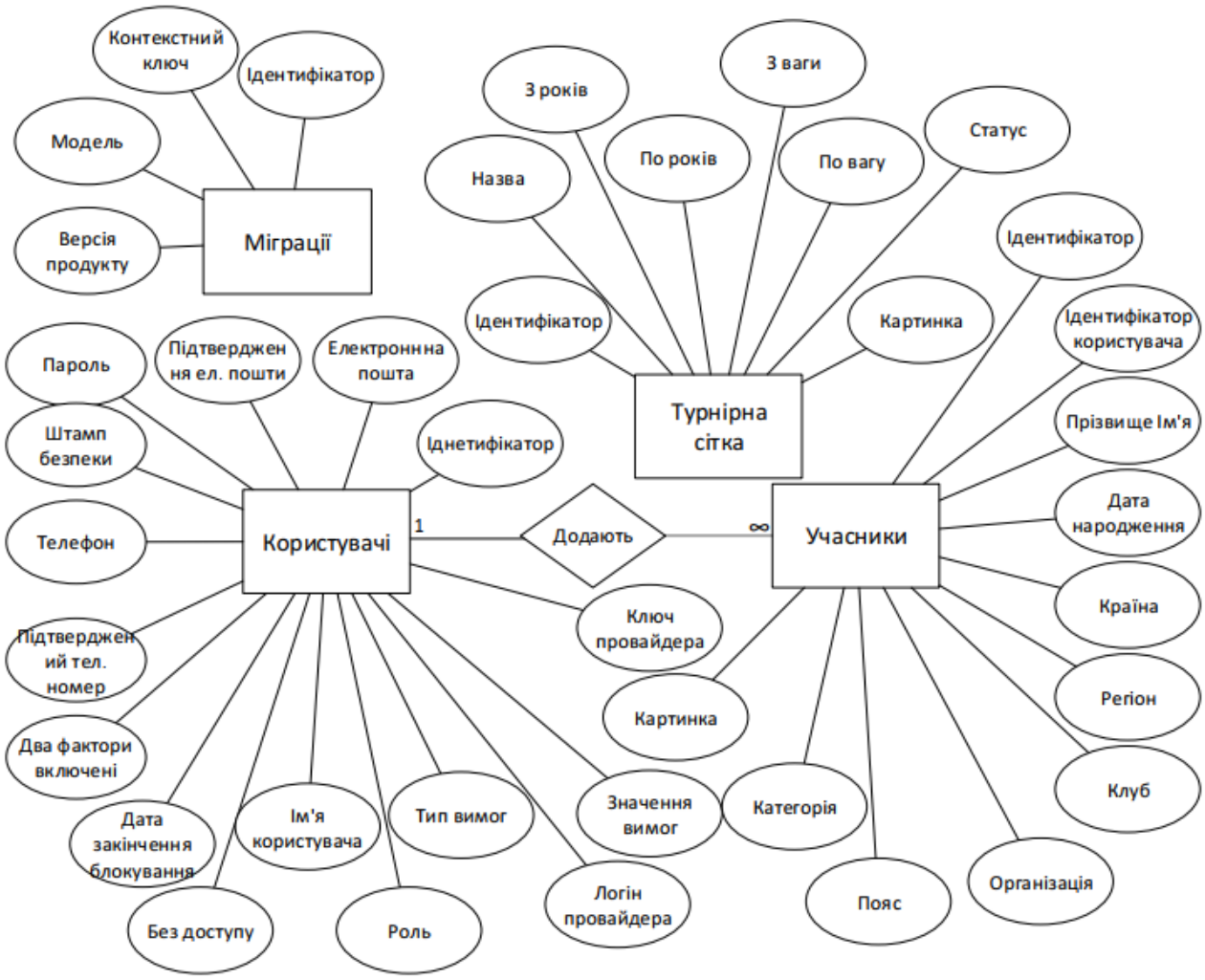

Рис. 2.8. Уточнена ER-діаграма розроблюваної бази даних

Перетворення ER-діаграми в схему БД виконується шляхом зіставлення кожної сутності і кожному зв'язку, що має атрибути, відносини (рис. 2.9).

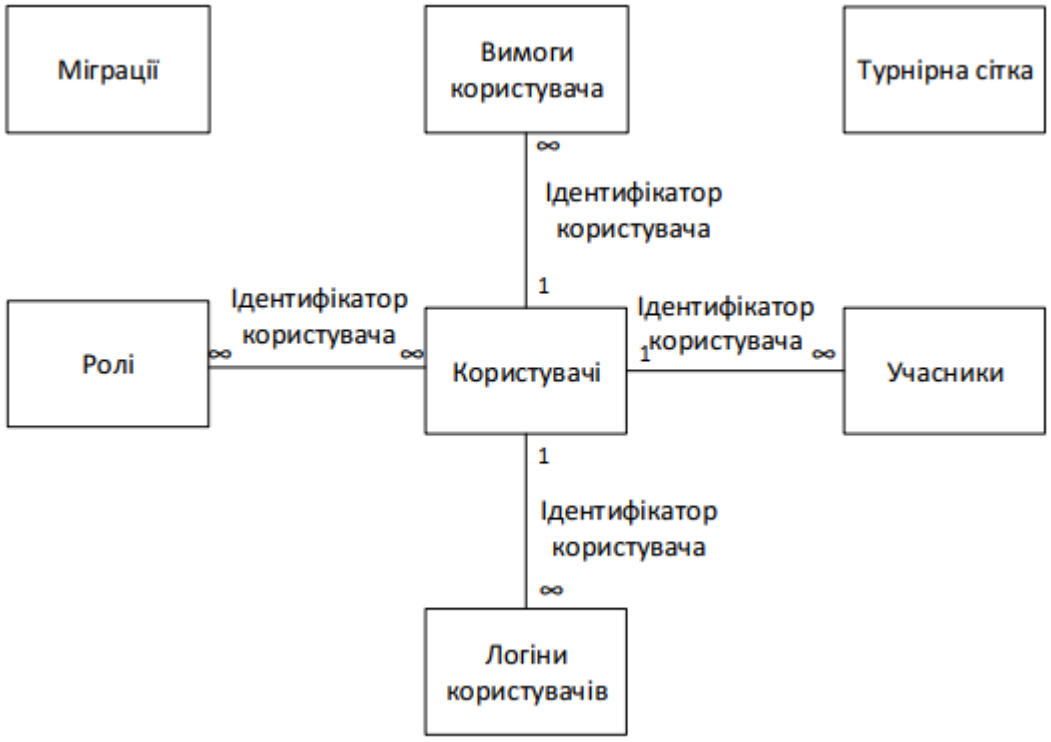

Рис. 2.9. Схема РБД, отримана із ER-діаграма розроблюваної системи

На даному етапі є зв’язок типу n:m (багато до багатьох) – зв’язок між відношеннями «Користувачі» та «Ролі». Такий зв’зок слід реалізувати через допоміжне відношення «Користувацькі ролі», яке являється з'єднанням первинних ключів відповідних відношень. У відношенні «Користувацькі ролі» будуть зберігатись ідентифікатори користувачей та відповідних їм ролей.

Уточнена схема РБД наведена на рис. 2.10.

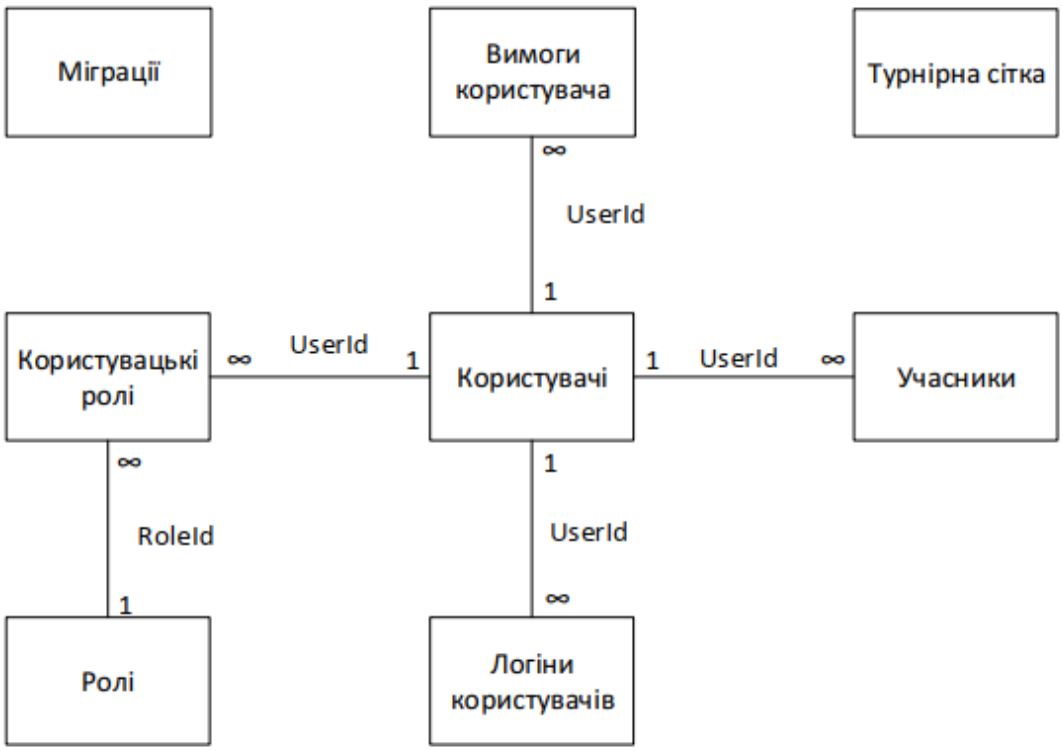

Рис. 2.10. Уточнена схема РБД, отримана із ER-діаграма розроблюваної системи

Для кожного відношення вказані атрибути з їх внутрішньою назвою, типом та довжиною. Типи даних позначаються наступним чином: N – числовий, F – числовий з плаваючою комою, C – символьний (nvarchar), B – бітовий, D – дата, V – бінарний; ключі: PK – первинний ключ, FK – зовнішній ключ.(табл. 2.3-2.10).

Таблиця 2.3

Таблиця «Вимоги користувача» (AspUserClaims)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Ідентифікатор	Id	N	PK
Ідентифікатор користувача	UserId	C(128)	FK
Тип вимог	ClaimType	C(128)	
Значення вимог	ClaimValue	C(128)	

Таблиця 2.4

Таблиця «Ролі» (AspNetRoles)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Ідентифікатор	id	C(128)	PK
Ім’я	name	C(256)	

Таблиця 2.5

Таблиця «Користувацькі ролі» (AspNetUserRoles)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Ідентифікатор ролі	UserId	C(128)	PK
Ідентифікатор користувача	RoleId	C(128)	FK

Таблиця 2.6

Таблиця «Користувачі» (AspNetUsers)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Ідентифікатор	Id	C(128)	PK
Електронна пошта	Email	C(128)	
Підтвердження електронної пошти	EmailConfirmed	B	
Пароль	PasswordHash	C(MAX)	
Штамп безпеки	SecurityStamp	C(128)	
Телефон	PhoneNumber	C(128)	
Підтверджуваний телефон	PhoneNumberConfirmed	B	
Два фактори включені	TwoFactorEnabled	B	
Дата закінчення блокування	LockoutEndDateUtc	D	
Без доступу	AccessFailedCount	N	
Імя	FirstNume	C(256)	
Прізвище	SecondNume	C(256)	
Фото	ImgUrl	C(256)	

Таблиця 2.7

Таблиця «Логіни користувачів» (AspNetUserLogins)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Логін провайдера	LoginProvider	C(128)	PK
Ключ провайдера	ProviderKey	C(128)	
Ідентифікатор користувача	UserId	C(128)	FK

Таблиця 2.8

Таблиця «Історія міграції» (_MigrationHistory)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Ідентифікатор	MigrationId	C(150)	PK
Контекстний ключ	ContextKey	C(300)	
Модель	Model	V	
Версія продукту	ProductVersion	C(32)	

Таблиця 2.9

Таблиця «Турнірні сітки» (TournamentGrid)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Ідентифікатор	Id	N	PK
Назва сітки	Content	N	
З років	AgeWith	N	
По років	AgeTo	D	
З ваги	WeigthWith	F	
По вагу	WeghyTo	F	
Статус	Status	B	
Картинка	ImgUrl	C(128)	

Таблиця 2.10

Таблиця «Учасники» (Participants)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Ідентифікатор	Id	N	PK
Ідентифікатор користувача	ApplicationUserId	C(128)	FK
Ім’я	Initsials	C(128)	
Дата народження	DateOfBirth	D	
Країна	Country	C(128)	
Регіон	Region	C(128)	
Клуб	Dojo	C(128)	
Організація	Organisation	C(128)	
Ступінь	Grade	C(128)	
Категорія	Category	C(128)	
Років	AgeTo	N	
Вага	Weight	F	
Картинка	ImagePath	C(128)	

Слід також зазначити, що на етапі створення уточненої схеми РБД були пройдені усі етапи нормалізації, а саме були розбиті складні атрибути на прості (Ім’я користувача на Ім’я та Прізвище) – 1 нормальна форма (далі - НФ). Не ключові атрибути у відношенні «Користувацькі ролі», яке має складовий первинний ключ, функціонально повністю залежать від первинних ключів – 2НФ. Всі дані в таблицях залежать винятково від первинного ключа – 3НФ. Відношення у даному випадку не порушують правил 4НФ, так як не містять нетривіальних багатозначних залежностей.

Таким чином в результаті була створена остаточна схема реляційної бази даних засобами СКБД (рис. 2.11).

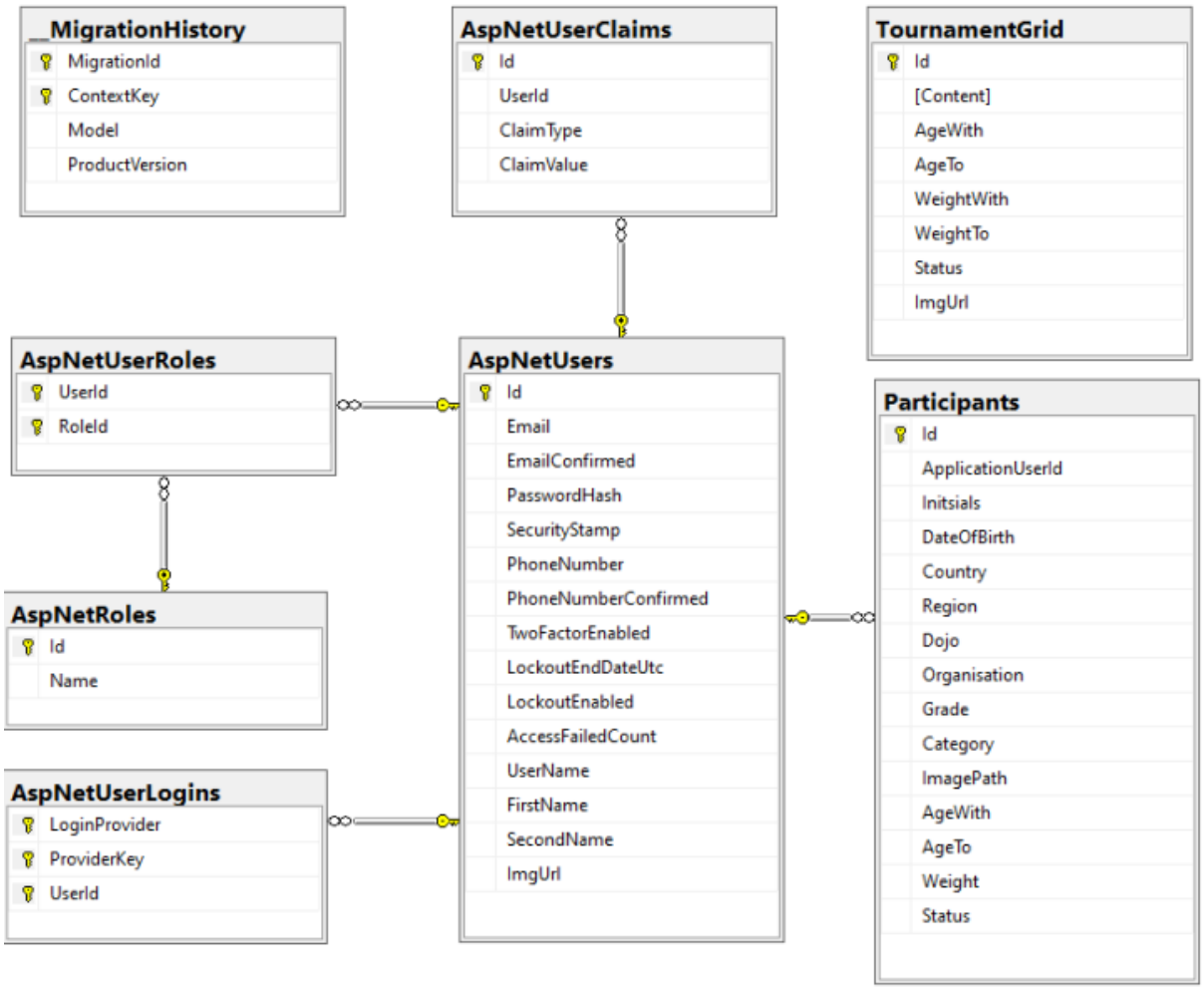

Рис. 2.11. Остаточна схема РБД розроблювальної системи

Додаткові обмеження цілісності:

– значення всіх числових атрибутів – більше 0 (або null, якщо поле необов’язкове);

– в атрибуті ImgUrl відношення «AspNetUsers» вказується шлях до зображення турнірної сітки;

– у відношенні «TournamentGrid» значення первинного ключа по-вині іти виключно по порядку, інакше можлива некоректна робота системи.

Нижче перерахований опис груп користувачів та прав доступу. Звичайні користувачі системи мають можливість тільки переглядати відомості про учасників змагань та сформовані турнірні таблиці і сітки на сайті.

Тренери мають можливість вносити та редагувати відомості про учасників і переглядати сформовані турнірні таблиці та сітки на сайті.

Адміністратор має можливість переглядати та видаляти відомості про учасників та формувати з загальної таблиці учасників турнірні таблиці і сітки.

2.4.2.3. Проектування інтерфейсу програмного додатку

Розглянемо проектування форм додатку. Додаток складається з чотирьох основних форм. Вони мають візуальні та не візуальні компоненти. Форми та їхні компоненти і події описані нижче.

Форма «FormAutorization» призначена для надання прав доступу користувачу, тобто щоб користувач мав змогу перейти до процесу створення турнірних таблиць та сіток. Ескіз форми зображений на рис. 2.7. Застосовані компоненти та події на формі описані у таблицях 2.11-2.12.

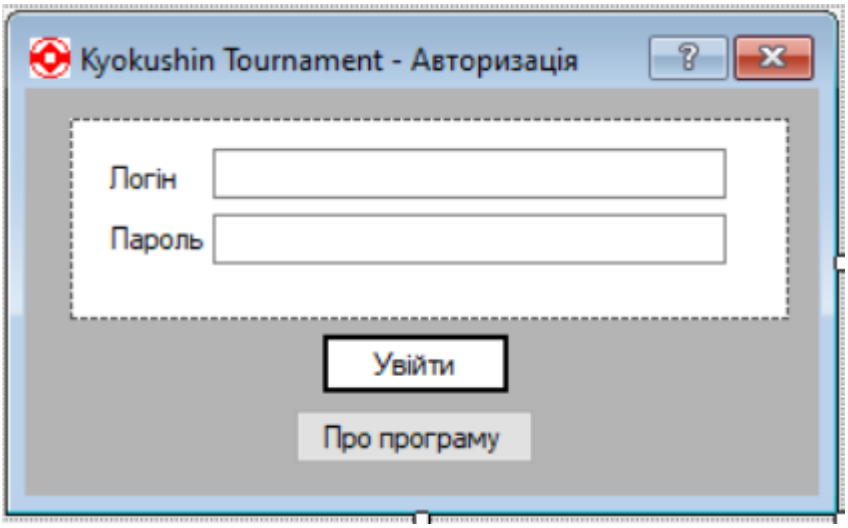

Рис. 2.7. Ескіз форми «FormAutorization»

Таблиця 2.11

Події форми FormAutorization

Назва події	Призначення
ButtonEnter_Click	Здійснює вхід в систему
FormAutorization_FormClosed	Здійснюється при закритті форми
FormAutorization_HelpButtonClicked	Відкриває вікно підказки

Таблиця 2.12

Компоненти форми FormAutorization

Тип компоненту	Назва компоненту	Призначення
Panel	Panel1	Для групування елементів керування вводу логіну та паролю
Button	buttonEnter	Для переходу на форму FormMain з відповідними правами доступу
Button	button1	Для відображення діалогового вікна з текстом про програму
Label	labelAdminLogin	Мітка для позначення відповідного поля
Label	labelAdminPassword	Мітка для позначення відповідного поля
TextBox	textBoxAdminLogin	Для вводу імені адміністратора
TextBox	textBoxAdminPassword	Для вводу паролю адміністратора

Форма FormMain призначена для заповнення бази даних даними про учасників. Ескіз форми зображений на рис. 2.8. Застосовані компоненти та події на даній формі описані у таблицях 2.13 і 2.14.

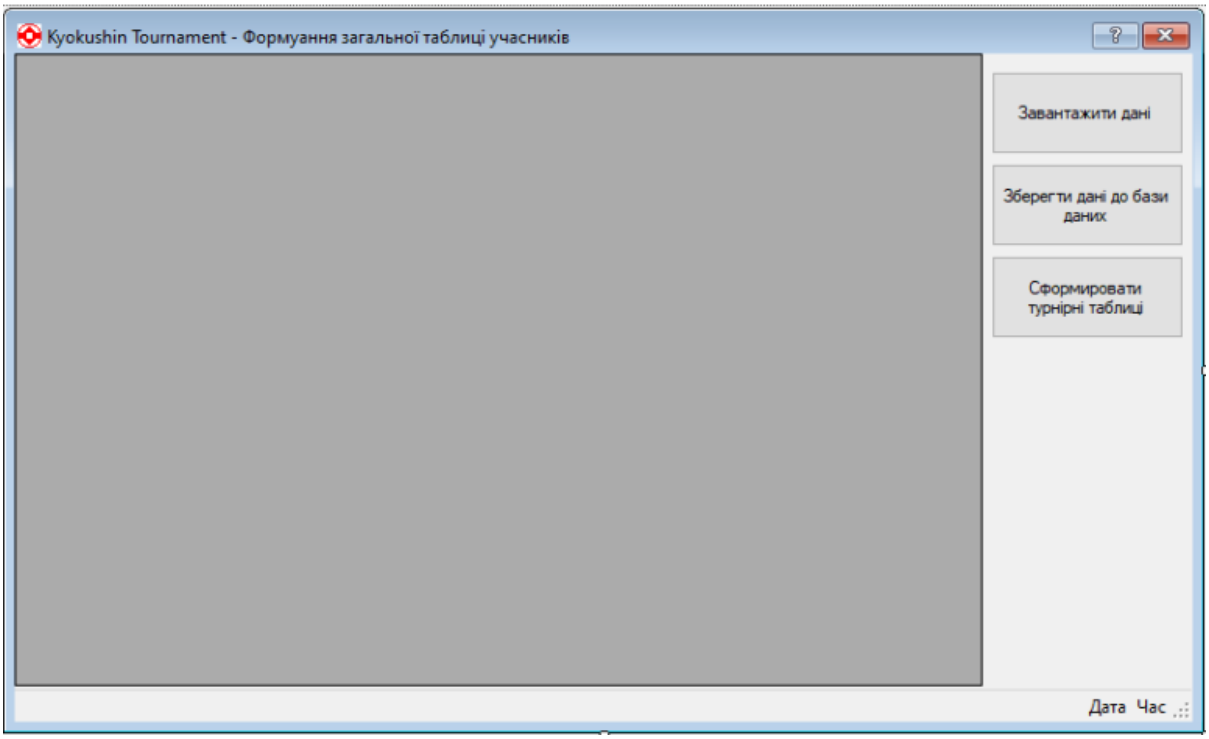

Рис. 2.8. Ескіз форми «FormMain»

Таблиця 2.13

Компоненти форми FormMain

Тип компоненту	Назва компоненту	Призначення
dataGridView	dataGridViewParticipant	Для відображення списку учас ників
Button	button1	Для завантаження даних з вебсайту до бази даних
Button	button2	Для оновлення даних у базі даних
Button	button3	Для переходу до створення тур нірних таблиць та сіток
Statustrip	Statustrip1	Рядок стану для відображення поточної дати та часу

Таблиця 2.14

Події форми FormMain

Назва події	Призначення
LoadExcelFile	Завантажує дані в dataGridView Participant
UploadParticipantsInDatabase	Оновлює дані у базі даних
LoadFormGrid	Здійснюється перехід на форму FormGrid
FormAutorization_HelpButtonClick ed	Відкриває вікно підказки для створення турнірних таблиць та сіток

Форма «FormGrid» призначена для формування, перегляду та збереження турнірних таблиць. Також вона призначена для створення графічного зображення турнірної сітки. Ескіз форми зображений на рис. 2.9. Застосовані компоненти та події на даній формі описані у таблицях 2.15 і 2.16.

Таблиця 2.15

Компоненти форми FormGrid

Тип компоненту	Назва компоненту	Призначення
tabControl	tabControlMain	Для відображення даних по вікових і вагових категоріях
tabControl	tabControl67	Для відображення даних по ваговій категорії 6-7років
tabControl	tabControl89	Для відображення даних по ваговій категорії 8-9 років
tabControl	tabControl1011	Для відображення даних по ваговій категорії 10-11 років
tabControl	tabControl1213	Для відображення даних по ваговій категорії 12-13 років
tabControl	tabControl1415	Для відображення даних по ваговій категорії 14-15 років
tabControl	tabControl1617	Для відображення даних по ваговій категорії 16-17років
tabControl	tabControl18	Для відображення даних по ваговій категорії 18+ років
dataGridView	dataGridView67bef25	Для відображення таблиці даних учасників у категорії 6-7 років до 25кг
dataGridView	dataGridView67bef30	Для відображення таблиці даних учасників у категорії 6-7 років до 30кг
dataGridView	dataGridView67af30	Для відображення таблиці даних учасників у категорії 6-7 років понад 30кг
dataGridView	dataGridView89bef30	Для відображення таблиці даних учасників у категорії 8-9 років до 30кг
dataGridView	dataGridView89bef35	Для відображення таблиці даних учасників у категорії 8-9 років до 35кг
dataGridView	dataGridView89bef40	Для відображення таблиці даних учасників у категорії 8-9 років до 40кг
dataGridView	dataGridView89af40	Для відображення таблиці даних учасників у категорії 8-9 років понад 40кг

Продовж. табл. 2.15

Тип компоненту	Назва компоненту	Призначення
dataGridView	dataGridView1011bef35	Для відображення таблиці даних учасників у категорії 10-11 років до 35кг
dataGridView	dataGridView1011bef40	Для відображення таблиці даних учасників у категорії 10-11 років до 40кг
dataGridView	dataGridView1011bef45	Для відображення таблиці даних учасників у категорії 10-11 років до 45кг
dataGridView	dataGridView1011bef50	Для відображення таблиці даних учасників у категорії 10-11 років до 50кг
dataGridView	dataGridView1011af50	Для відображення таблиці даних учасників у категорії 10-11 років понад 50кг
dataGridView	dataGridView1213bef40	Для відображення таблиці даних учасників у категорії 12-13 років до 40кг
dataGridView	dataGridView1213bef45	Для відображення таблиці даних учасників у категорії 12-13років до 45кг
dataGridView	dataGridView1213bef50	Для відображення таблиці даних учасників у категорії 12-13 років до 50кг
dataGridView	dataGridView1213bef55	Для відображення таблиці даних учасників у категорії 12-13років до 55кг
dataGridView	dataGridView1213af55	Для відображення таблиці даних учасників у категорії 12-13 років понад 55кг
dataGridView	dataGridView1415bef50	Для відображення таблиці даних учасників у категорії 14-15 років до 50кг
dataGridView	dataGridView1415bef55	Для відображення таблиці даних учасників у категорії 14-15 років до 55кг

Закінч. табл. 2.15

Тип компоненту	Назва компоненту	Призначення
dataGridView	dataGridView1415bef60	Для відображення таблиці даних учасників у категорії 14-15 років до 60кг
dataGridView	dataGridView1415bef65	Для відображення таблиці даних учасників у категорії 14-15 років до 65кг
dataGridView	dataGridView1415af65	Для відображення таблиці даних учасників у категорії 14-15 років понад 65кг
dataGridView	dataGridView1617bef60	Для відображення таблиці даних учасників у категорії 16-17 років до 60кг
dataGridView	dataGridView1617bef65	Для відображення таблиці даних учасників у категорії 16-17 років до 65кг
dataGridView	dataGridView1617bef70	Для відображення таблиці даних учасників у категорії 16-17 років до 70кг
dataGridView	dataGridView1617bef75	Для відображення таблиці даних учасників у категорії 16-17 років до 75кг
dataGridView	dataGridView1617af75	Для відображення таблиці даних учасників у категорії 16-17 років понад 75кг
dataGridView	dataGridView18bef70	Для відображення таблиці даних учасників у категорії 18+ років до 70кг
dataGridView	dataGridView18bef75	Для відображення таблиці даних учасників у категорії 18+ років до 75кг
dataGridView	dataGridView18bef80	Для відображення таблиці даних учасників у категорії 18+ років до 80кг
dataGridView	dataGridView18bef85	Для відображення таблиці даних учасників у категорії 18+ років до 85кг
dataGridView	dataGridView18af85	Для відображення таблиці даних учасників у категорії 18+ років понад 85кг

Таблиця 2.16

Події форми FormGrid

Назва події	Призначення
ViewGridToolStripMenuItem_Click	Відкриває форму перегляду зображень турнірних сіток
SaveGridsInExcelToolStripMenuItem_Click	Зберігає турнірні таблиці у файл
CreateGraphicGridToolStripMenuItem_Click	Створює та збергіє графічне зображення турнірної сітки
FormAutorization_HelpButtonClicked	Відкриває вікно підказки

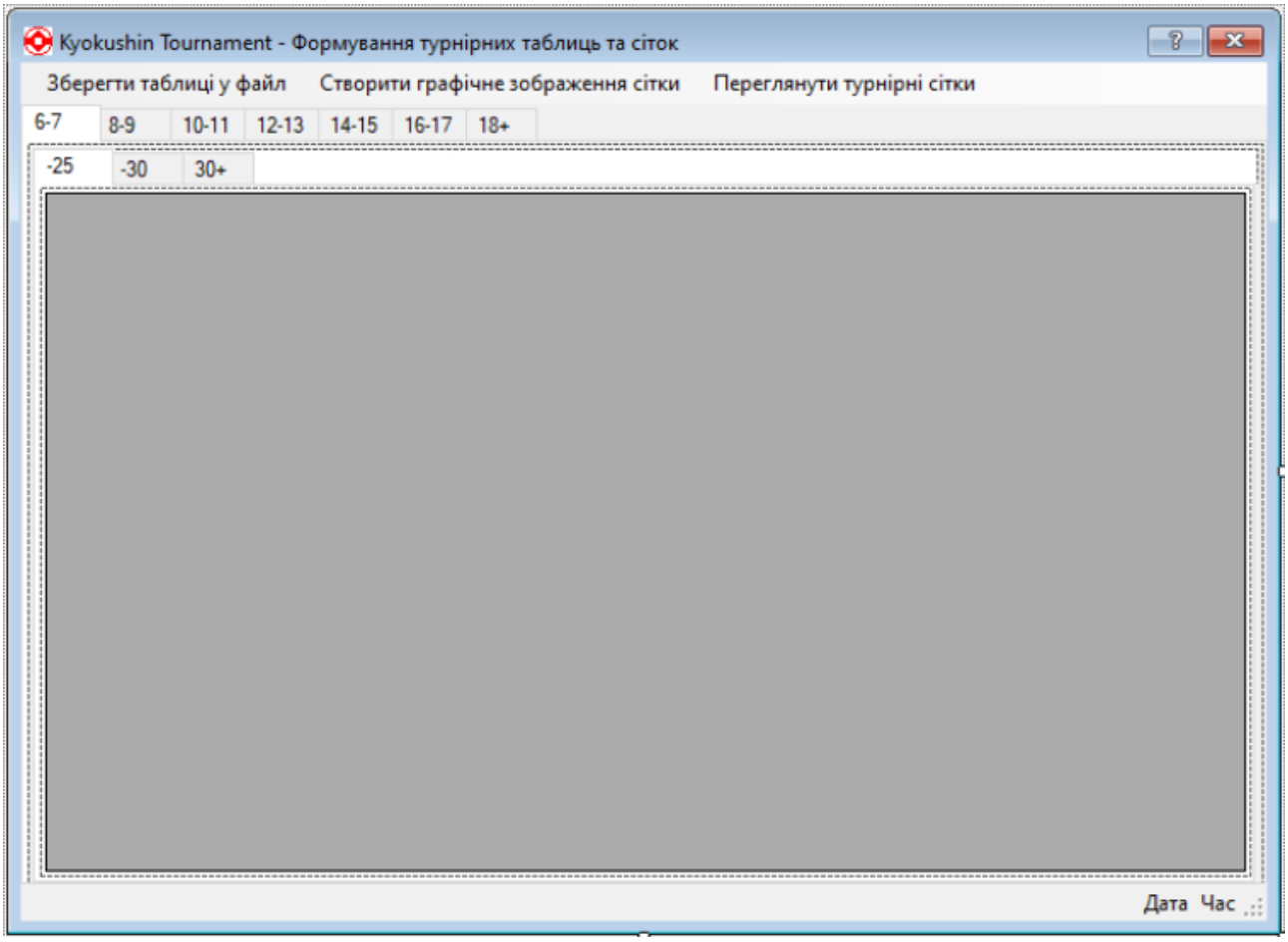

Рис. 2.9. Ескіз форми «FormGrid»

Форма «FormImages» призначена для перегляду турнірних сіток у графічній формі та можливості вивести їх на друк. Ескіз форми зображений на рис. 2.10. Застосовані компоненти та події на даній формі описані у таблицях 2.17 і 2.18.

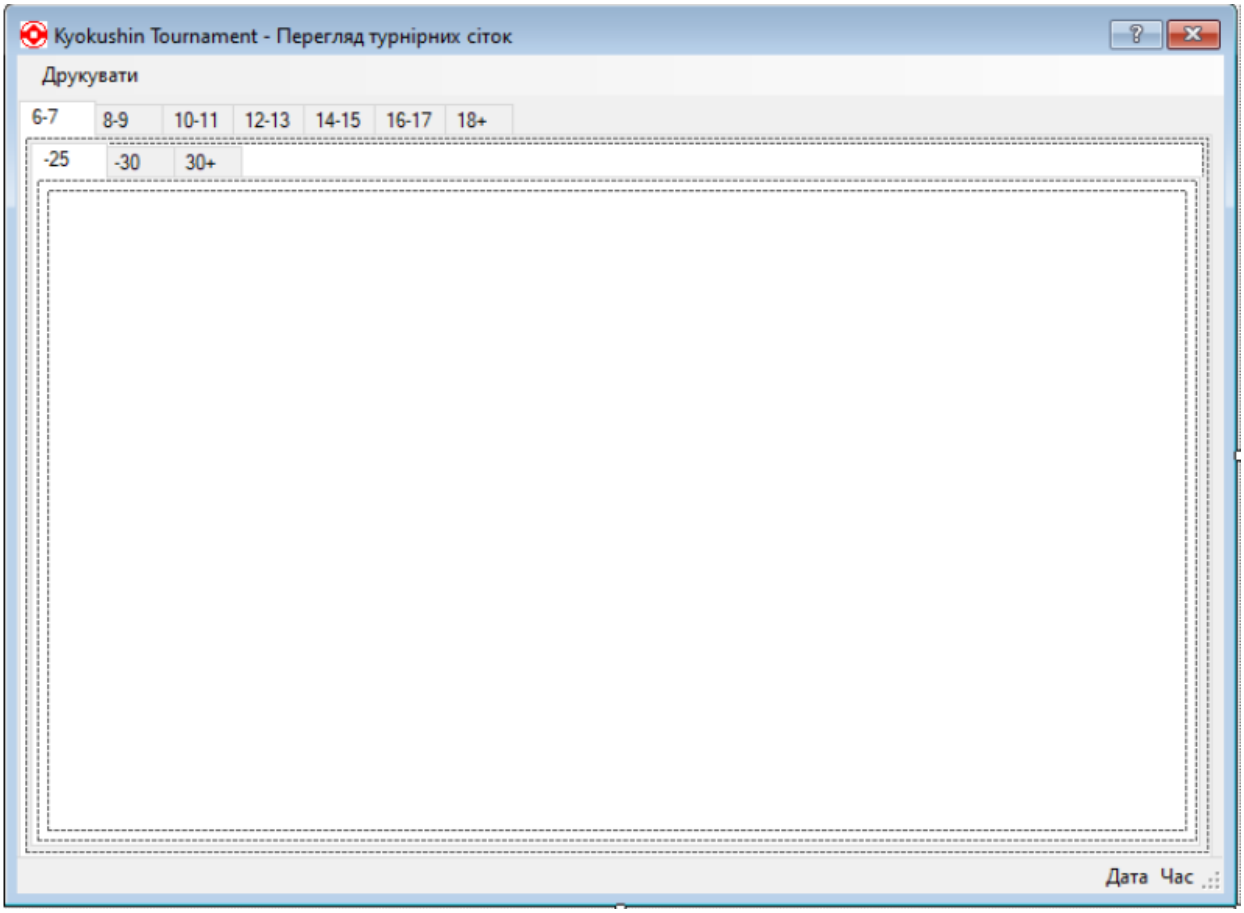

Рис. 2.10. Ескіз форми «FormImages»

Таблиця 2.17

Компоненти форми FormImages

Тип компоненту	Назва компоненту	Призначення
menuStrip	menuStrip1	Для відображення списку функцій
tabControl	tabControlMain	Для відображення даних по вікових і вагових категоріях
tabControl	tabControl67	Для відображення даних по ваговій категорії 6-7років
tabControl	tabControl89	Для відображення даних по ваговій категорії 8-9 років
tabControl	tabControl1011	Для відображення даних по ваговій категорії 10-11 років
tabControl	tabControl1213	Для відображення даних по ваговій категорії 12-13 років
tabControl	tabControl1415	Для відображення даних по ваговій категорії 14-15 років

Продовж. табл. 2.17

Тип компоненту	Назва компоненту	Призначення
tabControl	tabControl1617	Для відображення даних по ваговій категорії 16-17років
tabControl	tabControl18	Для відображення даних по ваговій категорії 18+ років
dataGridView	dataGridView67bef25	Для відображення таблиці даних учасників у категорії 6-7 років до 25кг
dataGridView	dataGridView67bef30	Для відображення таблиці даних учасників у категорії 6-7 років до 30кг
dataGridView	dataGridView67af30	Для відображення таблиці даних учасників у категорії 6-7 років понад 30кг
dataGridView	dataGridView89bef30	Для відображення таблиці даних учасників у категорії 8-9 років до 30кг
dataGridView	dataGridView89bef35	Для відображення таблиці даних учасників у категорії 8-9 років до 35кг
dataGridView	dataGridView89bef40	Для відображення таблиці даних учасників у категорії 8-9 років до 40кг
dataGridView	dataGridView89af40	Для відображення таблиці даних учасників у категорії 8-9 років понад 40кг
dataGridView	dataGridView1011bef35	Для відображення таблиці даних учасників у категорії 10-11 років до 35кг
dataGridView	dataGridView1011bef40	Для відображення таблиці даних учасників у категорії 10-11 років до 40кг
dataGridView	dataGridView1011bef45	Для відображення таблиці даних учасників у категорії 10-11 років до 45кг
dataGridView	dataGridView1011bef50	Для відображення таблиці даних учасників у категорії 10-11 років до 50кг

Закінч. табл. 2.17

Тип компоненту	Назва компоненту	Призначення
dataGridView	dataGridView1011af50	Для відображення таблиці даних учасників у категорії 10-11 років понад 50кг
dataGridView	dataGridView1213bef40	Для відображення таблиці даних учасників у категорії 12-13 років до 40кг
dataGridView	dataGridView1213bef45	Для відображення таблиці даних учасників у категорії 12-13років до 45кг
dataGridView	dataGridView1213bef50	Для відображення таблиці даних учасників у категорії 12-13 років до 50кг
dataGridView	dataGridView1213bef55	Для відображення таблиці даних учасників у категорії 12-13років до 55кг
dataGridView	dataGridView1213af55	Для відображення таблиці даних учасників у категорії 12-13 років понад 55кг
dataGridView	dataGridView1415bef50	Для відображення таблиці даних учасників у категорії 14-15 років до 50кг
dataGridView	dataGridView1415bef55	Для відображення таблиці даних учасників у категорії 14-15 років до 55кг
dataGridView	dataGridView1415bef60	Для відображення таблиці даних учасників у категорії 14-15 років до 60кг
dataGridView	dataGridView1415bef65	Для відображення таблиці даних учасників у категорії 14-15 років до 65кг
dataGridView	dataGridView1415af65	Для відображення таблиці даних учасників у категорії 14-15 років понад 65кг
dataGridView	dataGridView1617bef60	Для відображення таблиці даних учасників у категорії 16-17 років до 60кг
dataGridView	dataGridView1617bef65	Для відображення таблиці даних учасників у категорії 16-17 років до 65кг

Таблиця 2.18

Події форми FormImages

Назва події	Призначення
PrintToolStripMenuItem_Click	Виваодить на друк зображення турнірних сіток
FormAutorization_HelpButtonClicked	Відкриває вікно підказки

2.4.2.4. Опис алгоритмів програмного податку

Алгоритми роботи головних процесів, що відбуваються під час роботи програмного додатку зручно описати за допомогою блок-схем.

На рис. 2.11. зображена блок-схема алгоритму процесу створення турнірних таблиць.

На рис. 2.12-2.15. зображена блок-схема процесу створення графічного зображення турнірних сіток.

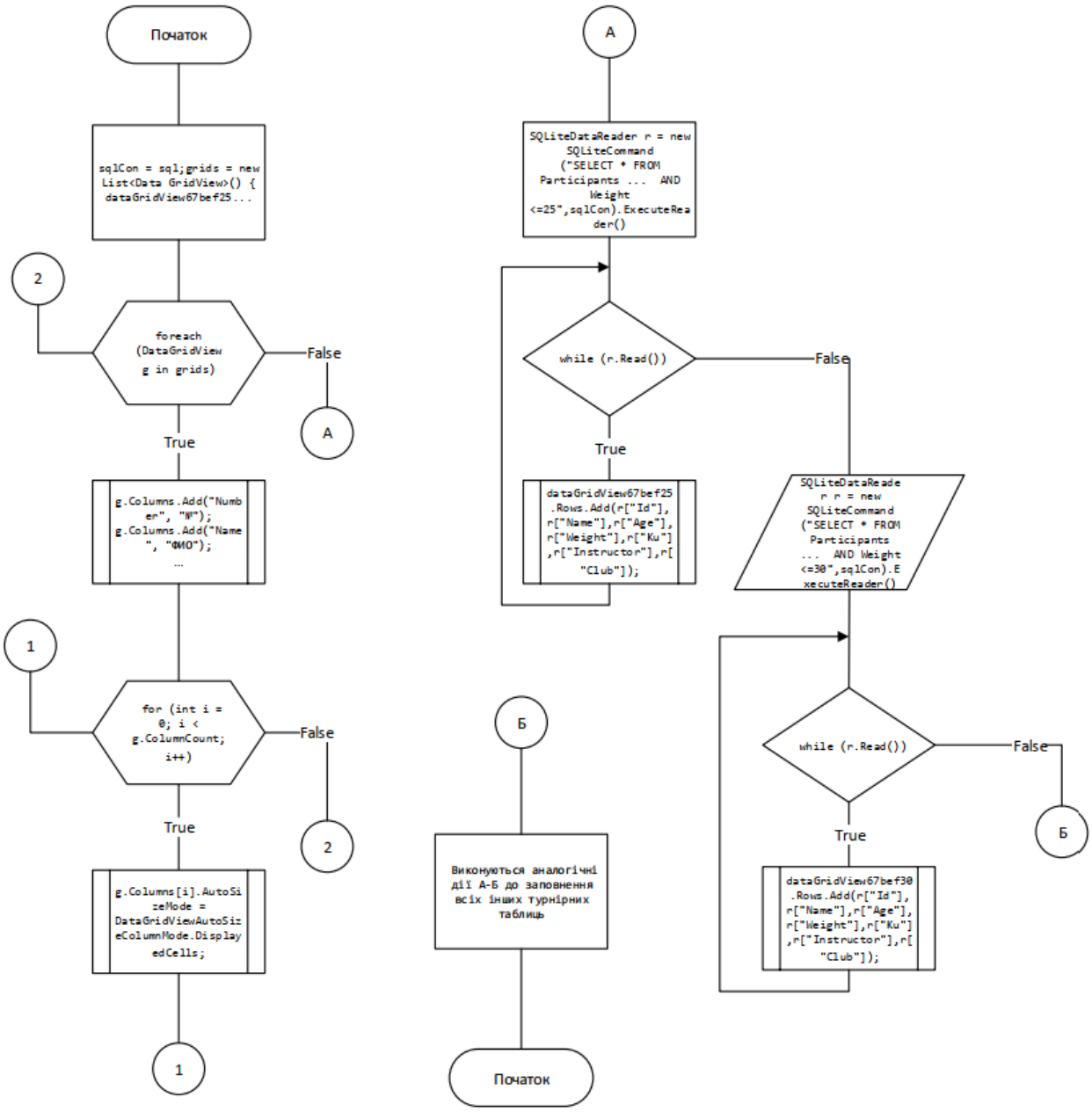

Рис. 2.11. Блок-схема процесу створення турнірних таблиць

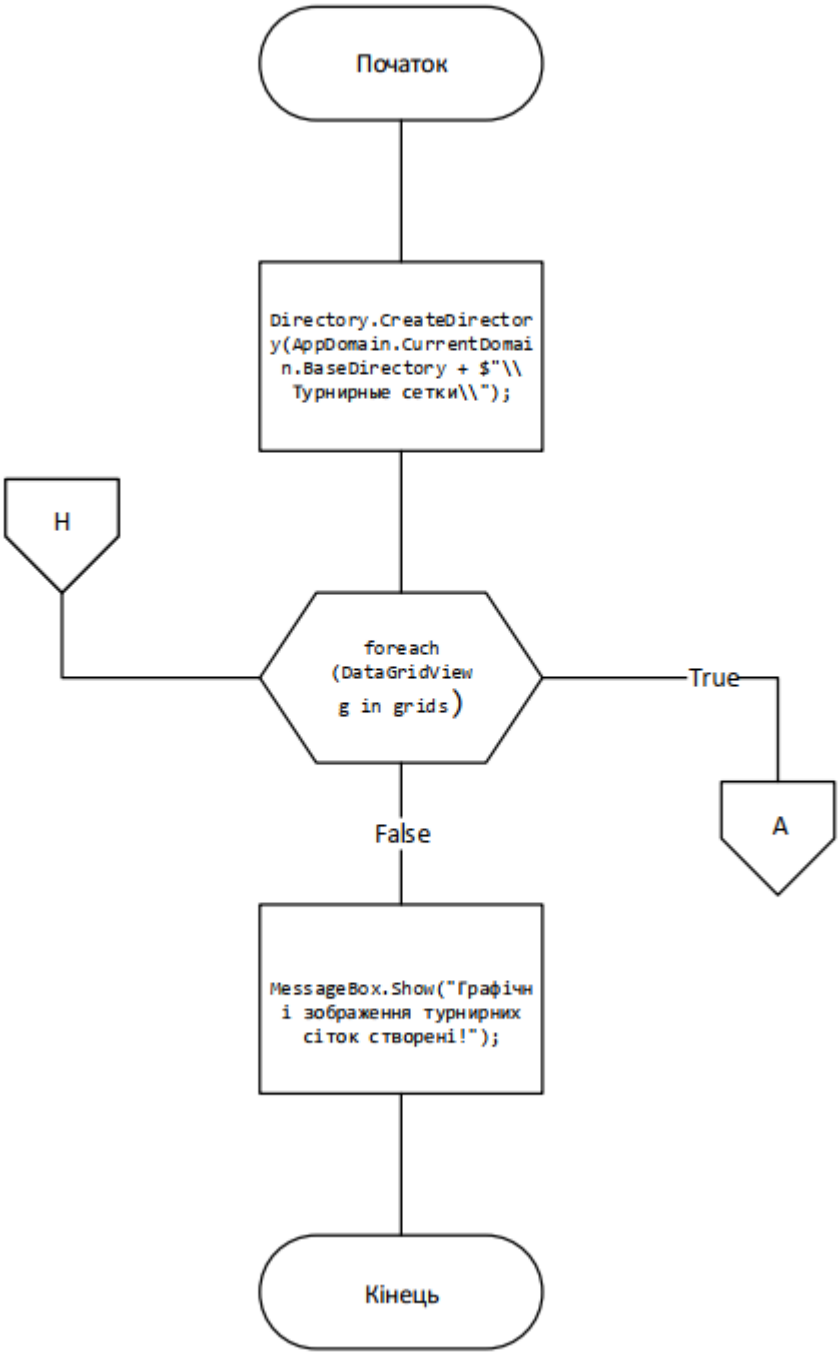

Рис. 2.12. Блок-схема процесу створення графічного зображення турнірних сіток

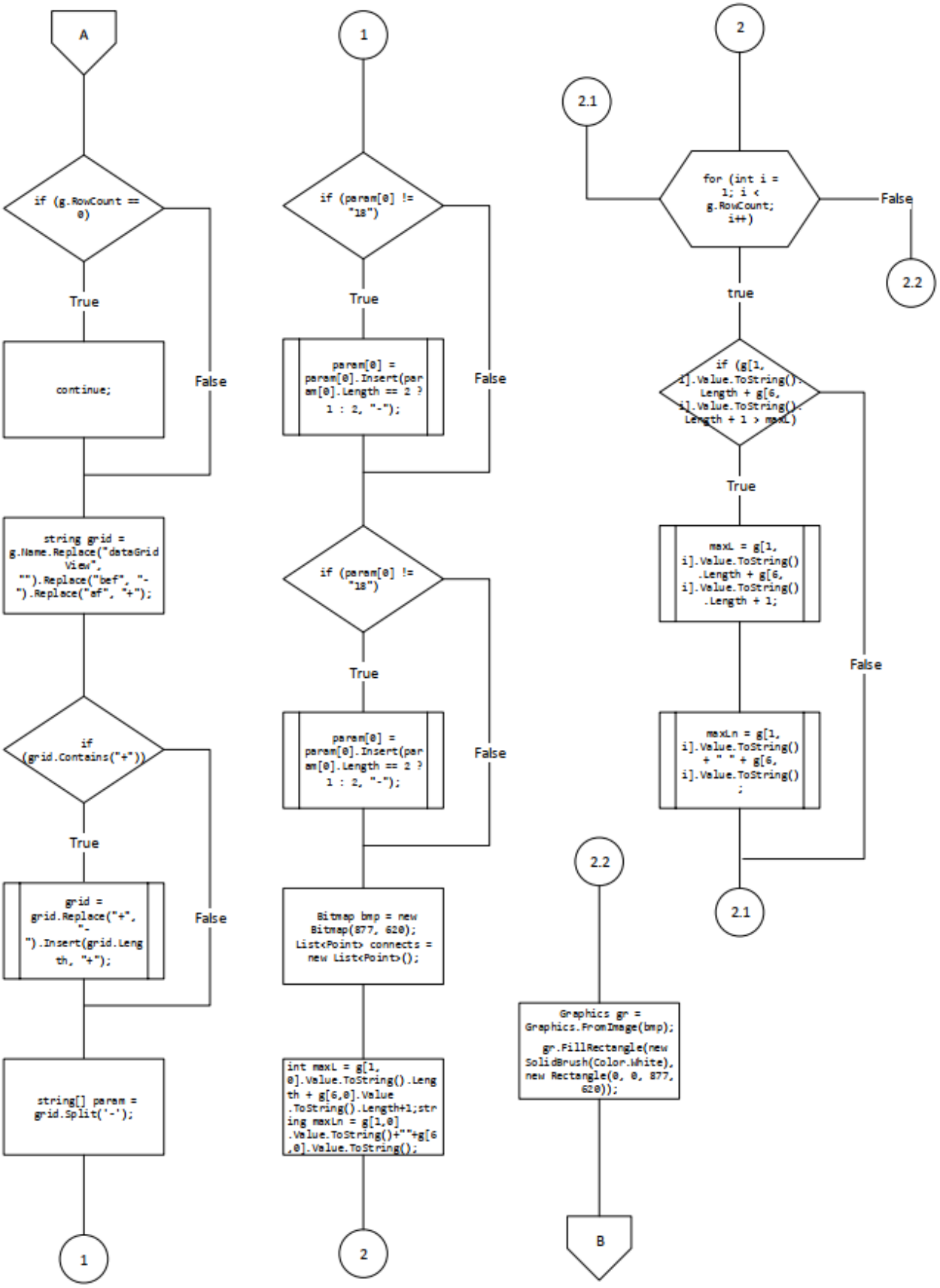

Рис. 2. 13. Продовження блок-схеми процесу створення графічного зображення турнірних сіток

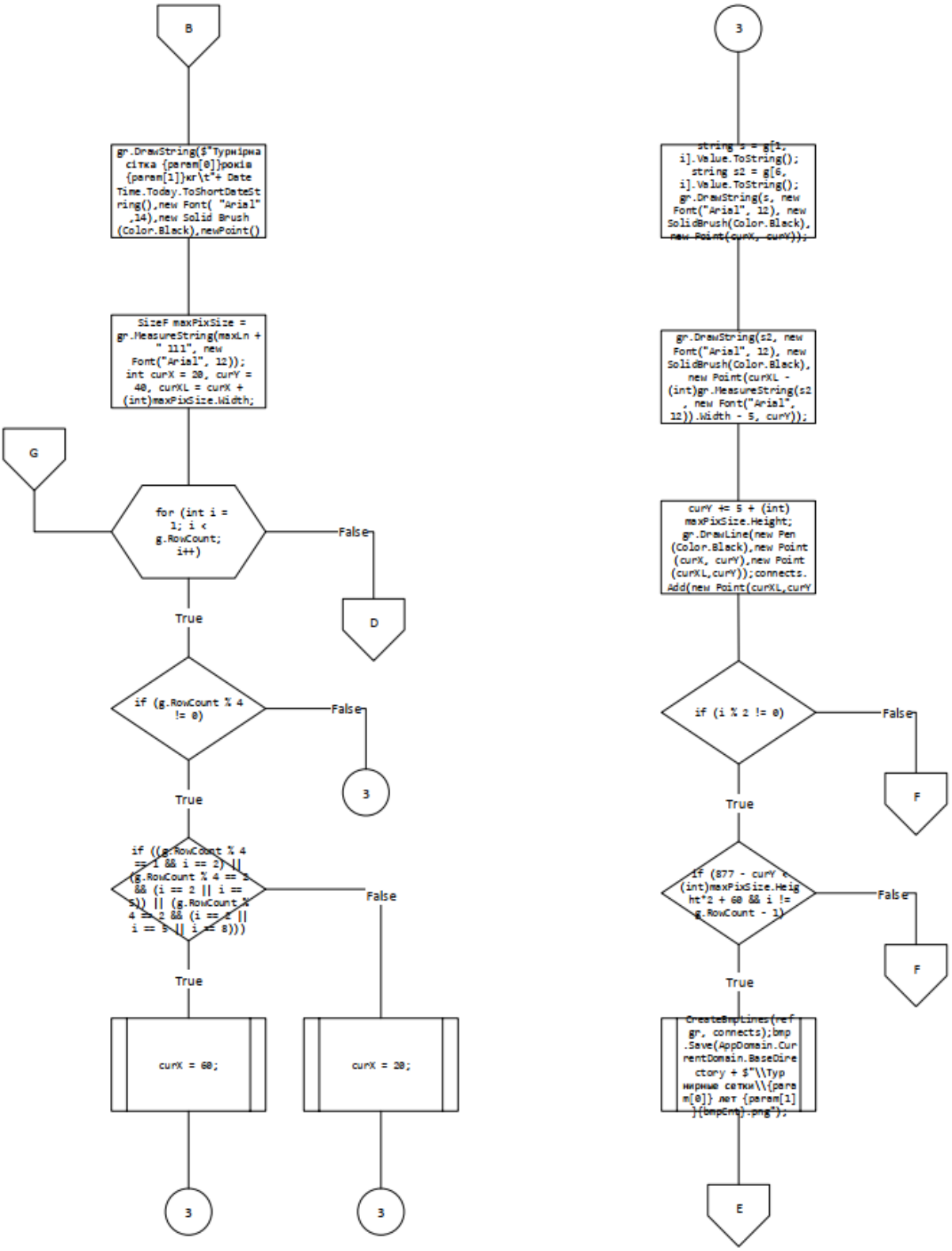

Рис. 2.14. Продовження блок-схеми процесу створення графічного зображення турнірних сіток

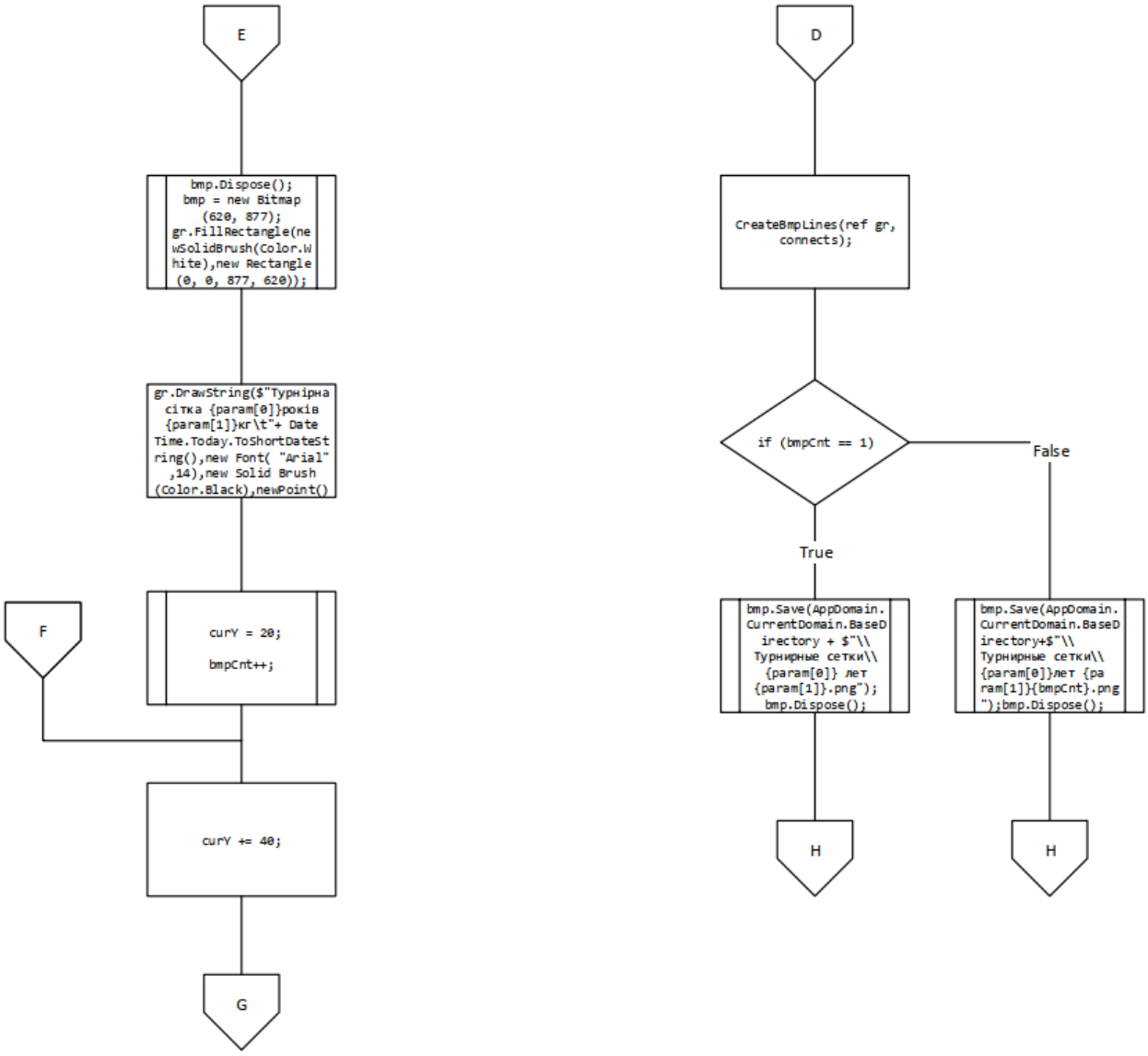

Рис. 2.15. Продовження блок-схеми процесу створення графічного зображення турнірних сіток

2.4.3. Розробка веб-сайту інформаційної системи

2.4.3.1. Файлова структура проекту веб-сайту

Як вже було сказано вище, формування переліку учасників змагань в інформаційній системі здійснюється за допомогою веб-сайту.

Розглянемо проектування веб-сайту, на якому тренери можуть додавати дані про учасників, неавторизовані користувачі переглядати дані про турнірні таблиці або сітки, а адміністратор має змогу додати завантажити графічні

зображення турнірної сітки які були створене раніше програмним додатком та змінити статус учасників для відображення їх в турнірній таблиці.

Проект складається з файлів, які перелічені на рис. 2.16:

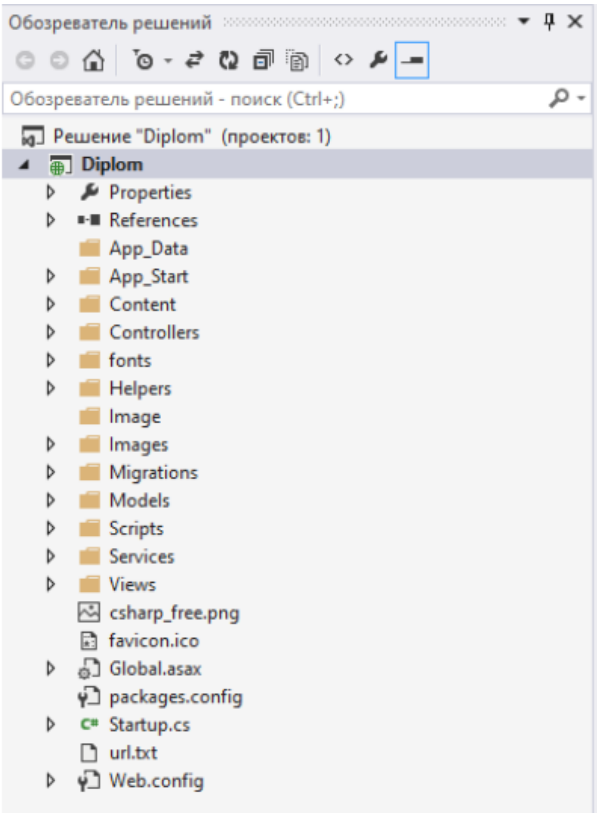

Рис. 2.16. Файлова структура проекту веб-сайту

При створенні проекту ASP.NET для розробки вебсайту середа Visual Studio надає кілька варіантів початкового вмісту, який потрібно для проекту. Ідея полягає в тому, щоб спростити процес вивчення розробникам-початківцям і застосувати заощаджуючи час рекомендовані прийоми для поширених засобів і завдань. Така підтримка втілена через шаблони, які використовуються для створення контролерів і уявлень, які створюються за допомогою шаблонного коду для виведення списків об'єктів даних, редагування властивостей моделі і т.д.

Середовище Visual Studio збирає проект, створений за допомогою шаблону MVC, використовуючи пакети NuGet , а це значить, що є можливість переглянути, які пакети застосовуються, вибравши пункт Manage NuGet

Packages for Solution (управління пакетами NuGet для вирішення) в меню Tools => Library Package Manager (Сервіс => Диспетчер бібліотечних пакетів). Це також означає можливість додавання тих же самих пакетів в будь-який проект, в тому числі і створений з використанням шаблону Empty.

Який би шаблон не був би обраний, все одно результуючі проекти мають дуже схожі структури папок. Деякі з елементів в проекті MVC грають особливі ролі, жорстко закодовані в ASP.NET або MVC Framework. Інші мають відношення до угод про іменування. Всі ці основні файли і папки описані в таблиці 2.17, причому деякі з них за замовчуванням в проектах не присутні [7].

Таблиця 2.19

Опис папок та файлі, які застосовані в проекті

Папка або файл	Опис
App_Data	Закриті дані (XML-файли чи БД, якщо використовується SQL ServExpress, SQLite або інші сховища на основі файлів
App_Start	Ця папка містить ряд основних налаштувань конфігурації для проекту, в тому числі визначення маршрутів і фільтрів, а також пакетів вмісту
Area	Області - це спосіб поділу додатку на менші час тини
bin	Сюди поміщається скомпільована збірка додатки MVC ра зом з усіма посилається збірками, що знаходяться не в GAC
Content	Поміщається статичний вміст: CSS-файли і зображення
Controllers	Сюди поміщаються класи контролерів
Models	Класи моделей уявлень і моделей предметної області, хоча все крім найпростіших додатків виграють від визначення моделі предметної області в окремому проекті
Scripts	для зберігання бібліотек JavaScript ,використовуваних в додатку. За замовчуванням Visual Studio додає бібліотеки jQuery і кілька інших популярних JavaScript-бібліотек
Views	У цій папці зберігаються уявлення і часткові уявлення, зазвичай згруповані разом в папках з іменами контролерів, з якими вони пов'язані
Views Shared	Зберігаються компонування та подання, які не є специфічними для будь-якого контролера

2.4.3.2. Розробка дизайну сайту

Розроблюваний веб-сайт містить 8 головних вікон. Ці вікна реалізуються за допомогою компонентів, опис яких наведений нижче.

Всі сторінки сайту мають рядок заголовку, де знаходяться посилання (кнопки) на інші сторінки сайту. Перелік посилань для звичайного користувача:

– логотип – посилається на компонент <Participants>;
– назва сайту – посилається на компонент <Participants>;
– турнірні таблиці – посилається на компонент <Draws>;
– турнірні сітки - посилається на компонент <TournamentGrid>;
– реєстрація - посилається на компонент <Account/Register>;
– логін - посилається на компонент <Account/Login>.

Перелік посилань для авторизованого користувача включає в себе вище перелічені посилання з деякими змінами:

– мої учасники – посилається на компонент <MyParticipants>;
– мій профіль – посилається на компонент <Profile/Detail>;
– вихід – посилається на компонент <Participants>.

Для користувача, який має права адміністратора, додається посилання Адмін панель, яка посилається на компонент <Admin>.

<Participants/> – являється головною сторінкою та містить опис всіх зареєстрованих учасників та назву змагання, ескіз зображений на рис. 2.17.

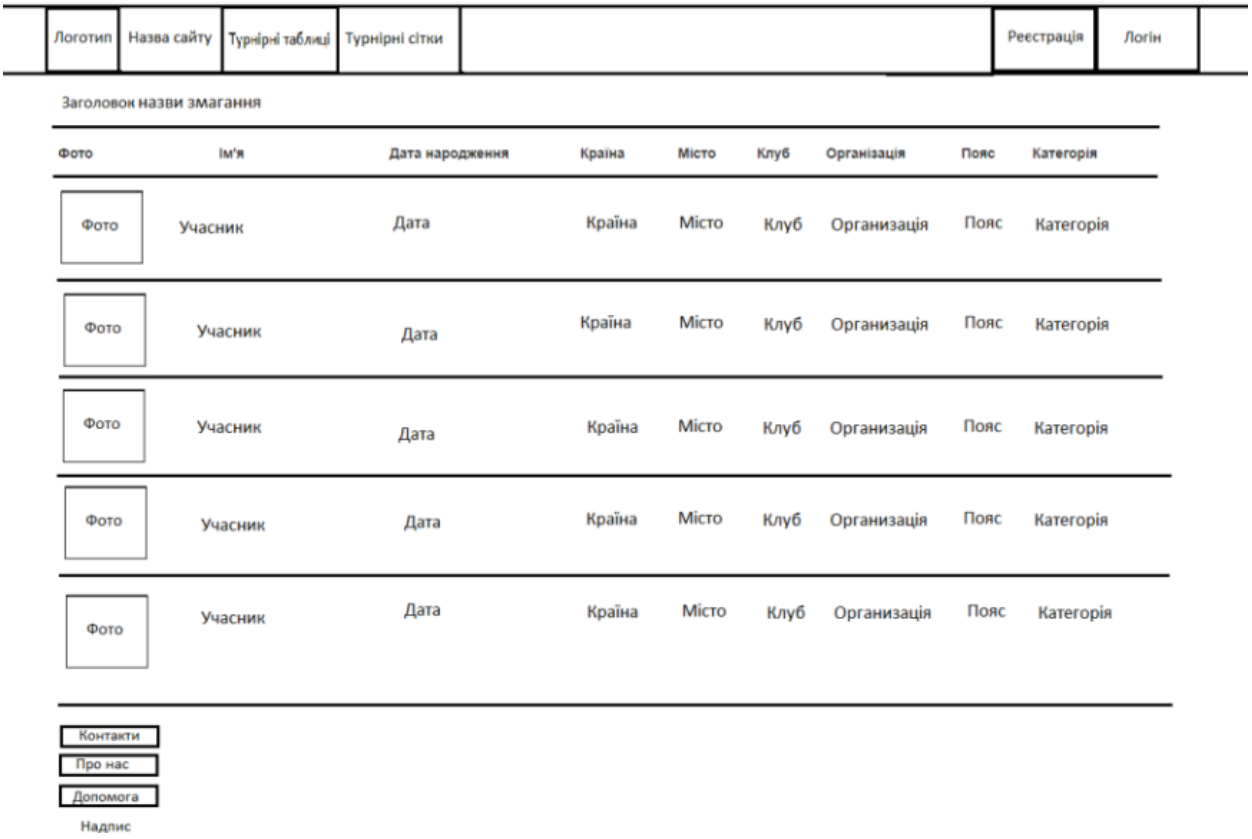

Рис. 2.17. Ескіз головного вікна (звичайний користувач)

В авторизованого користувача (тренера), у рядку головного меню з’являється кнопка «Мої учасники» і змінюються кнопки «Реєстрація» та «Логін» на «Мій профіль» та «Вихід» відповідно, ескіз зображений на рис. 2.18.

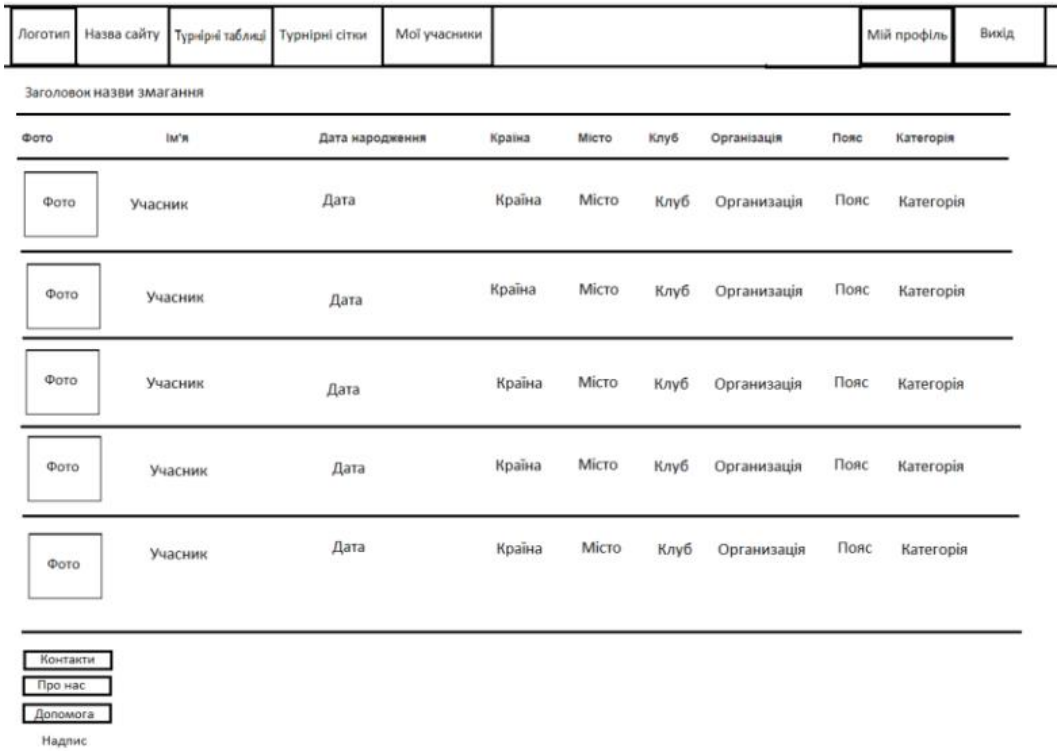

Рис. 2.18. Ескіз головного вікна (авторизований користувач)

<MyParticipants> – містить опис зареєстрованих учасників авторизованим користувачем, ескіз зображений на рис. 2.19.

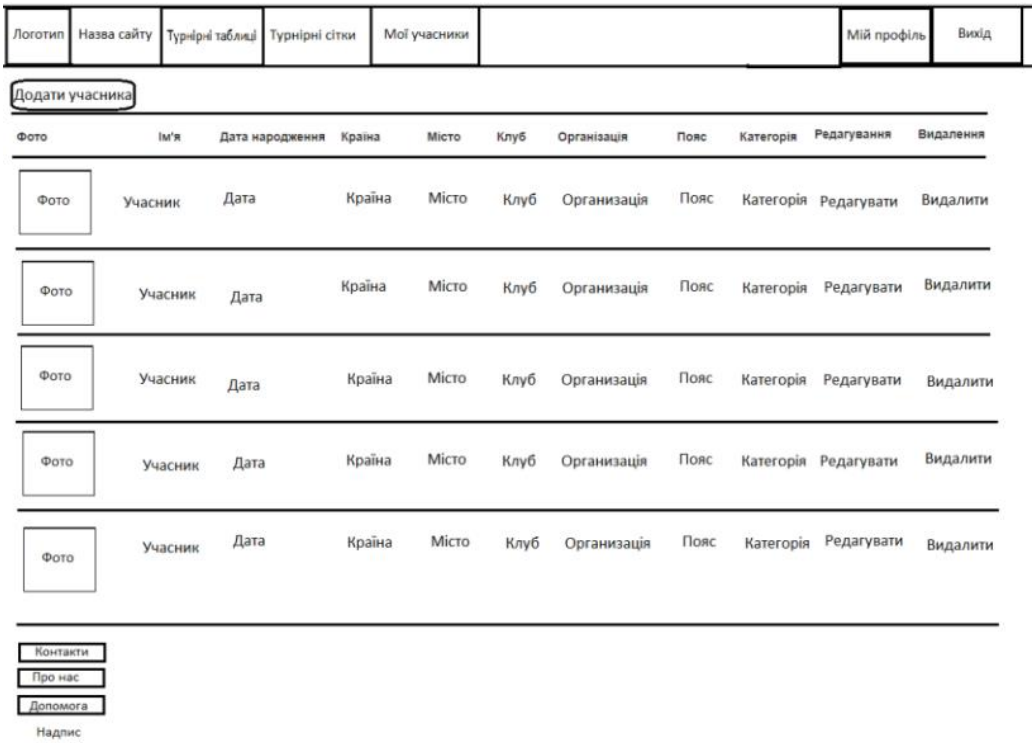

Рис. 2.19. Ескіз вікна «Мої учасники»

<Draws/Index/> – містить таблицю опису учасників по конкретній категорії, ескіз зображений на рис. 2.20.

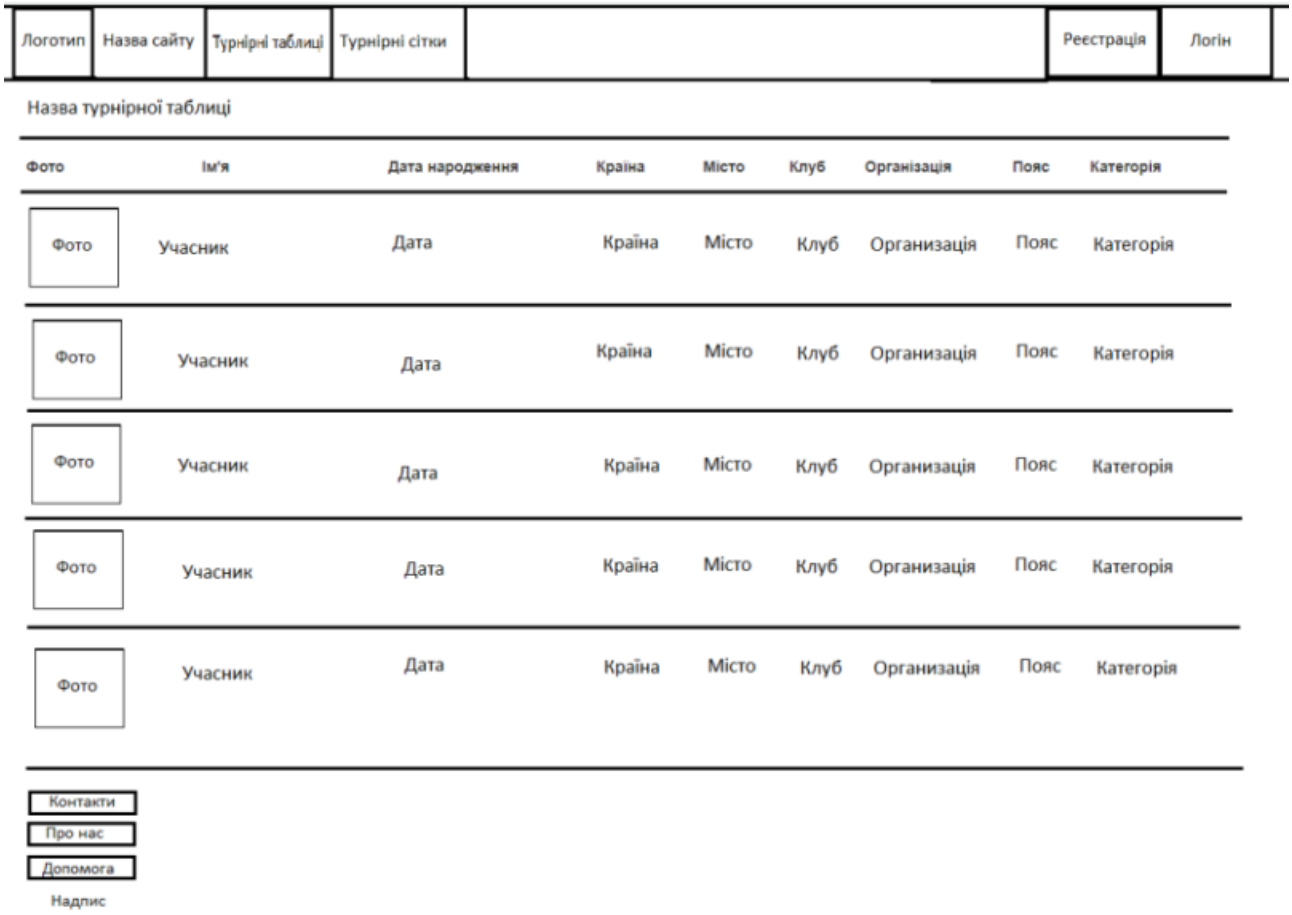

Рис. 2.20. Ескіз вікна «Турнірна таблиця»

<Draws/Index/> – містить графічне зображення турнірної сітки по конкретній категорії, ескіз зображений на рис. 2.21.

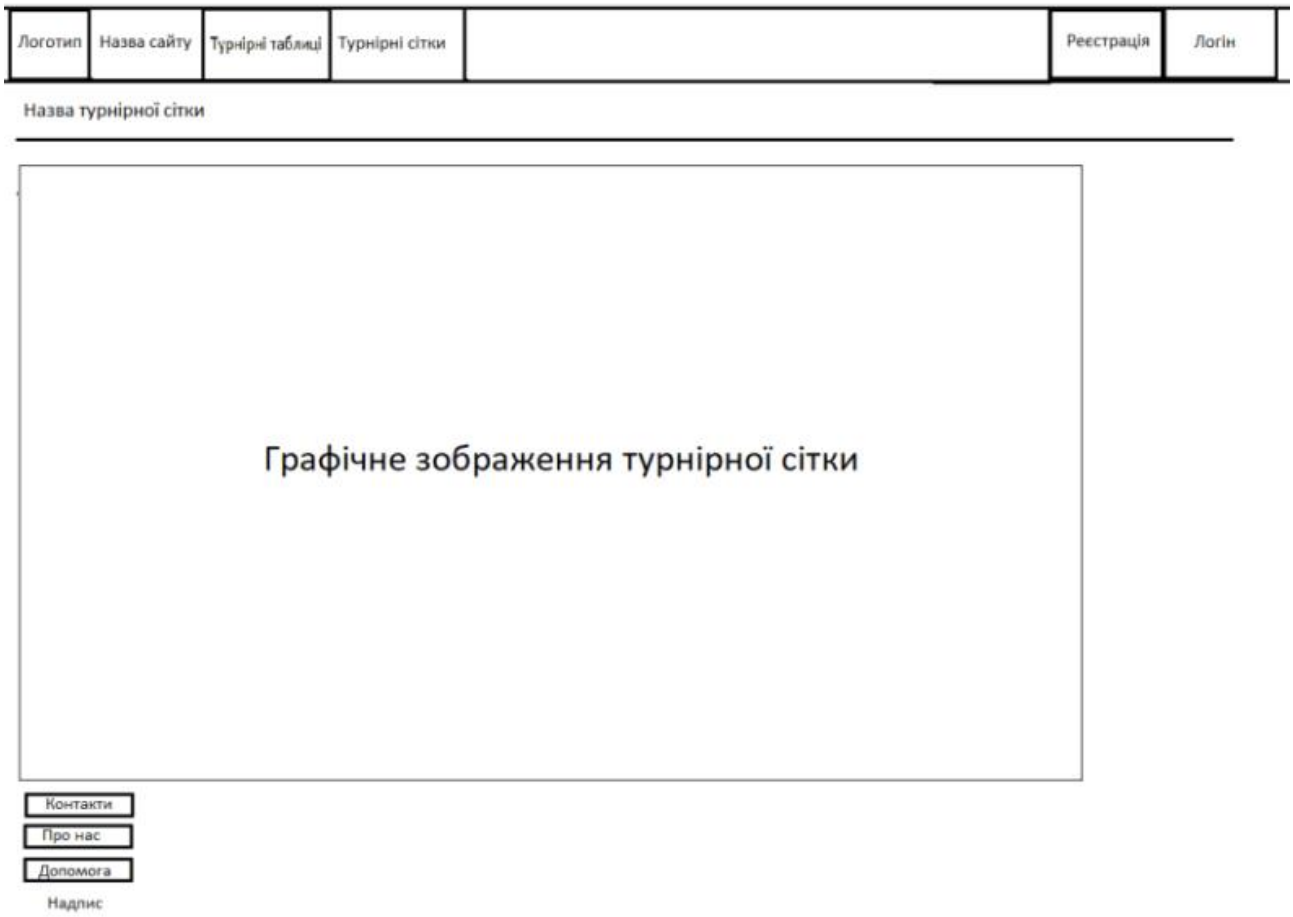

Рис. 2.21. Ескіз вікна «Турнірна сітка»

<Profile/Detail/> – містить інформацію про зареєстрованих користувачів, ескіз зображений на рис. 2.22.

<Acount/Register/> – містить поля для реєстрації користувачів, ескіз зображений на рис. 2.23.

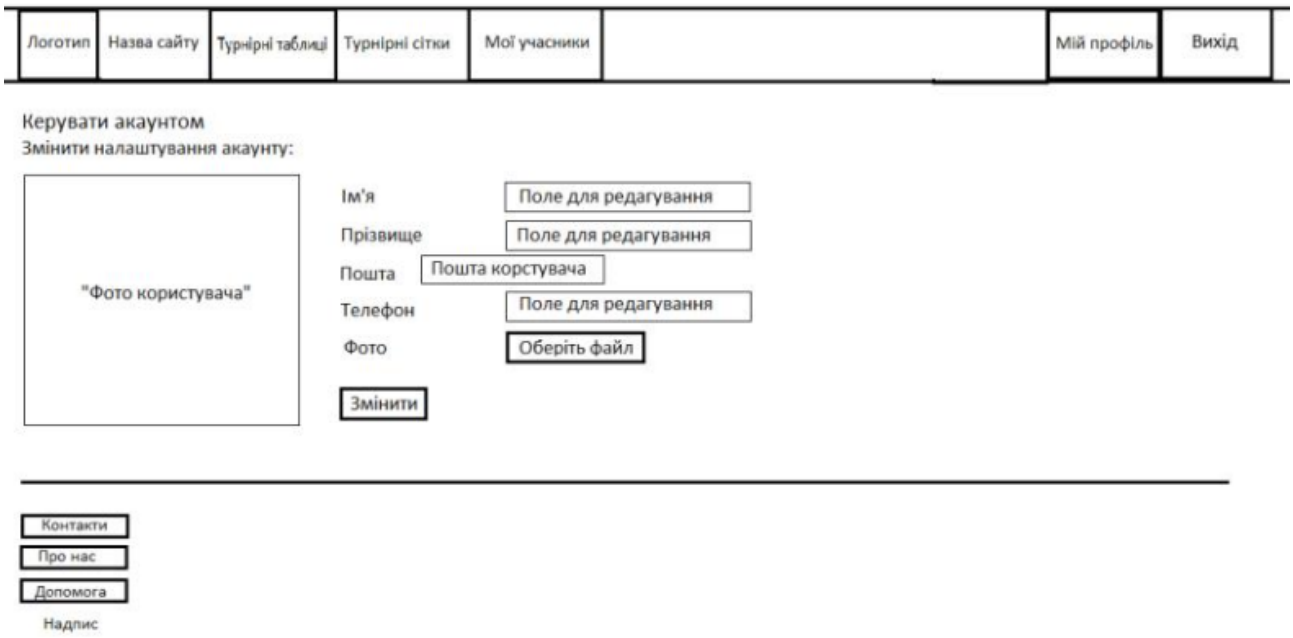

Рис. 2.22. Ескіз вікна «Мій профіль»

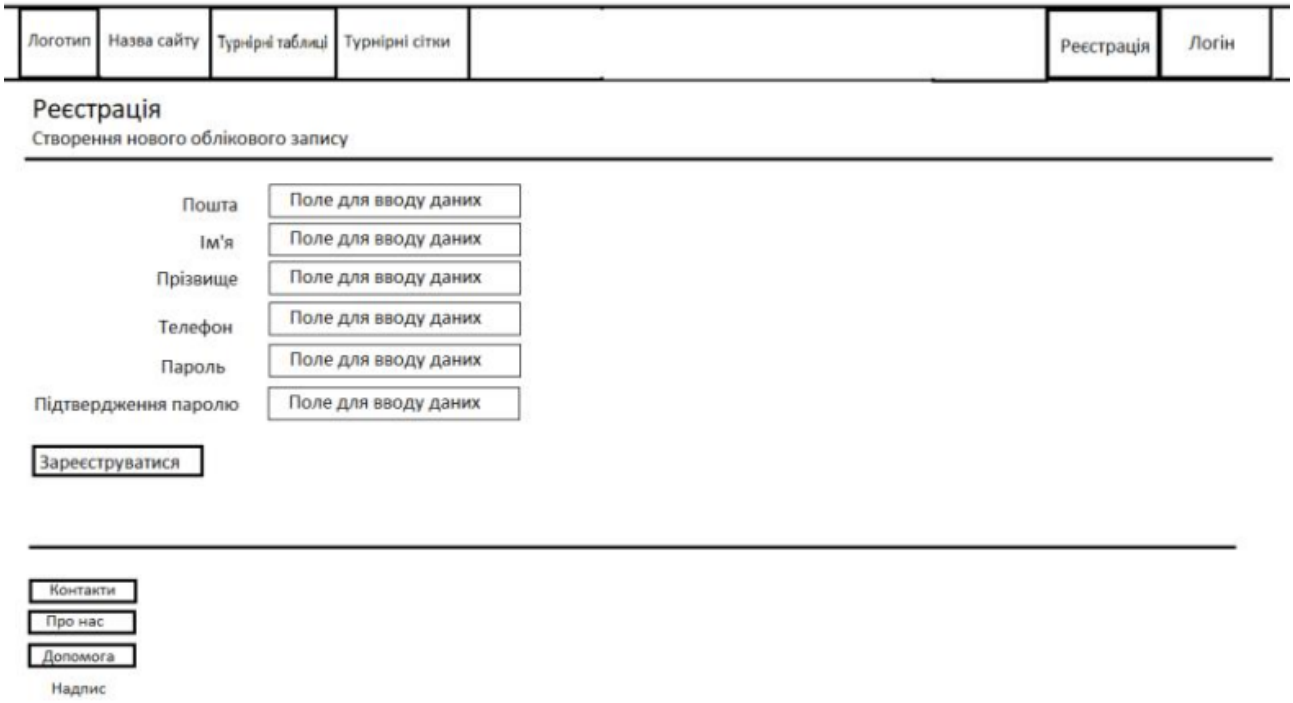

Рис. 2.23. Ескіз вікна «Реєстрація»

<Acount/Login/> – містить поля для реєстрації користувачів, ескіз зображений на рис. 2.24

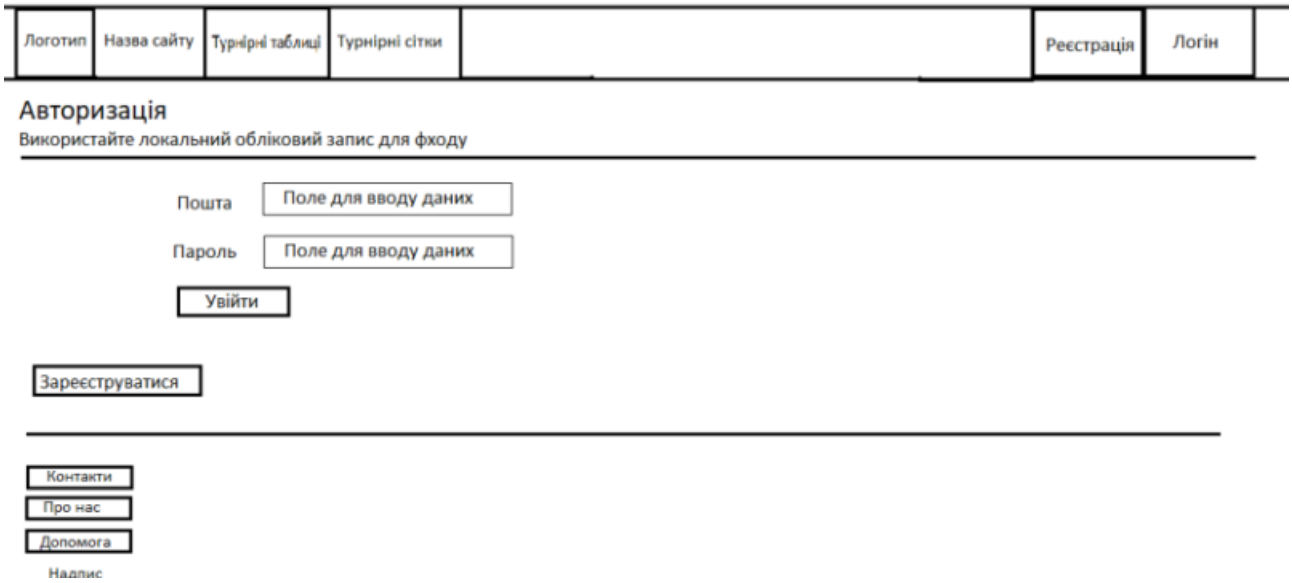

Рис. 2.24. Ескіз вікна «Авторизація»

<Admin/> – містить панель функції адміністратора, ескіз зобра-жений на рис. 2.25.

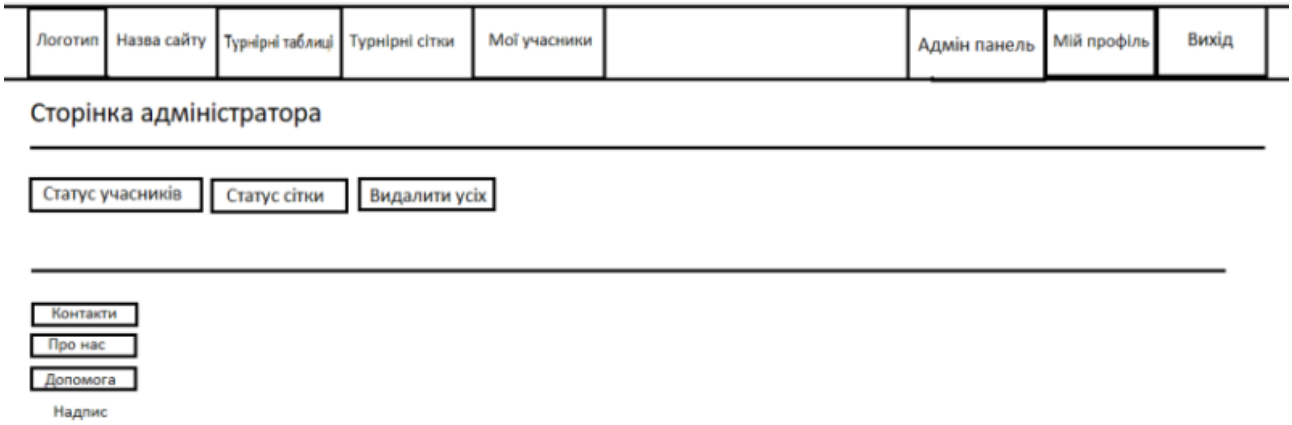

Рис. 2.25. Ескіз вікна «Сторінка адміністратора»

2.5. Обґрунтування та організація вхідних та вихідних даних програми

Вхідними даними системи виступають особисті дані про учасників змагань, які будуть додаватися до системи тренерами (авторизованими користувачами) на веб-сайті та дані для їхньої авторизації:

– від адміністратора: ідентифікатор, логін, пароль;

– від учасника: ідентифікатор, ім’я, дата народження, вік, вага, ступінь, тренер, клуб.

В якості вихідних даних виступати сформовані турнірні таблиці зареєстрованих на змагання учасників, сформовані турнірні таблиці та сітки по кожній категорії, які розташовані на сайті, побудовані турнірні сітки змагань та готові запити до бази даних:

– отримання списку всіх учасників;

– отримання списків учасників по категоріям.

2.6. Опис роботи розробленої системи

2.6.1. Використані технічні засоби

Для розробки даної програми використано ПК з наступними характеристиками:

- тип процесора: процесор з частотою 2.2 ГГц;
- ОЗУ об'ємом 4 Гб;
- 30 Мб доступного простору на жорсткому диску;
- жорсткий диск з частотою обертання 5400 об / хв.;
- дозвіл екрану 1024x768;
- клавіатура;
- маніпулятор «миша».

2.6.2. Використані програмні засоби

Для виконання даної роботи були застосовані наступні засоби розробки: середовище програмування Visual Studio 2017; мова програмування C#; конфігурація Windows Froms (для програмного додатку); СКБД SQLite (для програмного додатку); платформа ASP.NET MVC5 (для веб-сайту); СКБД MS SQL Server (для веб-сайту).

2.6.3. Виклик та завантаження програми

Програма написана за допомогою методів об’єктного-подібного програмування на мові C# та скомпільована з розширенням exe. Вона може працювати у операційних системах Windows та UNIX-подібних ОС. В останній операційній системі exe-файл можна запустити за допомогою безкоштовного емулятора під назвою WINE.

2.6.4. Опис інтерфейсу користувача

Розроблюваний програмний продукт складається з двох частин, а саме веб-сайту для формування переліку учасників змагань та додатку, який формує турнірні таблиці та сітки для спортивних змагань.

Щоб створити турнірні таблиці та сітки, треба сформувати загальну таблицю учасників. Формування таблиці здійснюється на підставі внесення даних про учасників тренерами засобами веб-сайту.

Запустивши вебсайт користувач потрапляє на головну сторінку, де можна переглянути перелік всіх учасників змагань, тренери яких подали вже заявку на змагання (рис. 2.26).

Щоб додати дані про учасників змагань до системи, потрібно авторизуватися на сайті, для цього необхідно натиснути на кнопку «Логін» у рядку головного меню та ввести адресу електронної пошти та пароль для акаунту (рис. 2.27).

Якщо ж користувач неавторизований, тоді йому потрібно зареєструватися (рис. 2.28), натиснувши на кнопку «Реєстрація» в рядку головного меню. Для реєстрації, користувачу необхідно ввести адресу своєї поштової скриньки, ім’я, прізвище, номер телефону та складний пароль.

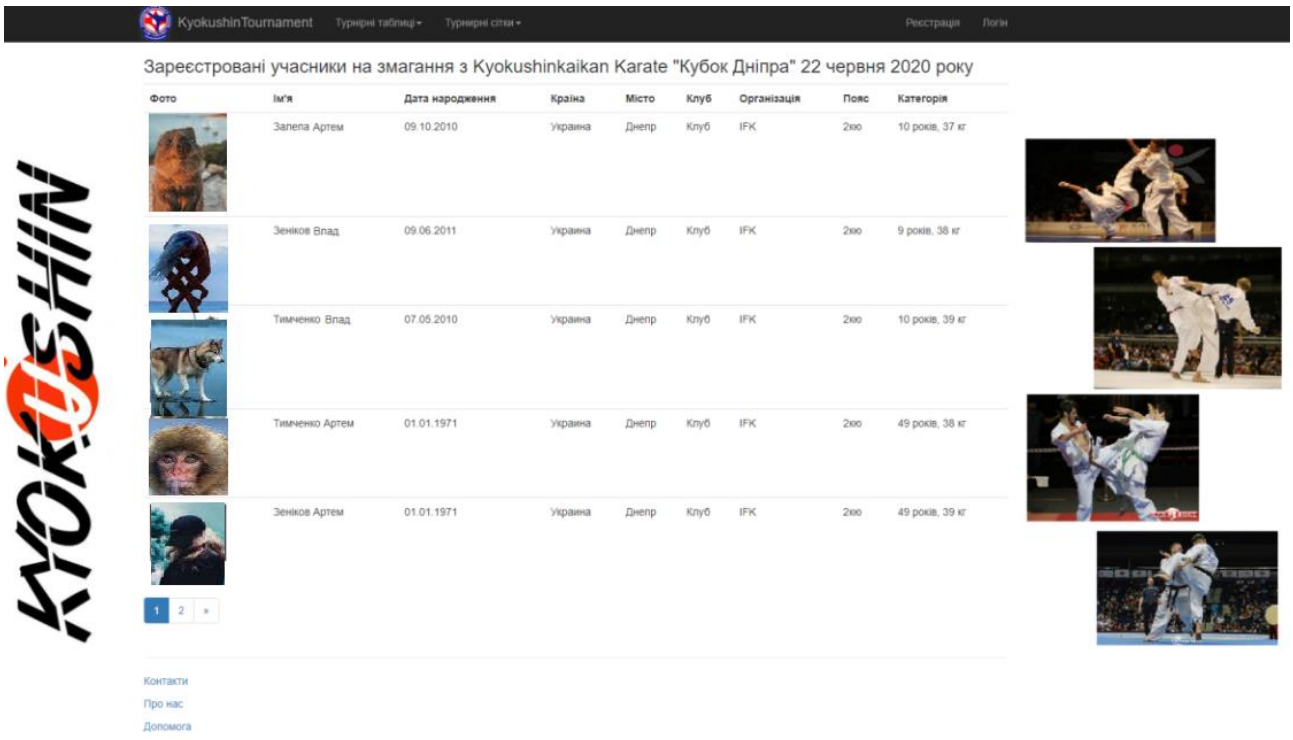

Рис. 2.26. Головна сторінка сайту

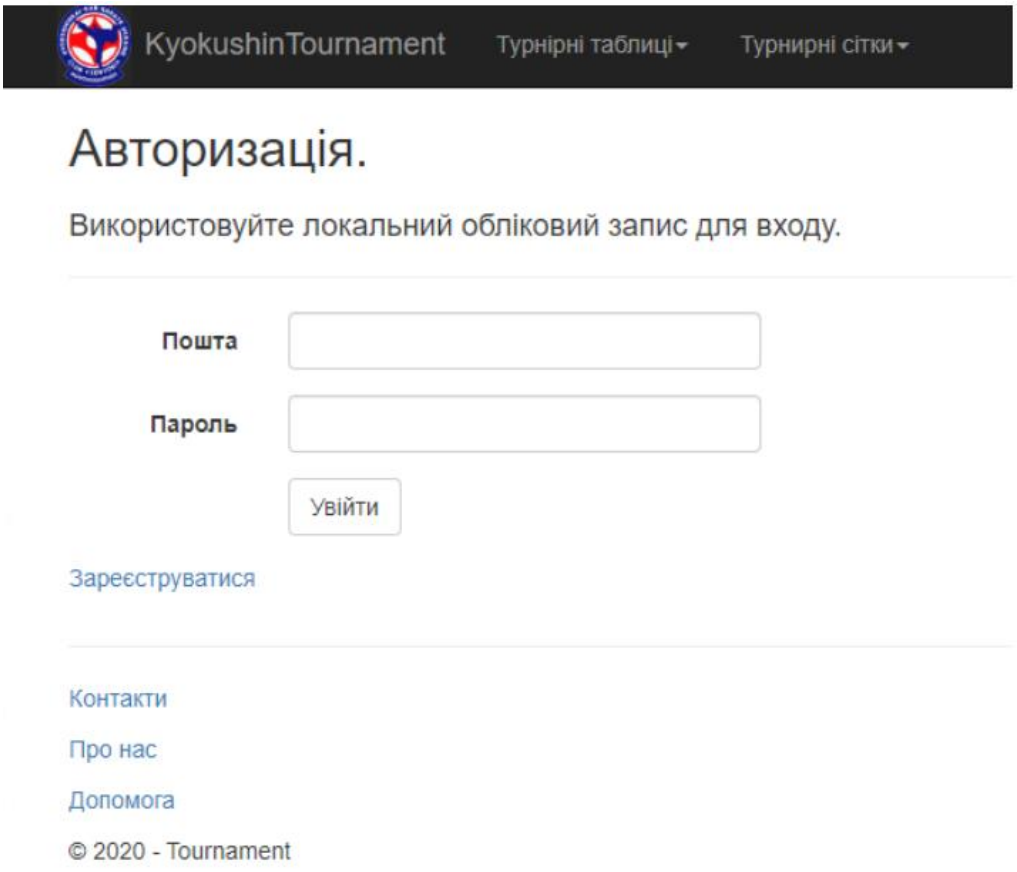

Рис. 2.27. Сторінка авторизації тренера на сайті

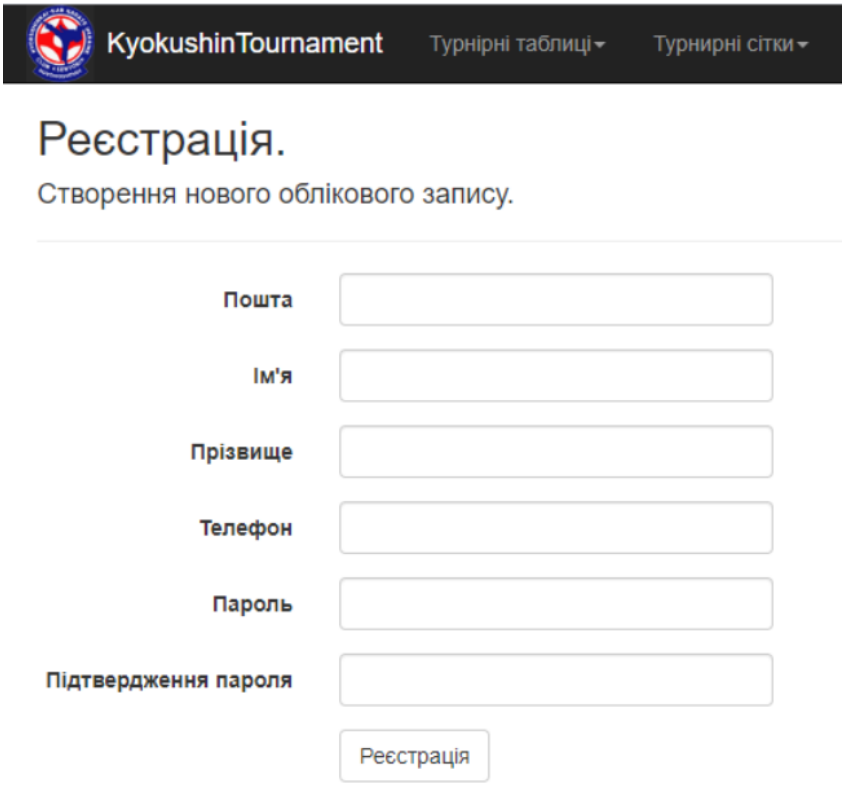

Рис. 2.28. Сторінка реєстрації тренера на сайті

Після авторизації користувач має змогу додавати дані про учасників змагань, для цього йому необхідно натиснути на кнопку «Мої учасники» у рядку головного меню (рис. 2.29).

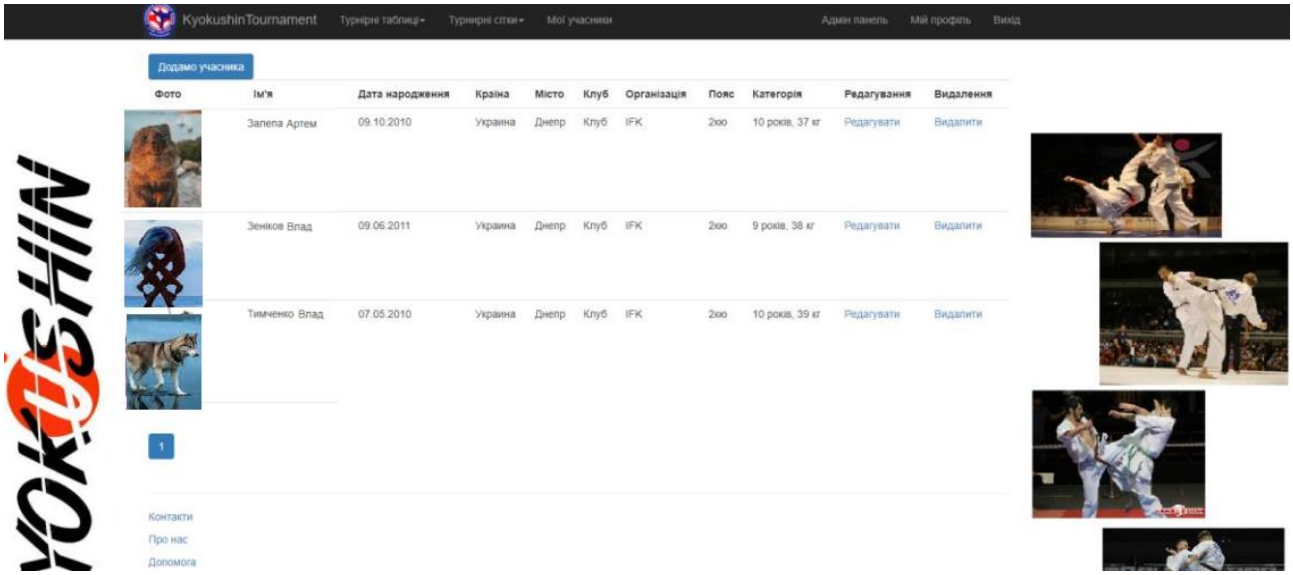

Рис. 2.29. Сторінка перегляду «Мої учасники» сайту

Щоб додати дані про учасника, потрібно натиснути на кнопку «Додамо учасника» і тренер перейде на сторінку додавання учасників (рис. 2.30). Тренеру потрібно ввести наступні дані про учасника: прізвище та ім’я, дату народження, країну, місто, клуб, пояс, вагу та фото. Після чого натиснути на кнопку «Створити».

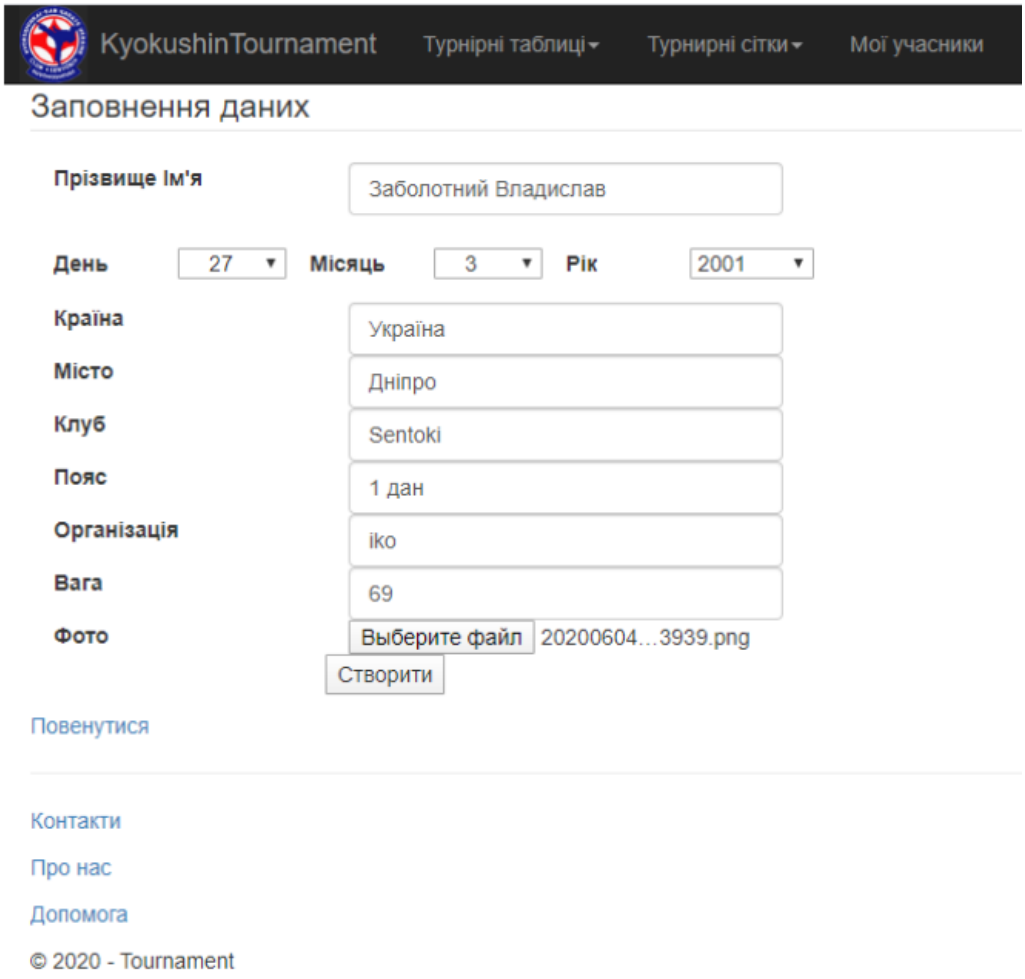

Рис. 2.30. Сторінка для додавання даних про учасників змагань

Після введення тренером даних необхідно натиснути на кнопку «Створити» і він відразу перейде на сторінку перегляду доданих ним учасників і зможе переглянути результат додавання (рис. 2.31).

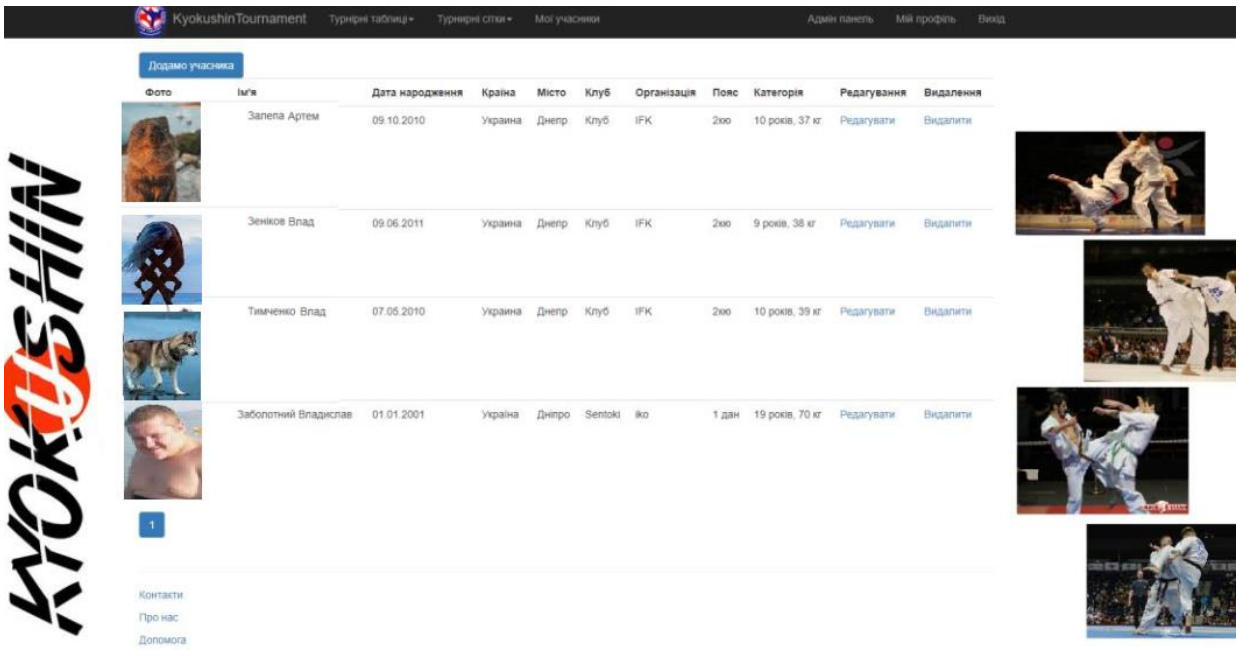

Рис. 2.31. Сторінка сайту з доданими учасниками тренером

В авторизованого користувача є також можливість редагувати дані про учасника, для цього йому потрібно натиснути на кнопку «Редагувати» навпроти учасника, після чого відкриється відповідне вікно «Редагування даних», де можна у відповідних полях замінити потрібні дані та натиснути «Зберегти».

Для того щоб видалити дані про учасника, потрібно натиснути на кнопку «Видалити» навпроти потрібного учасника, після чого тренер перейде на сторінку для видалення учасника (рис. 2.32).

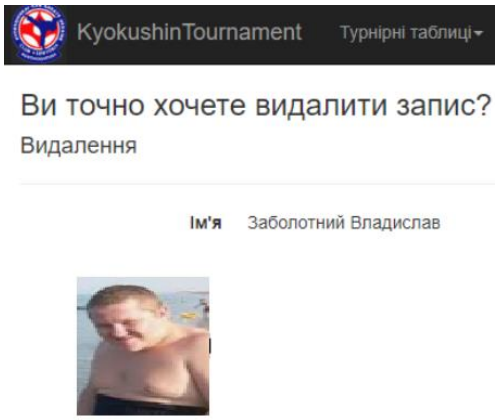

Рис. 2.32. Сторінка для видалення запису про учасника змагань

Слід зазначити, що адміністратор має адмін панель (рис. 2.33), через яку він може додавати турнірні сітки та змінювати статус учасників для турнірних таблиць. Також він має можливість видалити записи про всіх учасників, це потрібно тоді коли закінчились змагання.

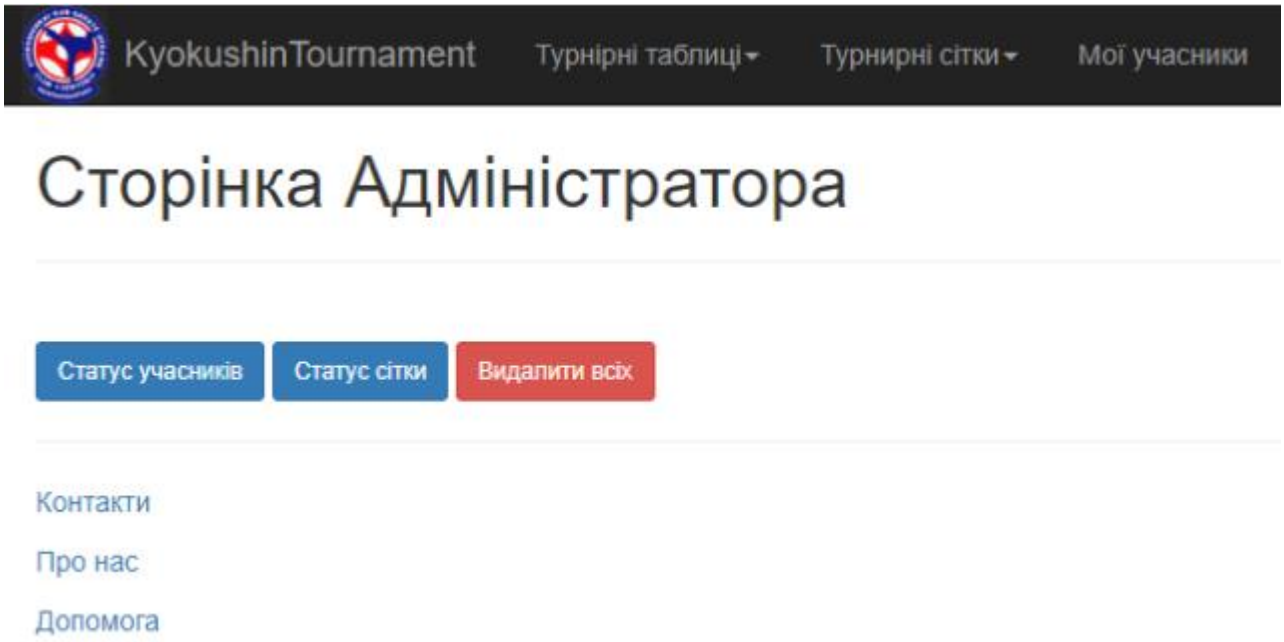

Рис. 2.33. Сторінка адміністратора для додавання турнірної сітки

Для того щоб змінити статус учасників адміністратору необхідно натиснути на відповідну кнопку «Статус учасників» на сторінці адміністратора. Користувач, який має адміністратора потрапляє на сторінку «Зміна статусу учасників» (рис. 2.34). Для зміни статусу необхідно натиснути на кнопку «Змінити» навпроти потрібного запису.

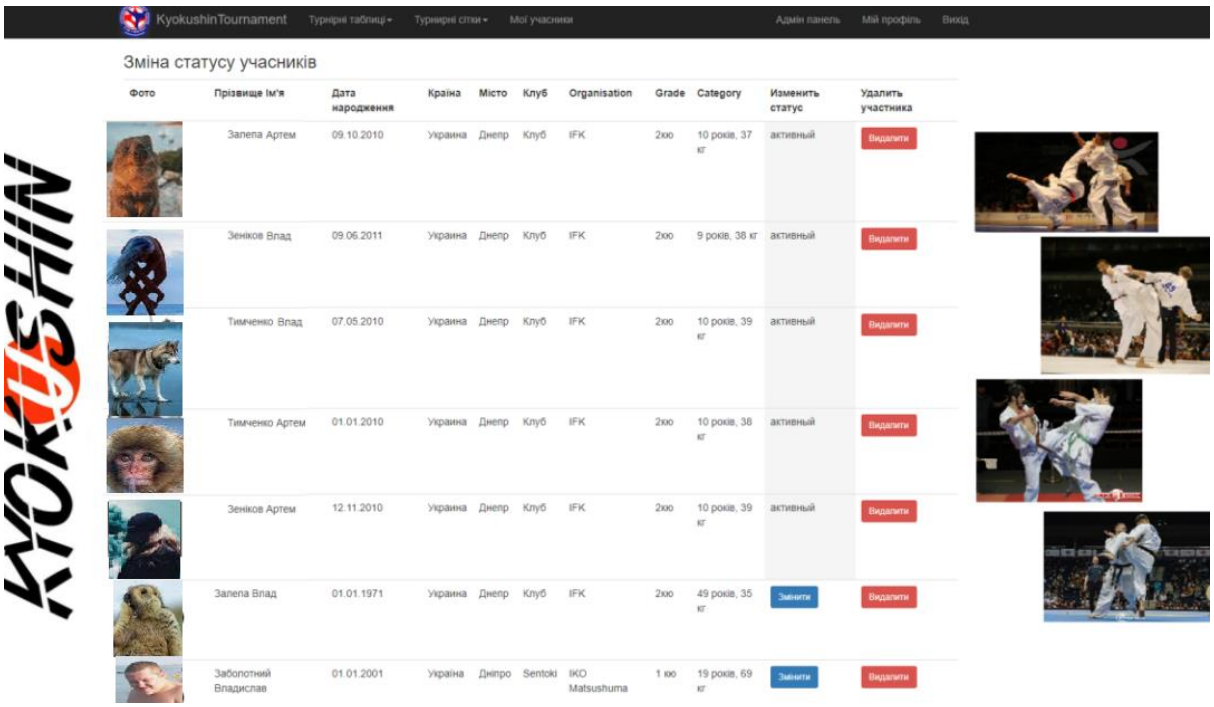

Рис. 2.34. Сторінка «Зміна статусу учасників» сайту

Після внесених змін статусу учасника, він потрапляє у відповідну турнірну таблицю в залежності від його ваги та віку (рис. 2.35). Турнірну таблицю може переглянути будь-який користувач, натиснувши на кнопку «Турнірна таблиця» та обравши потрібну категорію.

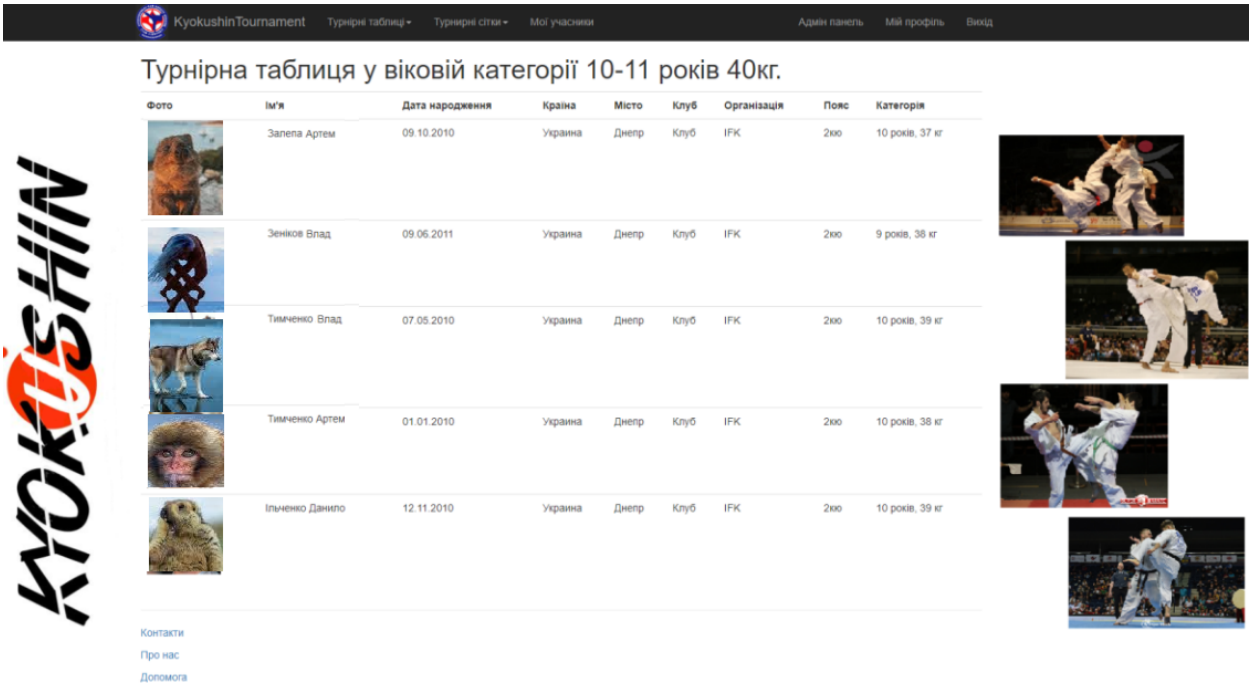

Рис. 2.35. Сторінка сайту для перегляду турнірної таблиці

Щоб додати графічне зображення турнірної сітки потрібно натиснути на кнопку «Статус сітки» на сторінці адміністратора, яка була приведена вище. Після чого адміністратор потрапляє на сторінку «Турнірні сітки», де він має змогу завантажити графічне зображення турнірної сітки та змінити її статус для того, щоб будь-які користувачі мали змогу переглянути її (рис. 2.36).

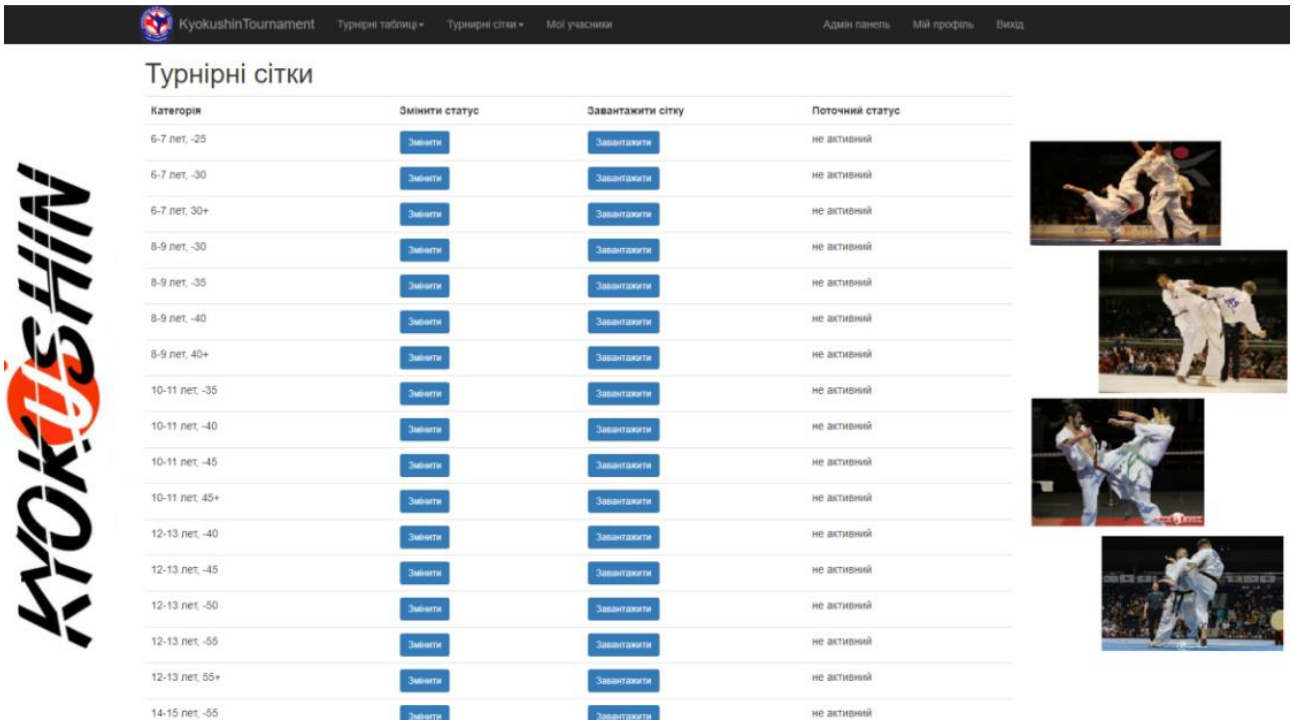

Рис. 2.36. Сторінка сайту для завантаження турнірних сіток

Після того, як адміністратор натиснув на кнопку «Завантажити» навпроти потрібної турнірної сітки, йому пропонується обрати зображення турнірної сітки, яке було сформоване раніше (рис. 2.37).

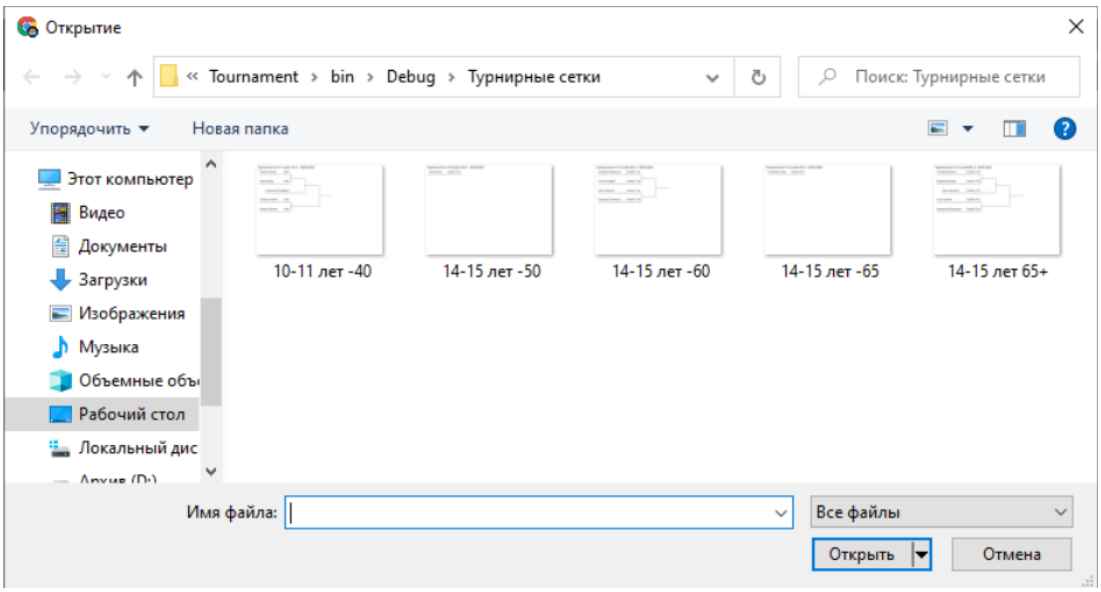

Рис. 2.37. Діалогове вікно для додавання турнірної сітки

Після того, як адміністратор додав зображення турнірної сітки та змінив її статус на «активний», тоді будь-який користувач сайту може переглянути її натиснувши на кнопку «турнірні сітки» та обравши потрібну категорію (див. рис. 2.38).

Рис. 2.38. Турнірна сітка для учасників змагань у графічному вигляді

Якщо у користувача веб-сайту виникли запитання щодо користування ним, він може у нижній частині будь-якого вікна веб-сайту знайти додаткову інформацію з приводу користування ним, натиснувши на посилання «Допомога» (рис. 2.39).

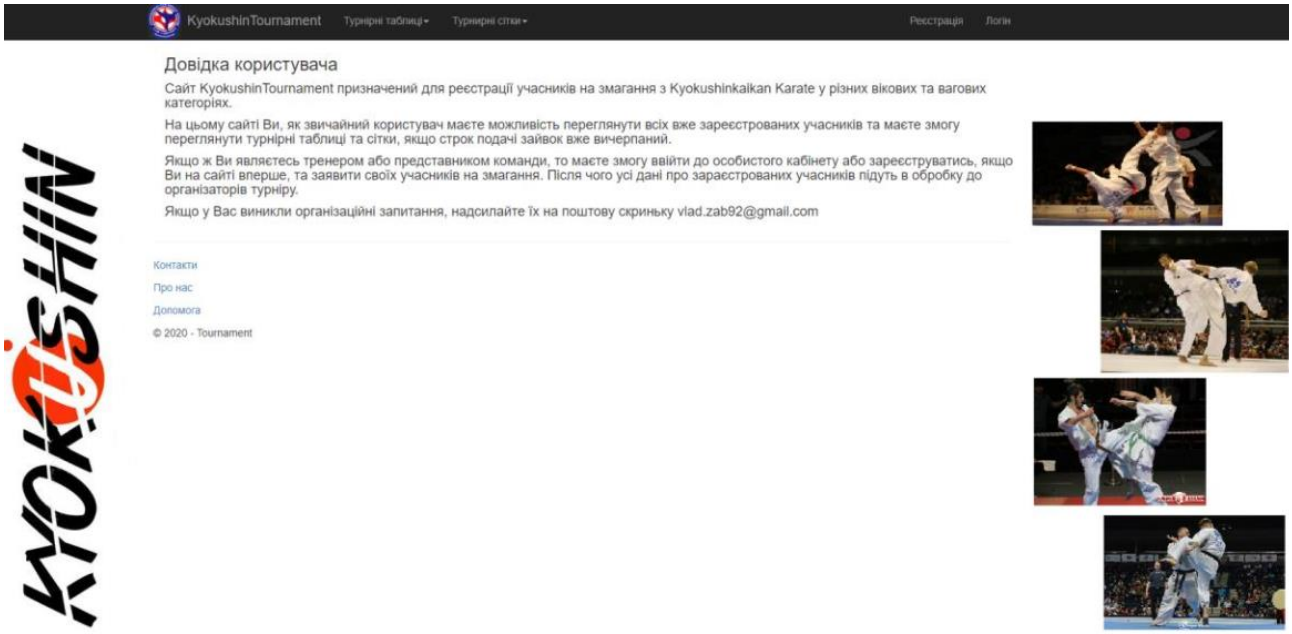

Рис. 2.39. Сторінка сайту «Довідка користувача»

Після внесення даних про учасників засобами веб-сайту, адміністратор має можливість завантажити дані до локальної бази даних програмного додатку та сформувати турнірні таблиці та сітки. Для цього потрібно запустити програмний додаток Kyokushin Tournament та авторизуватися (рис. 2.40).

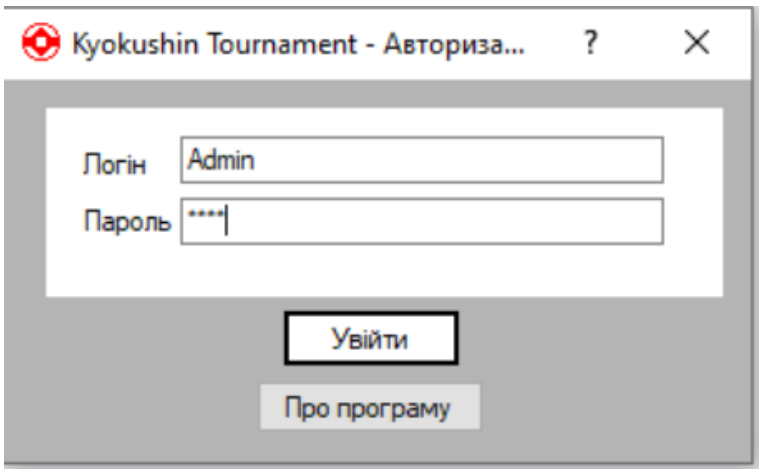

Рис. 2.40. Вікно для авторизації користувача у додатку

Якщо введений користувачем пароль не вірний, з’явиться відповідне повідомлення (рис. 2.41).

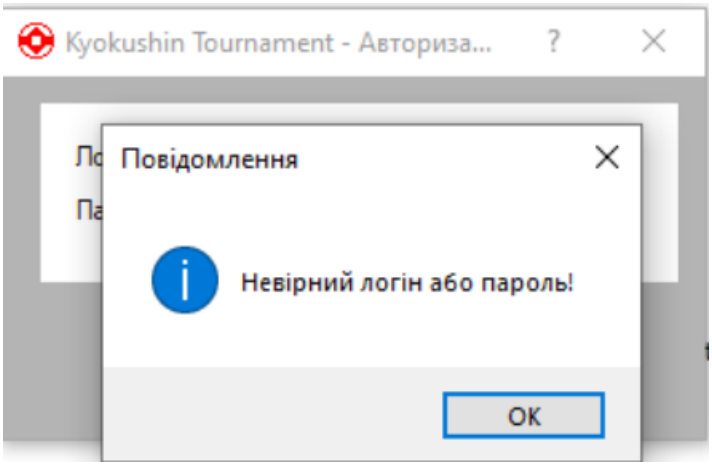

Рис. 2.41. Повідомлення про невірний логін або пароль при авторизації користувача

Слід зазначити, що кожне вікно додатку має кнопку «?» у рядку заголовку, натиснувши на яку користувач може отримати інструкцію по роботі з програмним додатком у даному вікні (рис. 2.42).

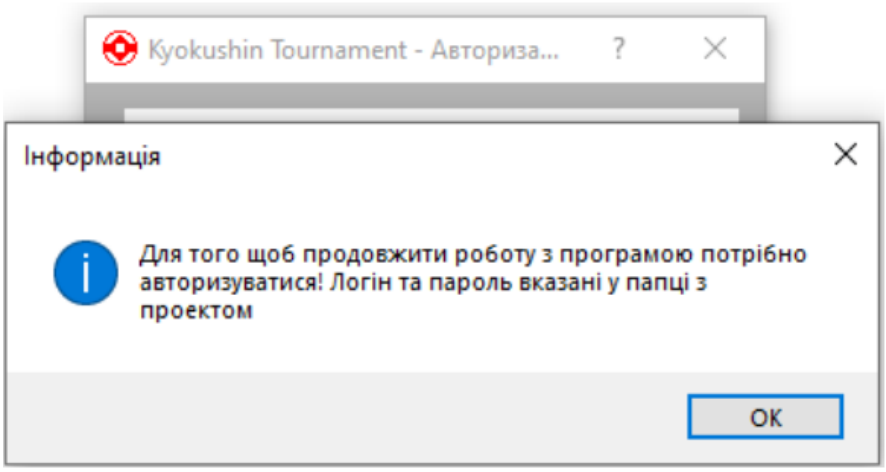

Рис. 2.42. Інструкція користувача по роботі у даному вікні

Для того, щоб продовжити роботу з додатком, потрібно ввести логін та пароль, які додаються до програми у текстовому файлі Password.txt (рис. 2.43).

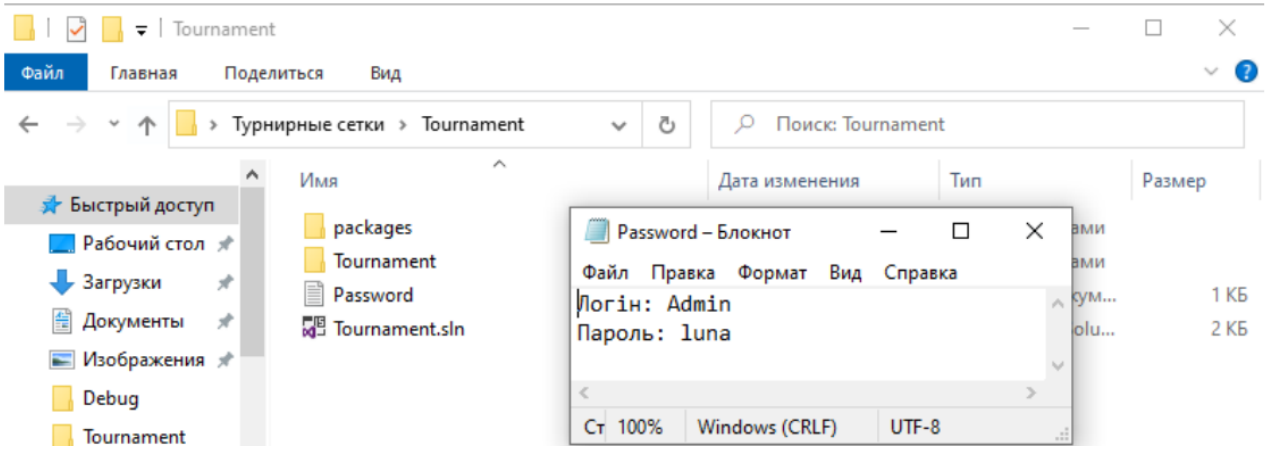

Рис. 2.43. Файл Password.txt з логіном та паролем для адміністратора

Після авторизації користувача, програма пропонує завантажити дані про всіх учасників у базу даних з веб-сайту, після чого з'явиться повідомлення про успішне завантаження (рис. 2.44) і дані відобразяться у таблиці.

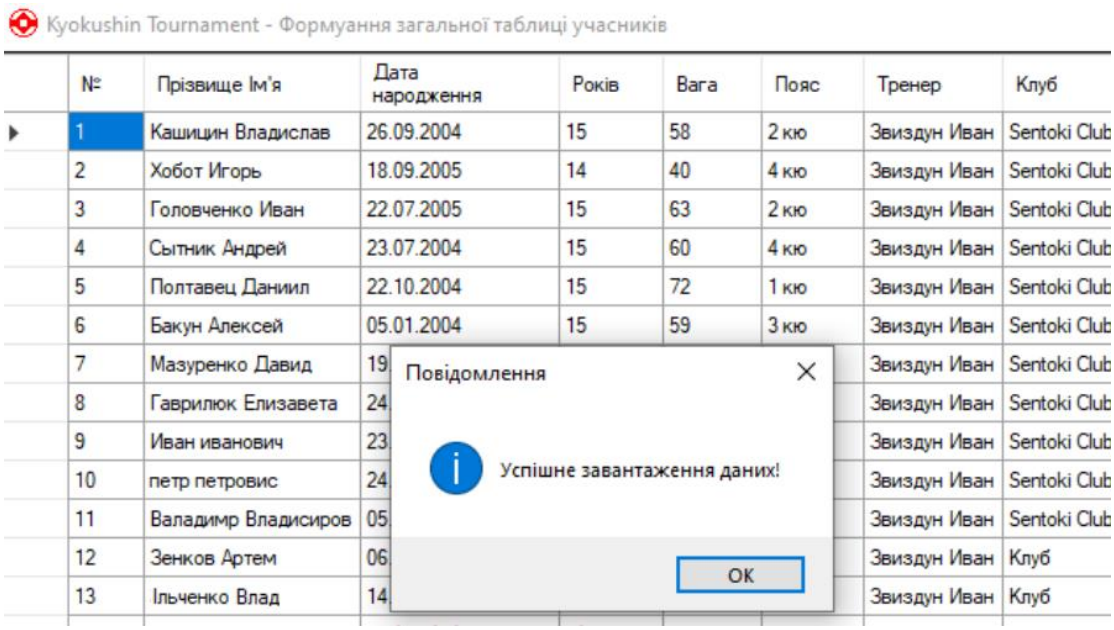

Рис. 2.44. Повідомлення про успішне завантаження даних про учасників змагань до таблиці

Далі потрібно натиснути на кнопку «Зберегти дані до бази даних». Коли дані запишуться в базу даних, також з’явиться повідомлення про успішний запис (рис. 2.45).

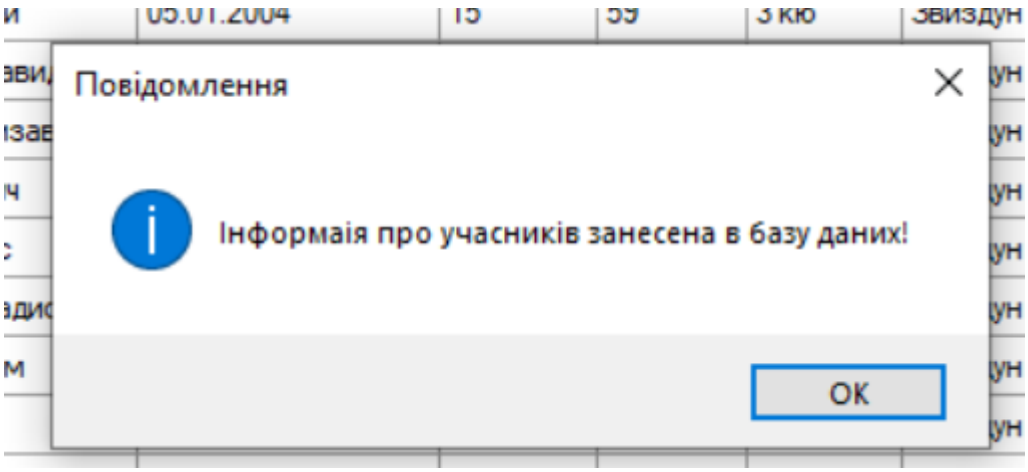

Рис. 2.45 Повідомлення про успішне завантаження даних

Після вище перерахованих дій користувачу необхідно натиснути на кнопку «Сформувати турнірні таблиці», з’явиться вікно для створення

турнірних таблиць та сіток (рис. 2.46). Таблиці сформуються в процесі відкриття вікна.

Рис. 2.46. Вікно «Формування турнірних таблиць та сіток»

Після перегляду правильності розміщення учасників по категоріям, користувач має можливість зберегти турнірні таблиці, натиснувши на кнопку меню «Зберегти таблиці у файл» (рис. 2.47), з’явиться повідомлення про успішне збереження даних турнірних сіток до файлу.

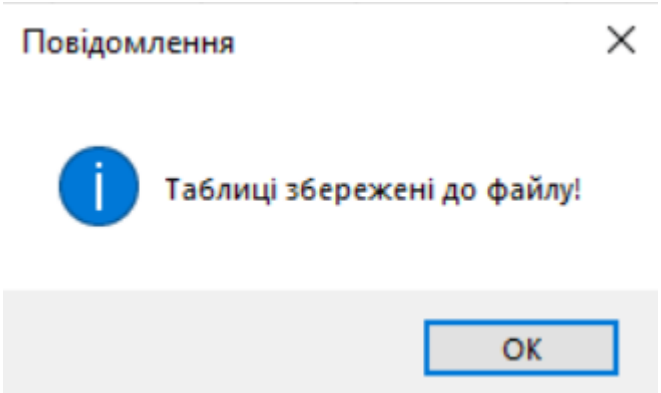

Рис. 2.47. Повідомлення про успішне збереження даних у файл

Дані збережуться у файл TournamentGrid з розширенням .xlsx, де кожний лист це окрема категорія (рис. 2.48).

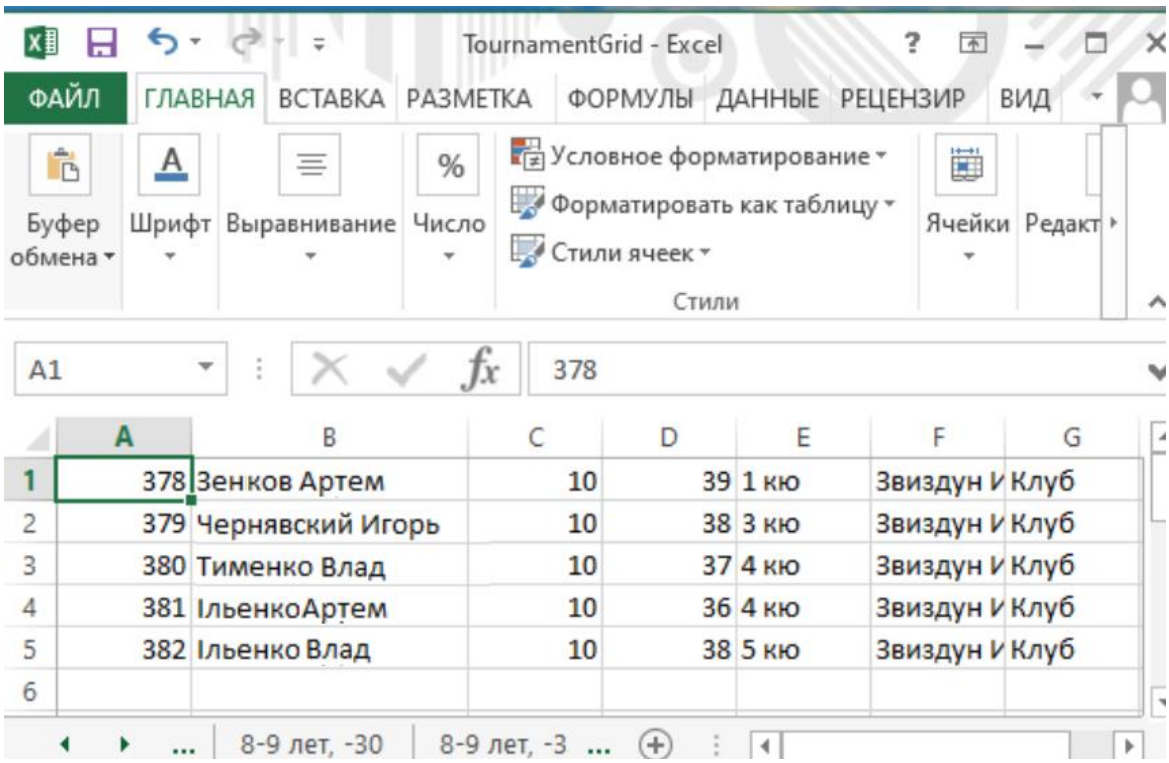

Рис. 2.48. Результат збереження турнірних таблиць до файлу

Для того, щоб сформувати графічне представлення турнірних сіток потрібно натиснути на кнопку меню «Створити графічне зображення сітки». Після успішного створення графічних зображень турнірних сіток з’явиться відповідне повідомлення (рис. 2.49).

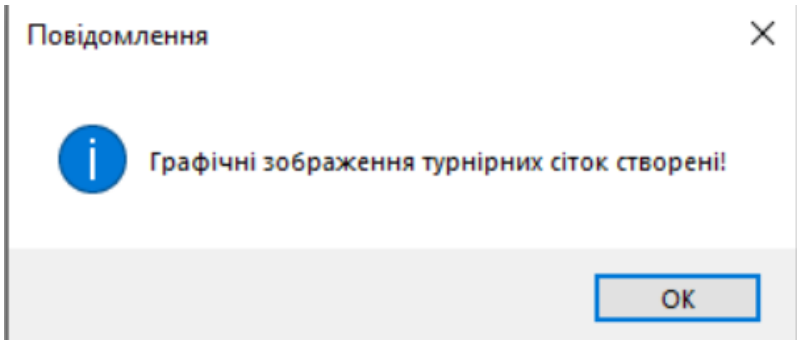

Рис. 2.49. Повідомлення про успішне створення графічних зображень турнірних сіток

Користувач має можливість переглянути турнірні сітки, для цього необхідно натиснути на кнопку «Переглянути турнірні сітки» і з’явиться вікно перегляду турнірних сіток (рис. 2.50).

Рис. 2.50. Вікно для перегляду створених турнірних сіток

У головному меню розташована кнопка «Друкувати», за необхідності адміністратор може на неї натиснути та вивести зображення турнірних сіток на друк (рис. 2.51).

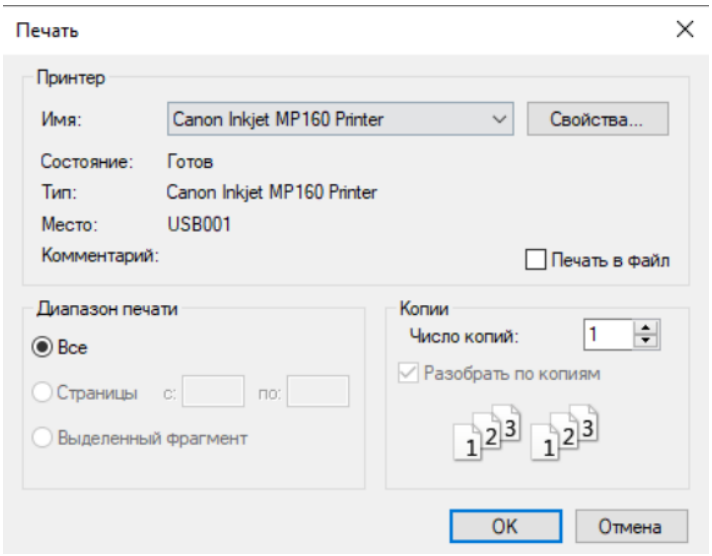

Рис. 2.51. Вікно для друку турнірних сіток.

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Вхідні дані:

- передбачуване число операторів – 1650;
- коефіцієнт складності програми – 1,5;
- коефіцієнт корекції програми в ході її розробки – 0,06;
- годинна заробітна плата програміста, грн / год – 55.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \quad (3.1)$$

де to - витрати праці на підготовку й опис поставленої задачі (50),

ти - витрати праці на дослідження алгоритму рішення задачі,

та - витрати праці на розробку блок-схеми алгоритму;

тп - витрати праці на програмування по готовій блок-схемі,

тотл - витрати праці на налагодження програми на ЕОМ,

тд - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \times C \times (1+ p) \text{ , людино-годин,} \quad (3.2)$$

де q – передбачуване число операторів,

C – коефіцієнт складності програми,

p – коефіцієнт кореляції програми в ході її розробки.

$$Q = 1650 \cdot 1{,}5 \cdot (1 + 0{,}06) = 2623.5 \text{ людино-годин.}$$

Витрати праці на вивчення опису задачі ti визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{QB}{(75...85)K}, \text{ людино-годин,} \qquad (3.3)$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі; B=1.2 ... 1.5,

k – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

Витрати праці на розробку алгоритму рішення задачі:

$$t_u = \frac{2623.5 \cdot 1{,}4}{78 \cdot 1{,}2} = 39.24, \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t\alpha = \frac{Q}{(20...25)K} \quad \text{людино-годин.,} \qquad (3.4)$$

$$t_{à} = \frac{2623.5}{21 \cdot 1{,}2} = 104.1 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)K} \quad \text{людино-годин,} \tag{3.5}$$

$$t_n = \frac{2623.5}{21 \cdot 1.2} = 104.1 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

$$t_{\text{отл}} = \frac{Q}{(4...5)K} \quad \text{людино-годин,} \tag{3.6}$$

$$t_{\text{îòë}} = \frac{2623.5}{4 \cdot 1.28} = 546.56 \text{ людино-годин.}$$

- за умови комплексного налагодження завдання:

$$t_{отл}^{к} = 1{,}2 \cdot t_{отл}; \tag{3.7}$$

$$t_{\text{îòë}} = 546.56 * 1{,}5 = 819{,}8$$

Витрати праці на підготовку документації:

$$t_{д} = t_{д\text{р}} + t_{до}, \text{ людино-годин,} \tag{3.8}$$

де tдр - трудомісткість підготовки матеріалів і рукопису.

$$t_{д\text{р}} = \frac{Q}{(15...20)K}, \text{ людино-годин.,} \tag{3.9}$$

$$t_{\text{äð}} = \frac{2623.5}{15 \cdot 1.2} = 145{,}75 \text{ людино-годин.}$$

tдо - трудомісткість редагування, печатки й оформлення документації:

$$t_{до} = 0{,}75 \cdot t_{д\text{р}}, \text{ людино-годин,} \tag{3.10}$$

$$tdo = 0{,}75 \cdot 145{,}75 = 109.31$$

$t_{äî} = 145.78 + 109.31 = 255.06$ людино-годин.

Отримуємо трудомісткість розробки програмного забезпечення:

$t = 50 + 39.24 + 104.10 + 104.10 + 546.56 + 255.06 = 1049.06$ люд-год.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ Кпо включають витрати на заробітну плату виконавця програми Зз/п і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$K_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн,} \qquad (3.11)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot C_{ПР}, \text{ грн,} \qquad (3.12)$$

де t - загальна трудомісткість, людино-годин,

Спр - середня годинна заробітна плата програміста, грн/година.

$Ç_{Çï} = 1049.06 \cdot 55 = 57698.3$ ãðí.

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \times C_{М}, \text{ грн,} \qquad (3.12)$$

де tотл - трудомісткість налагодження програми на ЕОМ, год.,

Смч - вартість машино-години ЕОМ, 3,5 грн/год.

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП.

$$Çîâ = 546.56 \times 3.5 = 1913 \text{ ãðí.}$$

$$Êïï = 57698.3 + 1913 = 59611.3 \text{ ãðí.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p} \text{ міс.} \qquad (3.13)$$

де Bk - число виконавців,

Fp - місячний фонд робочого часу (при 40 годинному робочому тижні Fp=176 годин).

$$T = \frac{1049}{1 \cdot 176} = 6 \text{ міс.}$$

Висновки: визначено трудомісткість розробленої інформаційної системи (1049 люд-год), проведений підрахунок вартості роботи по створенню програми (59611 грн.) та розраховано час на його створення (6 міс).

ВИСНОВКИ

В ході виконання кваліфікаційної роботи, було спроектовано та розроблено програмний продукт для автоматизації створення турнірних таблиць та сіток для змагань з бойових видів спорту.

Для виконання даної роботи були обрані наступні засоби розробки: середовище програмування Visual Studio 2017; мова програмування C#; конфігурація Windows Froms (для програмного додатку); СКБД SQLite (для програмного додатку); платформа ASP.NET MVC5 (для веб-сайту); СКБД MS SQL Server (для веб-сайту).

Створений програмний продукт має наступний функціонал:

– формує загальну таблицю учасників;
– створює турнірні таблиці по конкретній категорії учасників;
– створює турнірні сітки по конкретні категорії учасників;
– при необхідності виводить на друк турнірні сітки.

Для реалізації вище зазначених функцій було розділено програмний продукт на дві частини, а саме веб-сайт, де здійснюється внесення даних про учасників тренерами та програмний додаток для здійснення формування турнірних таблиць та сіток для спортивних змагань.

На веб-сайті можна переглянути дані про всіх зареєстрованих на змагання учасників і переглянути турнірні таблиці та сітки, якщо це гість (не авторизований користувач). Якщо це тренер, то перед ним відкривається додатково можливість додати дані про учасника на змагання та при необхідності редагувати їх або видаляти.

На веб-сайті існує аккаунт адміністратора, який в свою чергу має сторінку адміністратора. На цій сторінці він може змінити статус учасників для додавання їх в турнірну таблицю по конкретній категорії. Також у нього є можливість завантажити зображення турнірних сіток та змінити їх статус для відображення їх для будь-яких користувачів.

Програмний додаток розроблений же виключно для адміністратора, де він має змогу завантажити дані про всіх зареєстрованих учасників з веб-сайту та створити турнірні сітки, при необхідності роздрукувати їх.

Даний розроблений програмний продукт може бути застосований в організаціях, які організовують змагання з бойових видів спорту, наприклад: карате, тхейквондо, бокс і т.д.

В економічному розділі визначено трудомісткість розробленої інформаційної системи (1049 люд-год), проведений підрахунок вартості роботи по створенню програми (59611 грн.) та розраховано час на його створення (6 міс).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання : (ГОСТ 7.1-2003, IDT) : ДСТУ ГОСТ 7.1:2006. – Чинний з 2007–07–01. – К. : Держспоживстандарт України, 2007. – 47 с. – (Система стандартів з інформації, бібліотечної та видавничої справи) (Національний стандарт України).

2. Бусигін Б.С., Коротенко Г.М., Коротенко Л.М. Прикладна інформатика. Підручник для студентів комп'ютерних спеціальностей. – Дніпропетровськ: Видавництво НГУ, 2004. – 559 с. URL: http://www.programmer.dp.ua/book-ua-k01.php. дата звернення: 15.03.2019.

3. Бусыгин Б.С., Дивизинюк М.М., Коротенко Г.М., Коротенко Л.М. Введение в современную информатику. Учебник. – Севастополь: Издательство СНУЯЭиП, 2005. – 644 с. / URL: http://www.programmer.dp.ua/book-ru-k02.php. дата звернення: 15.01.2021.

4. Відомості про браузер, URL: https://uk.wikipedia.org /Браузер. дата звернення: 17.02.2021.

5. Відомості про діаграму послідовностей (Sequance diagram) URL: https://ru.wikipedia.org/wiki/Диаграмма_послеователь ностей. дата звернення: 18.02.2021.

6. Відомості про діаграму послідовностей (Use-case diagram) URL: .https://ru.wikipedia.org/wiki/Диаграмма_прецедентов/ дата звернення: 15.01.2021

7. Відомості про файли платформи розробки веб сайтів ASP.NET MVC5/ URL: https://professorweb.ru/my/ASP_NET/mvc/level1/1_3.php дата звернення: 17.02.2021.

8. Відомості про СКБД SQLite URL: http://unetway.com/tutorial/sqlite/. дата звернення: 15.01.2021.

9. Відомості про ADO.NET, URL: https://metanit.com /sharp/ adonet/1.1.php. дата звернення: 15.01.2021.

10. Герберт Шилдт. «C#: базовий курс. 4-е видання», - Київ: Видавничий дім «Вільямс», 2006. – 347с.

11. Джеймс Р. Грофф, Пол Н. Вайнберг, Эндрю Дж. Оппель «SQL. Полное руководство третье издание» Видавництво Віл’ямс 2015 -960 стр.

12. Джо Майо «Microsoft Visual Studio 2010» 2011. – 267с.

13. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення : ДСТУ 3008-95. – Чинний від 1996–01–01. – К. : Держстандарт України, 1996. – 39 с.

14. ДСТУ 2394-94 Інформація та документація. Комплектування фонду, бібліографічний опис, аналіз документів. Терміни та визначення. – Чинний від 01.01.1995. - Київ : Держстандарт України, 1994. – 88 с.

15. Загальні відомості про MS Excel, URL: : http://www. neumeka.ru/excel.html. дата звернення: 18.02.2021.

16. Інформація про комп’ютерні системи проведення змагань/ URL: https://ru.wikipedia.org/wiki/Компьютерная_система_проведения_соревнований. дата звернення: 15.01.2021. Інформація про використані бібліотеки/ URL: https://docs.microsoft.com/en-us/dotnet/api/system.data?view =netframework-4.8 дата звернення: 15.02.2021.

17. Інформація про ASP.NET MVC / URL: https://metanit.com/sharp/mvc5 /1.1.php. дата звернення: 15.01.2021.

18. Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності “Комп’ютерні системи ” / Укладачі О.Г. Вагонова, Нікітіна О.Б. Н.Н. Романюк – Дніпропетровськ: Національний гірничий університет. – 2013. – 23с.

19. Методичні рекомендації до виконання кваліфікаційних робіт бакалаврів напряму підготовки 122 «Комп’ютерні науки» галузі знань 12 Інформаційні технології/, Л.М. Коротенко , О.С. Шевцова; Нац. гірн. ун-т. – Д : ДВНЗ НГУ, 2019. – 65 с.

20. Методичні рекомендації щодо написання, оформлення та представлення учнівських науково-дослідницьких робіт учнів – членів Малої

академії наук України / Г.Г. Півняк, Л.М. Коротенко, І.М. Удовик, Є.М. Головня – Д.: ДВНЗ «Національний гірничий університет», 2017. – 24 с.

21. Никита Культин «Microsoft Visual C# в задачах и примерах (2-еиздание)» 2014. – 354с.

22. Переваги та недоліки Sql Server URL: : http://programmersworld.xyz/article/18/62. дата звернення: 15.01.2021.

23. Порвівняння операційних систем, URL: https://ichip.ru/ sovety/chto-luchshe-windows-7-ili-windows-10-sravnitelnaya-tablica-197442. дата звернення: 15.02.2021.

24. Правила проведения турниров по швейцарской системе ФИДЕ// URL: https://www.fide.com/fide/handbook.html?id=18&view=category дата звернення: 17.01.2021.

25. Прийоми програмування з фалами, URL: https://sites .google.com/site/c4plus/najprostisa-programa-na-c/opis-roboti-z-fajlami. дата звернення: 15.01.2021.

26. Скорочення слів в українській мові у бібліографічному описі. Загальні правила та вимоги : ДСТУ 3582-97. – Чинний від 1998–07–01. – К. : Держстандарт України, 1998. – 24 с. – (Державний стандарт України).

27. Эндрю Троелсен, Филипп Джепикс «Язык программирования C# 7 и платформы .NET и .NET Core, 8-е издание, том 1» 2019. – 607с.

ДОДАТОК А

КОД ПРОГРАМИ

FormAvtorization

```
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;
using System.Data.SQLite;
namespace Tournament
{
public partial class FormAutorization: Form
{
private static SQLiteConnection sqlCon;
public FormAutorization()
{
InitializeComponent();
try
{
sqlCon = new SQLiteConnection("Data Source = tournament.db"); sqlCon.Open();
SQLiteCommand cmd = new SQLiteCommand("SELECT Name FROM Instructors", sqlCon); using
(SQLiteDataReader reader = cmd.ExecuteReader()) {
while (reader.Read())
{
comboBoxInstructors.Items.Add(reader["Name"].ToString());
}
}
}
catch (SQLiteException ex)
{
MessageBox.Show($"Соединение с базой не установлено!\n {ex.Message}");
}
}
private void FormAutorization_FormClosed(object sender, FormClosedEventArgs e) {
Application.Exit();
}
private void ButtonEnter_Click(object sender, EventArgs e)
{
SQLiteCommand cmd = new SQLiteCommand("SELECT Login, Password FROM Admins",
sqlCon);
using (SQLiteDataReader reader = cmd.ExecuteReader())
{
while (reader.Read())
{
if (textBoxAdminLogin.Text.Trim() == reader["Login"].ToString() &&
textBoxAdminPassword.Text.Trim() == reader["Password"].ToString())
{
new FormMain(ref sqlCon, textBoxAdminLogin.Text.Trim(), "ad-
min").Show();
return;
}
}
}
MessageBox.Show("Неверные логин и пароль!");
}
private void FormAutorization_Load(object sender, EventArgs e)
```

```
{
}
}
}
```

FormMain

```
namespace Tournament
{
public partial class FormMain: Form
{
private static SQLiteConnection sqlCon;
private static string Account { get; set; }
private static string Status { get; set; }
public FormMain(ref SQLiteConnection sql, string login, string status)
{
InitializeComponent();
sqlCon = sql;
Account = login;
Status = status;
}
private async void LoadExcelFile(object sender, EventArgs e)
{
string excelFile = "";
using (OpenFileDialog op = new OpenFileDialog())
{
op.Filter = "Excel (.xlsx, .xls)|*.xlsx;*.xls;| Все файлы (*.*)|*.*"; op.Title = "Выберите файл с участниками";
if (op.ShowDialog() != DialogResult.Cancel)
excelFile = op.FileName;
}
if (excelFile == "")
{
MessageBox.Show("Файл с данными не выбран");
return;
}
Excel.Application exApp = new Excel.Application(); Excel.Workbook wBook = exApp.Workbooks.Open(excelFile);
Excel.Worksheet wSheet = wBook.Sheets[1];
dataGridViewParticipant.Columns.Add("Number", "№"); dataGridViewParticipant.Columns.Add("Name", "ФИО");
dataGridViewParticipant.Columns.Add("Birthday", "Дата рождения"); dataGridViewParticipant.Columns.Add("Age",
"Лет"); dataGridViewParticipant.Columns.Add("Weight", "Вес"); dataGridViewParticipant.Columns.Add("Ku",
"Степень"); dataGridViewParticipant.Columns.Add("Instructor", "Тренер");
dataGridViewParticipant.Columns.Add("Club", "Клуб");
for (int i = 0; i < dataGridViewParticipant.ColumnCount; i++)
dataGridViewParticipant.Columns[i].AutoSizeMode = DataGridViewAu-toSizeColumnMode.DisplayedCells;
try
{
await Task.Run(() =>
{
for (int i = 2; i <= wSheet.UsedRange.Rows.Count; i++)
{
List<object> data = new List<object>();
for (int j = 1; j <= 8; j++)
{
if (j == 3)
{
DateTime dt =
DateTime.FromOADate((double)wSheet.UsedRange.Cells[i, j].Value2); data.Add(dt.ToString("dd.MM.yyyy"));
}
else if (j == 5)
{
data.Add(Convert.ToInt16(Math.Round((double)wSheet.UsedRange.Cells[i, j].Value2)));
}
else data.Add(wSheet.UsedRange.Cells[i, j].Value2);
```

```
}
dataGridViewParticipant.Invoke(new Action(() => dataGridViewPartici-pant.Rows.Add(data.ToArray())));
}
});
if (Status == "instructor")
{
SQLiteCommand cmd = new SQLiteCommand($"SELECT Club FROM Instructors WHERE Name = '{Account}'",
sqlCon);
string club = "";
using (SQLiteDataReader reader = cmd.ExecuteReader())
{
while (reader.Read())
{
club = reader["Club"].ToString();
break;
}
}
foreach (DataGridViewRow r in dataGridViewParticipant.Rows)
if (r.Cells[7].Value.ToString() != club)
r.ReadOnly = true;
}
}
catch (Exception ex)
{
MessageBox.Show(ex.Message);
}
finally
{
exApp.Quit();
exApp = null;
wSheet = null;
wBook = null;
MessageBox.Show("Загрузка данных из файла завершена!");
}
}
private async void UploadParticipantsInDatabase(object sender, EventArgs e)
{
if (dataGridViewParticipant.RowCount == 0)
{
MessageBox.Show("Сначала необходимо загрузить файл с данными!", "Увага", Mes-sageBoxButtons.OK,
MessageBoxIcon.Error);
return;
}
try
{
SQLiteCommand cmdClear = new SQLiteCommand("DELETE FROM Participants", sqlCon);
cmdClear.ExecuteNonQuery();
await Task.Run(() =>
{
for (int i = 0; i < dataGridViewParticipant.Rows.Count; i++)
{
List<object> data = new List<object>();
for (int j = 0; j < 8; j++)
{
data.Add(dataGridViewParticipant[j, i].Value.ToString());
}
string cmdText = $@"INSERT INTO Participants (Name, Birthday, Age, Weight, Ku, Instructor, Club)
VALUES ('{data[1].ToString()}','{data[2].ToString()}','{data[3].ToString()}',
'{data[4].ToString()}','{data[5].ToString()}','{data[6].ToString()}','{data[7].ToString()}')"; SQLiteCommand cmd =
new SQLiteCommand(cmdText, sqlCon); cmd.ExecuteNonQuery();
}
});
```

```
}
catch (Exception ex)
{
MessageBox.Show(ex.Message);
}
finally
{
MessageBox.Show("Информация о участниках занесена в базу данных!");
}
}
private void LoadFormGrid(object sender, EventArgs e)
{
new FormGrid(ref sqlCon).Show();
}
}
}
```

FormGrid

```
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;
using System.Data.SQLite;
using System.Runtime.InteropServices;
using Excel = Microsoft.Office.Interop.Excel;
using Word = Microsoft.Office.Interop.Word;
using System.IO;
namespace Tournament
{
public partial class FormGrid: Form
{
private static SQLiteConnection sqlCon;
private static List<DataGridView> grids;
public FormGrid(ref SQLiteConnection sql)
{
InitializeComponent();
sqlCon = sql;
grids = new List<DataGridView>() { dataGridView67bef25, dataGridView67bef30, data-GridView67af30,
dataGridView89bef30, dataGridView89bef35, dataGridView89bef40, data-
GridView89af40,
dataGridView1213bef40, dataGridView1213bef45, dataGridView1213bef50, data-GridView1213bef55,
dataGridView1213af55,
dataGridView1415bef50, dataGridView1415bef55, dataGridView1415bef60, data-GridView1415bef65,
dataGridView1415af65,
dataGridView1617bef60, dataGridView1617bef65, dataGridView1617bef70, data-GridView1617bef75,
dataGridView1617af75,
dataGridView18bef70, dataGridView18bef75, dataGridView18bef80, data-GridView18bef85, dataGridView18af85};
foreach (DataGridView g in grids)
{
g.Columns.Add("Number", "№");
g.Columns.Add("Name", "ФИО");
g.Columns.Add("Age", "Лет");
g.Columns.Add("Weight", "Вес");
g.Columns.Add("Ku", "Степень");
g.Columns.Add("Instructor", "Тренер");
g.Columns.Add("Club", "Клуб");
for (int i = 0; i < g.ColumnCount; i++)
```

```
g.Columns[i].AutoSizeMode = DataGridViewAutoSizeColumnMode.DisplayedCells;
}
try
{
using (SQLiteDataReader r = new SQLiteCommand("SELECT * FROM Participants WHERE Age >= 6 AND Age <=
7 AND Weight <= 25", sqlCon).ExecuteReader())
{
while (r.Read())
{
dataGridView67bef25.Rows.Add(r["Id"], r["Name"], r["Age"], r["Weight"], r["Ku"], r["Instructor"], r["Club"]);
}
}
using (SQLiteDataReader r = new SQLiteCommand("SELECT * FROM Participants WHERE Age >= 6 AND Age <=
7 AND Weight > 25 AND Weight <= 30", sqlCon).ExecuteReader())
{
while (r.Read())
{
dataGridView67bef30.Rows.Add(r["Id"], r["Name"], r["Age"], r["Weight"], r["Ku"], r["Instructor"], r["Club"]);
}
}
using (SQLiteDataReader r = new SQLiteCommand("SELECT * FROM Participants WHERE Age >= 6 AND Age <=
7 AND Weight > 30", sqlCon).ExecuteReader())
{
while (r.Read())
{
dataGridView67af30.Rows.Add(r["Id"], r["Name"], r["Age"], r["Weight"], r["Ku"], r["Instructor"], r["Club"]);
}
}
using (SQLiteDataReader r = new SQLiteCommand("SELECT * FROM Participants WHERE Age >= 8 AND Age <=
9 AND Weight <= 30", sqlCon).ExecuteReader())
{
while (r.Read())
{
dataGridView89bef30.Rows.Add(r["Id"], r["Name"], r["Age"], r["Weight"], r["Ku"], r["Instructor"], r["Club"]);
}
}
using (SQLiteDataReader r = new SQLiteCommand("SELECT * FROM Participants WHERE Age >= 8 AND Age <=
9 AND Weight > 30 AND Weight <= 35", sqlCon).ExecuteReader())
{
while (r.Read())
{
dataGridView89bef35.Rows.Add(r["Id"], r["Name"], r["Age"], r["Weight"], r["Ku"], r["Instructor"], r["Club"]);
}
}
using (SQLiteDataReader r = new SQLiteCommand("SELECT * FROM Participants WHERE Age >= 8 AND Age <=
9 AND Weight > 35 AND Weight <= 40", sqlCon).ExecuteReader())
{
while (r.Read())
{
dataGridView89bef40.Rows.Add(r["Id"], r["Name"], r["Age"], r["Weight"], r["Ku"], r["Instructor"], r["Club"]);
}
}
using (SQLiteDataReader r = new SQLiteCommand("SELECT * FROM Participants WHERE Age >= 8 AND Age <=
9 AND Weight > 40", sqlCon).ExecuteReader())
{
while (r.Read())
{
dataGridView89af40.Rows.Add(r["Id"], r["Name"], r["Age"], r["Weight"], r["Ku"], r["Instructor"], r["Club"]);
}
}
using (SQLiteDataReader r = new SQLiteCommand("SELECT * FROM Participants WHERE Age >= 10 AND Age
<= 11 AND Weight <= 35", sqlCon).ExecuteReader())
```

```
{
while (r.Read())
{
dataGridView1011bef35.Rows.Add(r["Id"], r["Name"], r["Age"], r["Weight"], r["Ku"], r["Instructor"], r["Club"]);
}
}
using (SQLiteDataReader r = new SQLiteCommand("SELECT * FROM Participants WHERE Age >= 10 AND Age
<= 11 AND Weight > 35 AND Weight <= 40", sqlCon).ExecuteReader())
{
while (r.Read())
{
dataGridView1011bef40.Rows.Add(r["Id"], r["Name"], r["Age"], r["Weight"], r["Ku"], r["Instructor"], r["Club"]);
}
}
using (SQLiteDataReader r = new SQLiteCommand("SELECT * FROM Participants WHERE Age >= 10 AND Age
<= 11 AND Weight > 40 AND Weight <= 45", sqlCon).ExecuteReader())
{
while (r.Read())
{
dataGridView1011bef45.Rows.Add(r["Id"], r["Name"], r["Age"], r["Weight"], r["Ku"], r["Instructor"], r["Club"]);
}
}
using (SQLiteDataReader r = new SQLiteCommand("SELECT * FROM Participants WHERE Age >= 10 AND Age
<= 11 AND Weight > 45 AND Weight <= 50", sqlCon).ExecuteReader())
{
while (r.Read())
{
dataGridView1011bef50.Rows.Add(r["Id"], r["Name"], r["Age"], r["Weight"], r["Ku"], r["Instructor"], r["Club"]);
}
}
using (SQLiteDataReader r = new SQLiteCommand("SELECT * FROM Participants WHERE Age >= 10 AND Age
<= 11 AND Weight > 50", sqlCon).ExecuteReader())
{
while (r.Read())
{
dataGridView1011af50.Rows.Add(r["Id"], r["Name"], r["Age"], r["Weight"], r["Ku"], r["Instructor"], r["Club"]);
}
}
using (SQLiteDataReader r = new SQLiteCommand("SELECT * FROM Participants WHERE Age >= 12 AND Age
<= 13 AND Weight <= 40", sqlCon).ExecuteReader())
{
while (r.Read())
{
dataGridView1213bef40.Rows.Add(r["Id"], r["Name"], r["Age"], r["Weight"], r["Ku"], r["Instructor"], r["Club"]);
}
}
using (SQLiteDataReader r = new SQLiteCommand("SELECT * FROM Participants WHERE Age >= 12 AND Age
<= 13 AND Weight > 40 AND Weight <= 45", sqlCon).ExecuteReader())
{
while (r.Read())
{
dataGridView1213bef45.Rows.Add(r["Id"], r["Name"], r["Age"], r["Weight"], r["Ku"], r["Instructor"], r["Club"]);
}
}
using (SQLiteDataReader r = new SQLiteCommand("SELECT * FROM Participants WHERE Age >= 12 AND Age
<= 13 AND Weight > 45 AND Weight <= 50", sqlCon).ExecuteReader())
{
while (r.Read())
{
dataGridView1213bef50.Rows.Add(r["Id"], r["Name"], r["Age"], r["Weight"], r["Ku"], r["Instructor"], r["Club"]);
}
}
```

```
using (SQLiteDataReader r = new SQLiteCommand("SELECT * FROM Participants WHERE Age >= 12 AND Age <= 13 AND Weight > 50 AND Weight <= 55", sqlCon).ExecuteReader())
{
while (r.Read())
{
dataGridView1213bef55.Rows.Add(r["Id"], r["Name"], r["Age"], r["Weight"], r["Ku"], r["Instructor"], r["Club"]);
}
}
using (SQLiteDataReader r = new SQLiteCommand("SELECT * FROM Participants WHERE Age >= 12 AND Age <= 13 AND Weight > 55", sqlCon).ExecuteReader())
{
while (r.Read())
{
dataGridView1213af55.Rows.Add(r["Id"], r["Name"], r["Age"], r["Weight"], r["Ku"], r["Instructor"], r["Club"]);
}
}
using (SQLiteDataReader r = new SQLiteCommand("SELECT * FROM Participants WHERE Age >= 14 AND Age <= 15 AND Weight <= 50", sqlCon).ExecuteReader())
{
while (r.Read())
{
dataGridView1415bef50.Rows.Add(r["Id"], r["Name"], r["Age"], r["Weight"], r["Ku"], r["Instructor"], r["Club"]);
}
}
using (SQLiteDataReader r = new SQLiteCommand("SELECT * FROM Participants WHERE Age >= 14 AND Age <= 15 AND Weight > 50 AND Weight <= 55", sqlCon).ExecuteReader())
{
while (r.Read())
{
dataGridView1415bef55.Rows.Add(r["Id"], r["Name"], r["Age"], r["Weight"], r["Ku"], r["Instructor"], r["Club"]);
}
}
using (SQLiteDataReader r = new SQLiteCommand("SELECT * FROM Participants WHERE Age >= 14 AND Age <= 15 AND Weight > 55 AND Weight <= 60", sqlCon).ExecuteReader())
{
while (r.Read())
{
dataGridView1415bef60.Rows.Add(r["Id"], r["Name"], r["Age"], r["Weight"], r["Ku"], r["Instructor"], r["Club"]);
}
}
using (SQLiteDataReader r = new SQLiteCommand("SELECT * FROM Participants WHERE Age >= 14 AND Age <= 15 AND Weight > 60 AND Weight <= 65", sqlCon).ExecuteReader())
{
while (r.Read())
{
dataGridView1415bef65.Rows.Add(r["Id"], r["Name"], r["Age"], r["Weight"], r["Ku"], r["Instructor"], r["Club"]);
}
}
using (SQLiteDataReader r = new SQLiteCommand("SELECT * FROM Participants WHERE Age >= 14 AND Age <= 15 AND Weight > 65", sqlCon).ExecuteReader())
{
while (r.Read())
{
dataGridView1415af65.Rows.Add(r["Id"], r["Name"], r["Age"], r["Weight"], r["Ku"], r["Instructor"], r["Club"]);
}
}
using (SQLiteDataReader r = new SQLiteCommand("SELECT * FROM Participants WHERE Age >= 16 AND Age <= 17 AND Weight <= 60", sqlCon).ExecuteReader())
{
while (r.Read())
{
dataGridView1617bef60.Rows.Add(r["Id"], r["Name"], r["Age"], r["Weight"], r["Ku"], r["Instructor"], r["Club"]);
```

```
}
}
using (SQLiteDataReader r = new SQLiteCommand("SELECT * FROM Participants WHERE Age >= 16 AND Age
<= 17 AND Weight > 60 AND Weight <= 65", sqlCon).ExecuteReader())
{
while (r.Read())
{
dataGridView1617bef65.Rows.Add(r["Id"], r["Name"], r["Age"], r["Weight"], r["Ku"], r["Instructor"], r["Club"]);
}
}
using (SQLiteDataReader r = new SQLiteCommand("SELECT * FROM Participants WHERE Age >= 16 AND Age
<= 17 AND Weight > 65 AND Weight <= 70", sqlCon).ExecuteReader())
{
while (r.Read())
{
dataGridView1617bef70.Rows.Add(r["Id"], r["Name"], r["Age"], r["Weight"], r["Ku"], r["Instructor"], r["Club"]);
}
}
using (SQLiteDataReader r = new SQLiteCommand("SELECT * FROM Participants WHERE Age >= 16 AND Age
<= 17 AND Weight > 70 AND Weight <= 75", sqlCon).ExecuteReader())
{
while (r.Read())
{
dataGridView1617bef75.Rows.Add(r["Id"], r["Name"], r["Age"], r["Weight"], r["Ku"], r["Instructor"], r["Club"]);
}
}
using (SQLiteDataReader r = new SQLiteCommand("SELECT * FROM Participants WHERE Age >= 16 AND Age
<= 17 AND Weight > 75", sqlCon).ExecuteReader())
{
while (r.Read())
{
dataGridView1617af75.Rows.Add(r["Id"], r["Name"], r["Age"], r["Weight"], r["Ku"], r["Instructor"], r["Club"]);
}
}
using (SQLiteDataReader r = new SQLiteCommand("SELECT * FROM Participants WHERE Age > 18 AND Weight
<= 70", sqlCon).ExecuteReader())
{
while (r.Read())
{
dataGridView18bef70.Rows.Add(r["Id"], r["Name"], r["Age"], r["Weight"], r["Ku"], r["Instructor"], r["Club"]);
}
}
using (SQLiteDataReader r = new SQLiteCommand("SELECT * FROM Participants WHERE Age > 18 AND Weight
> 70 AND Weight <= 75", sqlCon).ExecuteReader())
{
while (r.Read())
{
dataGridView18bef75.Rows.Add(r["Id"], r["Name"], r["Age"], r["Weight"], r["Ku"], r["Instructor"], r["Club"]);
}
}
using (SQLiteDataReader r = new SQLiteCommand("SELECT * FROM Participants WHERE Age > 18 AND Weight
> 75 AND Weight <= 80", sqlCon).ExecuteReader())
{
while (r.Read())
{
dataGridView18bef80.Rows.Add(r["Id"], r["Name"], r["Age"], r["Weight"], r["Ku"], r["Instructor"], r["Club"]);
}
}
using (SQLiteDataReader r = new SQLiteCommand("SELECT * FROM Participants WHERE Age > 18 AND Weight
> 80 AND Weight <= 85", sqlCon).ExecuteReader())
{
while (r.Read())
```

```
{
dataGridView18bef85.Rows.Add(r["Id"], r["Name"], r["Age"], r["Weight"], r["Ku"], r["Instructor"], r["Club"]);
}
}
using (SQLiteDataReader r = new SQLiteCommand("SELECT * FROM Participants WHERE Age > 18 AND Weight
> 85", sqlCon).ExecuteReader())
{
while (r.Read())
{
dataGridView18af85.Rows.Add(r["Id"], r["Name"], r["Age"], r["Weight"], r["Ku"], r["Instructor"], r["Club"]);
}
}
}
catch (SQLiteException ex)
{
MessageBox.Show(ex.Message);
}
finally
{
}
}
private void CreateGridToolStripMenuItem_Click(object sender, EventArgs e)
{
RichTextBox rtbHeader = new RichTextBox
{
Font = new Font("Arial", 14, FontStyle.Bold)
};
RichTextBox rtb = new RichTextBox
{
Font = new Font("Arial", 12, FontStyle.Regular)
};
string tempRtfFile =
Path.Combine(Path.GetDirectoryName(Application.ExecutablePath), "temp.rtf");
string path = Path.Combine(Path.GetDirectoryName(Application.ExecutablePath), "TournamentGrid.docx");
Word.Application word = new Word.Application();
Word.Document worddoc = word.Documents.Add();
object missing = Type.Missing;
object objUnit = Word.WdUnits.wdStory;
rtbHeader.Text = "";
rtbHeader.SaveFile(tempRtfFile, RichTextBoxStreamType.RichText); word.Selection.Range.InsertFile(tempRtfFile);
word.Selection.Range.Paragraphs.Alignment =
Word.WdParagraphAlignment.wdAlignParagraphLeft; word.Selection.EndKey(ref objUnit, ref missing);
File.Delete(tempRtfFile);
rtb.Text = "";
rtb.SaveFile(tempRtfFile, RichTextBoxStreamType.RichText); word.Selection.Range.InsertFile(tempRtfFile);
word.Selection.Range.Paragraphs.Alignment =
Word.WdParagraphAlignment.wdAlignParagraphLeft; word.Selection.EndKey(ref objUnit, ref missing);
File.Delete(tempRtfFile);
foreach (DataGridView g in grids)
{
if (g.RowCount == 0)
continue;
string grid = g.Name.Replace("dataGridView", "").Replace("bef","-").Replace("af","+");
if (grid.Contains("+"))
grid = grid.Replace("+", "-").Insert(grid.Length, "+"); string[] param = grid.Split('-'); if (param[0] != "18")
param[0] = param[0].Insert(param[0].Length == 2 ? 1 : 2,"-"); if (!param[1].Contains("+"))
param[1] = param[1].Insert(0, "-");
rtbHeader.Text = $"{param[0]} лет, весовая категория {param[1]} кг:\n"; rtbHeader.SaveFile(tempRtfFile,
RichTextBoxStreamType.RichText); word.Selection.Range.InsertFile(tempRtfFile);
word.Selection.Range.Paragraphs.Alignment =
Word.WdParagraphAlignment.wdAlignParagraphLeft; word.Selection.EndKey(ref objUnit, ref missing);
File.Delete(tempRtfFile);
```

```
bool even = g.RowCount % 2 == 0;
rtb.Text = "";
if (g.RowCount > 1)
{
int n = 1;
for (int i = 0; i < (even ? g.RowCount - 1 : g.RowCount - 2); i += 2)
{
rtb.Text += $"{n++}. {g[1, i].Value.ToString()} - {g[1, i + 1].Value.ToString().Trim()};\n";
}
if (!even)
{
rtb.Text += $"{n}. {g[1, g.RowCount - 1].Value.ToString()} - победитель
пары {n - 1};\n";
}
}
else if (g.RowCount == 1)
{
rtb.Text += $"1. {g[1, 0].Value.ToString()};\n";
}
rtb.Text += "\n\n";
rtb.SaveFile(tempRtfFile, RichTextBoxStreamType.RichText); word.Selection.Range.InsertFile(tempRtfFile);
word.Selection.Range.Paragraphs.Alignment =
Word.WdParagraphAlignment.wdAlignParagraphLeft; word.Selection.EndKey(ref objUnit, ref missing);
File.Delete(tempRtfFile);
}
worddoc.SaveAs(path);
if (worddoc != null)
{
Marshal.FinalReleaseComObject(worddoc);
worddoc = null;
}
if (word != null)
{
word.Quit(SaveChanges: false);
Marshal.FinalReleaseComObject(word);
word = null;
}
MessageBox.Show("Турнирная сетка сохранена в файл!");
}
private void SaveGridsInExcelToolStripMenuItem_Click(object sender, EventArgs e) {
string path = AppDomain.CurrentDomain.BaseDirectory + "TournamentGrid.xlsx"; Excel.Application exApp = new
Excel.Application(); Excel.Workbook wBook = exApp.Workbooks.Add();
for (int i = 0; i < grids.Count; i++)
wBook.Sheets.Add();
int p = 1;
foreach (DataGridView g in grids)
{
string grid = g.Name.Replace("dataGridView", "").Replace("bef", "-").Replace("af", "+");
if (grid.Contains("+"))
grid = grid.Replace("+", "-").Insert(grid.Length, "+"); string[] param = grid.Split('-'); if (param[0] != "18")
param[0] = param[0].Insert(param[0].Length == 2 ? 1 : 2, "-"); if (!param[1].Contains("+"))
param[1] = param[1].Insert(0, "-"); wBook.Sheets[p].Name = $"{param[0]} лет, {param[1]}";
for(int i = 0; i < g.RowCount; i++)
{
wBook.Sheets[p].Rows[i + 1].Columns[1] = g[0, i].Value.ToString(); wBook.Sheets[p].Rows[i + 1].Columns[2] = g[1,
i].Value.ToString(); wBook.Sheets[p].Rows[i + 1].Columns[3] = g[2, i].Value.ToString(); wBook.Sheets[p].Rows[i +
1].Columns[4] = g[3, i].Value.ToString(); wBook.Sheets[p].Rows[i + 1].Columns[5] = g[4, i].Value.ToString();
wBook.Sheets[p].Rows[i + 1].Columns[6] = g[5, i].Value.ToString(); wBook.Sheets[p].Rows[i + 1].Columns[7] = g[6,
i].Value.ToString();
}
p++;
}
```

```
wBook.SaveAs(path);
exApp.Quit();
exApp = null;
wBook = null;
MessageBox.Show("Таблицы сохранены в файл!");
}
private void CreateGraphicGridToolStripMenuItem_Click(object sender, EventArgs e) {
Directory.CreateDirectory(AppDomain.CurrentDomain.BaseDirectory + $"\\Турнирныесетки\\");
foreach (DataGridView g in grids)
{
if (g.RowCount == 0)
continue;
string grid = g.Name.Replace("dataGridView", "").Replace("bef", "-").Replace("af", "+");
if (grid.Contains("+"))
grid = grid.Replace("+", "-").Insert(grid.Length, "+"); string[] param = grid.Split('-'); if (param[0] != "18")
param[0] = param[0].Insert(param[0].Length == 2 ? 1 : 2, "-"); if (!param[1].Contains("+"))
param[1] = param[1].Insert(0, "-");
Bitmap bmp = new Bitmap(877, 620);
List<Point> connects = new List<Point>();
int maxL = g[1, 0].Value.ToString().Length + g[6, 0].Value.ToString().Length +
1;
string maxLn = g[1, 0].Value.ToString() + " " + g[6, 0].Value.ToString(); for (int i = 1; i < g.RowCount; i++)
if (g[1, i].Value.ToString().Length + g[6, i].Value.ToString().Length + 1 >maxL)
{
maxL = g[1, i].Value.ToString().Length + g[6, i].Value.ToString().Length + 1;
maxLn = g[1, i].Value.ToString() + " " + g[6, i].Value.ToString();
}
Graphics gr = Graphics.FromImage(bmp);
gr.FillRectangle(new SolidBrush(Color.White), new Rectangle(0, 0, 877, 620)); gr.DrawString($"{param[0]} лет
{param[1]} кг", new Font("Arial", 14), new
SolidBrush(Color.Black), new Point(5, 5));
int bmpCnt = 1;
{
SizeF maxPixSize = gr.MeasureString(maxLn + " 111", new Font("Arial", 12)); int curX = 20, curY = 40, curXL =
curX + (int)maxPixSize.Width; for (int i = 0; i < g.RowCount; i++)
{
if (g.RowCount % 4 != 0)
{
if ((g.RowCount % 4 == 1 && i == 2) || (g.RowCount % 4 == 2 && (i ==         2 || i == 5)) || (g.RowCount % 4 == 2
&& (i == 2 || i == 5 || i == 8))) curX = 60;
else
curX = 20;
}
string s = g[1, i].Value.ToString();
string s2 = g[6, i].Value.ToString();
gr.DrawString(s, new Font("Arial", 12), new SolidBrush(Color.Black),
new Point(curX, curY));
gr.DrawString(s2, new Font("Arial", 12), new SolidBrush(Color.Black), new Point(curXL - (int)gr.MeasureString(s2,
new Font("Arial", 12)).Width - 5, curY));
curY += 5 + (int)maxPixSize.Height;
gr.DrawLine(new Pen(Color.Black), new Point(curX, curY), new
Point(curXL, curY));
connects.Add(new Point(curXL, curY));
if (i % 2 != 0)
{
if (877 - curY < (int)maxPixSize.Height*2 + 60 && i != g.RowCount -1)
{
CreateBmpLines(ref gr, connects); bmp.Save(AppDomain.CurrentDomain.BaseDirectory + $"\\Турнирные
сетки\\{param[0]} лет {param[1]} {bmpCnt}.png"); bmp.Dispose();
bmp = new Bitmap(620, 877);
gr.FillRectangle(new SolidBrush(Color.White), new Rectangle(0,
0, 877, 620));
```

```
gr.DrawString($"{param[0]} лет {param[1]} кг", new

Font("Arial", 14), new SolidBrush(Color.Black), new Point(5, 5)); curY = 20;
bmpCnt++;
}
}
curY += 40;
}
CreateBmpLines(ref gr, connects);
if (bmpCnt == 1)
{
bmp.Save(AppDomain.CurrentDomain.BaseDirectory + $"\\Турнирные сетки\\{param[0]} лет {param[1]}.png");
bmp.Dispose();
}
else
{
bmp.Save(AppDomain.CurrentDomain.BaseDirectory + $"\\Турнирные сетки\\{param[0]} лет {param[1]}
{bmpCnt}.png");
bmp.Dispose();
}
}
}
MessageBox.Show("Графические изображения турнирных сеток созданы!");
}
private void CreateBmpLines(ref Graphics gr, List<Point> connects)
{
List<Point> buf = new List<Point>();
int p = 0;
if (connects.Count == 1)
return;
while (connects.Count > 0)
{
if (connects.Count % 4 != 0 && connects.Count > 4)
{
if (connects.Count % 2 != 0)
{
DrawLineConnect(ref gr, connects[p], connects[p + 1]); gr.DrawLine(new Pen(Color.Black), new Point(connects[p].X,
connects[p].Y + ((connects[p + 1].Y - connects[p].Y) / 2)), new Point(connects[p].X + 100, con-nects[p].Y +
((connects[p + 1].Y - connects[p].Y) / 2)));
gr.DrawLine(new Pen(Color.Black), connects[p + 2], new Point(connects[p + 2].X + 100, connects[p + 2].Y));
DrawLineConnect(ref gr, new Point(connects[p + 2].X + 100, connects[p + 2].Y), new Point(connects[p].X + 100,
connects[p].Y + ((connects[p + 1].Y - connects[p].Y) / 2)));
gr.DrawLine(new Pen(Color.Black), new Point(connects[p + 2].X + 100, connects[p + 2].Y + ((connects[p].Y +
((connects[p + 1].Y - connects[p].Y) / 2) - connects[p + 2].Y) / 2)), new Point(connects[p + 2].X + 200, connects[p +
2].Y + ((connects[p].Y + ((con-nects[p + 1].Y - connects[p].Y) / 2) - connects[p + 2].Y) / 2)));
buf.Add(new Point(connects[p + 2].X + 200, connects[p + 2].Y + ((con-nects[p].Y + ((connects[p + 1].Y -
connects[p].Y) / 2) - connects[p + 2].Y) / 2)));
connects.RemoveRange(0, 3);
}
else
{
DrawLineConnect(ref gr, connects[p], connects[p + 1]); gr.DrawLine(new Pen(Color.Black), new Point(connects[p].X,
connects[p].Y + ((connects[p + 1].Y - connects[p].Y) / 2)), new Point(connects[p].X + 100, con-nects[p].Y +
((connects[p + 1].Y - connects[p].Y) / 2)));
buf.Add(new Point(connects[p].X + 100, connects[p].Y + ((connects[p + 1].Y - connects[p].Y) / 2)));
connects.RemoveRange(0, 2);}}
else
{
if (connects.Count == 4 || connects.Count % 4 == 0)
{
gr.DrawLine(new Pen(Color.Black), connects[p], connects[p + 1]); gr.DrawLine(new Pen(Color.Black), new
Point(connects[p].X, con-
```

```
necks[p].Y + ((connects[p + 1].Y - connects[p].Y) / 2)), new Point(connects[p].X + 100, con-necks[p].Y + ((connects[p
+ 1].Y - connects[p].Y) / 2)));
gr.DrawLine(new Pen(Color.Black), connects[p + 2], connects[p + 3]);
gr.DrawLine(new Pen(Color.Black), new Point(connects[p + 2].X, con-necks[p + 2].Y + ((connects[p + 3].Y -
connects[p + 2].Y) / 2)), new Point(connects[p + 2].X + 100, connects[p + 2].Y + ((connects[p + 3].Y - connects[p +
2].Y) / 2)));
gr.DrawLine(new Pen(Color.Black), new Point(connects[p].X + 100, con-necks[p].Y + ((connects[p + 1].Y -
connects[p].Y) / 2)), new Point(connects[p + 2].X + 100, connects[p + 2].Y + ((connects[p + 3].Y - connects[p + 2].Y) /
2)));
int y1 = connects[p].Y + ((connects[p + 1].Y - connects[p].Y) / 2);
int y2 = connects[p + 2].Y + ((connects[p + 3].Y - connects[p + 2].Y) /2);
gr.DrawLine(new Pen(Color.Black), new Point(connects[p + 2].X + 100, y1 + ((y2 - y1) / 2)), new Point(connects[p +
2].X + 200, y1 + ((y2 - y1) / 2)));
buf.Add(new Point(connects[p + 2].X + 200, y1 + ((y2 - y1) / 2))); connects.RemoveRange(0, 4);
}
if (connects.Count == 3)
{
DrawLineConnect(ref gr, connects[p], connects[p + 1]); gr.DrawLine(new Pen(Color.Black), new Point(connects[p].X,
connects[p].Y + ((connects[p + 1].Y - connects[p].Y) / 2)), new Point(connects[p].X + 100, con-necks[p].Y +
((connects[p + 1].Y - connects[p].Y) / 2)));
gr.DrawLine(new Pen(Color.Black), connects[p + 2], new Point(connects[p + 2].X + 100, connects[p + 2].Y));
DrawLineConnect(ref gr, new Point(connects[p + 2].X + 100, connects[p + 2].Y), new Point(connects[p].X + 100,
connects[p].Y + ((connects[p + 1].Y - connects[p].Y) / 2)));
gr.DrawLine(new Pen(Color.Black), new Point(connects[p + 2].X + 100, connects[p + 2].Y + ((connects[p].Y +
((connects[p + 1].Y - connects[p].Y) / 2) - connects[p + 2].Y) / 2)), new Point(connects[p + 2].X + 200, connects[p +
2].Y + ((connects[p].Y + ((con-necks[p + 1].Y - connects[p].Y) / 2) - connects[p + 2].Y) / 2)));
buf.Add(new Point(connects[p + 2].X + 200, connects[p + 2].Y + ((con-necks[p].Y + ((connects[p + 1].Y -
connects[p].Y) / 2) - connects[p + 2].Y) / 2)));
connects.RemoveRange(0, 3);
}
if (connects.Count == 2)
{
DrawLineConnect(ref gr, connects[p], connects[p + 1]); gr.DrawLine(new Pen(Color.Black), new Point(connects[p].X,
connects[p].Y + ((connects[p + 1].Y - connects[p].Y) / 2)), new Point(connects[p].X + 100, con-necks[p].Y +
((connects[p + 1].Y - connects[p].Y) / 2)));
buf.Add(new Point(connects[p].X + 100, connects[p].Y + ((connects[p + 1].Y - connects[p].Y) / 2)));
connects.RemoveRange(0, 2);
}
}
if (connects.Count == 0 && buf.Count > 1)
{
connects.AddRange(buf);
buf.Clear();
}
}
}
private void DrawLineConnect(ref Graphics gr, Point p1, Point p2)
{
if (p2.X != p1.X)
{
gr.DrawLine(new Pen(Color.Black), new Point(Math.Min(p1.X, p2.X), Math.Max(p1.Y, p2.Y)), new
Point(Math.Max(p1.X, p2.X), Math.Max(p1.Y, p2.Y)));
p2.X = Math.Max(p1.X, p2.X);
p1.X = p2.X;
}
gr.DrawLine(new Pen(Color.Black), p1, p2);
}}}
```

BD

```
public override void Up()
{
CreateTable(
"dbo.Participants",
c => new
{
Id = c.Int(nullable: false, identity: true),
Name = c.String(),
User_Id = c.String(maxLength: 128),
})
.PrimaryKey(t => t.Id)
.ForeignKey("dbo.AspNetUsers", t => t.User_Id)
.Index(t => t.User_Id);
CreateTable(
"dbo.AspNetUsers",
c => new
{
Id = c.String(nullable: false, maxLength: 128),
Email = c.String(maxLength: 256),
EmailConfirmed = c.Boolean(nullable: false),
PasswordHash = c.String(),
SecurityStamp = c.String(),
PhoneNumber = c.String(),
PhoneNumberConfirmed = c.Boolean(nullable: false),
TwoFactorEnabled = c.Boolean(nullable: false),
LockoutEndDateUtc = c.DateTime(),
LockoutEnabled = c.Boolean(nullable: false),
AccessFailedCount = c.Int(nullable: false),
UserName = c.String(nullable: false, maxLength: 256),
})
.PrimaryKey(t => t.Id)
.Index(t => t.UserName, unique: true, name: "UserNameIndex");
CreateTable(
"dbo.AspNetUserClaims",
c => new
{
Id = c.Int(nullable: false, identity: true),
UserId = c.String(nullable: false, maxLength: 128),
ClaimType = c.String(),
ClaimValue = c.String(),
})
.PrimaryKey(t => t.Id)
.ForeignKey("dbo.AspNetUsers", t => t.UserId, cascadeDelete: true)
.Index(t => t.UserId);
CreateTable(
"dbo.AspNetUserLogins",
c => new
{
LoginProvider = c.String(nullable: false, maxLength: 128),
ProviderKey = c.String(nullable: false, maxLength: 128),
UserId = c.String(nullable: false, maxLength: 128),
})
.PrimaryKey(t => new { t.LoginProvider, t.ProviderKey, t.UserId })
.ForeignKey("dbo.AspNetUsers", t => t.UserId, cascadeDelete: true)
.Index(t => t.UserId);
CreateTable(
"dbo.AspNetUserRoles",
c => new
{
UserId = c.String(nullable: false, maxLength: 128),
RoleId = c.String(nullable: false, maxLength: 128),
```

```
})
.PrimaryKey(t => new { t.UserId, t.RoleId })
.ForeignKey("dbo.AspNetUsers", t => t.UserId, cascadeDelete: true)
.ForeignKey("dbo.AspNetRoles", t => t.RoleId, cascadeDelete: true)
.Index(t => t.UserId)
.Index(t => t.RoleId);
CreateTable(
"dbo.AspNetRoles",
c => new
{
Id = c.String(nullable: false, maxLength: 128),
Name = c.String(nullable: false, maxLength: 256),
})
.PrimaryKey(t => t.Id)
.Index(t => t.Name, unique: true, name: "RoleNameIndex");
CreateTable(
"dbo.TournamentGrid",
c => new
{
Id = c.Int(nullable: false, identity: true),
Content = c.String(),
AgeWith = c.Int(nullable: false),
AgeTo = c.Int(nullable: false),
WeightWith = c.Double(nullable: false),
WeightTo = c.Double(nullable: false),
Status = c.Boolean(nullable: false),
})
.PrimaryKey(t => t.Id);
AddColumn("dbo.Participants", "Status", c => c.Boolean(nullable: false));
}
```

AccountController

```
using System;
using System.Globalization;
using System.Linq;
using System.Security.Claims;
using System.Threading.Tasks;
using System.Web;
using System.Web.Mvc;
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.Owin;
using Microsoft.Owin.Security;
using Diplom.Models;
namespace Diplom.Controllers
{
[Authorize]
public class AccountController : Controller
{
private ApplicationSignInManager _signInManager;
private ApplicationUserManager _userManager;
public AccountController()
{
}
public AccountController(ApplicationUserManager userManager, ApplicationSignInManager signInManager ) {
UserManager = userManager;
SignInManager = signInManager;
}
public ApplicationSignInManager SignInManager
{
get
{
return _signInManager ?? HttpContext.GetOwinContext().Get<ApplicationSignInManager>();
```

```
}
private set
{
_signInManager = value;
}

}

public ApplicationUserManager UserManager
{
get
{
return _userManager ?? HttpContext.GetOwinContext().GetUserManager<ApplicationUserManager>();
}
private set
{
_userManager = value;
}
}
[AllowAnonymous]
public ActionResult Login(string returnUrl)
{
ViewBag.ReturnUrl = returnUrl;
return View();
}
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult> Login(LoginViewModel model, string returnUrl)
{
if (!ModelState.IsValid)
{
return View(model);
}
var result = await SignInManager.PasswordSignInAsync(model.Email, model.Password, model.RememberMe,
shouldLockout: false);
switch (result)
{
case SignInStatus.Success:
return RedirectToLocal(returnUrl);
case SignInStatus.LockedOut:
return View("Lockout");
case SignInStatus.RequiresVerification:
return RedirectToAction("SendCode", new { ReturnUrl = returnUrl, RememberMe = model.RememberMe });
case SignInStatus.Failure:
default:
ModelState.AddModelError("", "Недійсна спроба входу.");
return View(model);
}
}
[AllowAnonymous]
public ActionResult Register()
{
return View();
}
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult> Register(RegisterViewModel model)
{
if (ModelState.IsValid)
{
```

```
var user = new ApplicationUser
{
UserName = model.Email,
Email = model.Email,
FirstName = model.FirstName,
SecondName = model.SecondName,
PhoneNumber = model.Telephone,
ImgUrl = "/Images/Profile/default.png"
};
var result = await UserManager.CreateAsync(user, model.Password);
if (result.Succeeded)
{
await SignInManager.SignInAsync(user, isPersistent:false, rememberBrowser:false); await
UserManager.AddToRoleAsync(user.Id.ToString(), "Coach");
return RedirectToAction("Index", "Participants");
}
AddErrors(result);
}
return View(model);
}
```

AdminController

```
using Diplom.Models;
using Diplom.Services.Abstraction;
using Diplom.Services.Implementation;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Hosting;
using System.Web.Mvc;
namespace Diplom.Controllers
{
[Authorize(Roles = "Admin")]
public class AdminController : Controller
{
ISimpleService service = new SimpleService();
// GET: Admin
public ActionResult Index()
{
string text = Sys-
tem.IO.File.ReadAllText(Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "url.txt"));
TextImg textiimg = new TextImg();
textiimg.ImagePath = text;
textiimg.ImageOldPath = text;
return View(textiimg);
}
public ActionResult DeleteAll()
{
service.ClearAll();
return RedirectToAction("Index");
}
// public ActionResult Create([Bind(Include = "ID, ImageTitle")] Image img, HttpPost-edFileBase file)
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Index([Bind(Include = " ImageUpload, ImagePath, ImageOldPath")] TextImg participant)
{
if (participant.ImageUpload == null)
{
```

```
ModelState.AddModelError("ImageUpload", "This field is required.");
}
if (ModelState.IsValid)
{
try
{
System.IO.File.Delete(System.IO.File.ReadAllText(Path.Combine(AppDomain.CurrentDomain.BaseDirector
y, "url.txt")));
}
catch
{
}
System.IO.File.WriteAllText(Path.Combine(AppDomain.CurrentDomain.BaseDirectory,"url.txt"),participant.ImagePat
h);
return RedirectToAction("Index");
}
return View(participant);
}
public ActionResult Allparticipants()
{
var model = service.GetAllParticipents();
return View(model);
}
//[HttpPost, ActionName("Change")]
public ActionResult Change(int? id)
{
Participent participent = service.GetById(id.Value);
if (participent == null)
{
return HttpNotFound();
}
service.SetStatus(id.Value);
return RedirectToAction("Allparticipants");
}
public ActionResult Tournaments()
{
var model = service.GetTournamentGrids();
return View(model);
}
public ActionResult Statustournament(int? id)
{
var tournament = service.TournamentGetById(id.Value);
if (tournament == null)
{
return HttpNotFound();
}
service.StatusTournament(id.Value);
return RedirectToAction("Tournaments");
}
public ActionResult Detail(int? id)
{
var tournament = service.TournamentGetById(id.Value); var model = new TournamentModel {
Id = id.Value,
Content = tournament.Content
};
return View(model);
}
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Detail(TournamentModel tournament, HttpPostedFileBase image)
{
if (image!=null)
```

```
{
var tournamentId = tournament.Id;
string fileName = System.IO.Path.GetFileName(image.FileName); image.SaveAs(Server.MapPath("/Image/" +
fileName)); service.UpdateImageUrl(tournamentId, fileName);
}
return RedirectToAction("Tournaments");
}
public ActionResult Delete(int id)
{
service.DeleteByID(id);
return RedirectToAction("Allparticipants");
}
}
}
```

DrawsController

```
using Diplom.Services.Abstraction;
using Diplom.Services.Implementation;
using System.Linq;
using System.Web.Mvc;
namespace Diplom.Controllers
{
public class DrawsController : Controller
{
private int[,] ages = new int[,]
{
{1,6,7,20,25},
{2,6,7,25,30},
{3,6,7,30,70},
{4,8,9,20,30},
{5,8,9,30,35},
{6,8,9,35,40},
{7,8,9,40,75},
{8,10,11,20,35},
{9,10,11,35,40},
{10,10,11,40,45},
{11,10,11,45,90},
{12,12,13,30,40},
{13,12,13,40,45},
{14,12,13,45,50},
{15,12,13,50,55},
{16,12,13,55,80},
{17,14,15,40,55},
{18,14,15,55,60},
{19,14,15,60,65},
{20,14,15,65,80},
{21,16,17,40,60},
{22,16,17,60,65},
{23,16,17,65,70},
{24,16,17,70,75},
{25,16,17,75,90},
{26,18,18,40,70},
{27,18,18,70,75},
{28,18,18,75,80},
{29,18,18,80,85},
{30,18,18,85,120}
};
ISimpleService service = new SimpleService();
public DrawsController() { }
public ActionResult Index(int id = 1)
{
```

```
var iter = id - 1;
var ageWith = ages[iter, 1];
var ageTo = ages[iter, 2];
var weightWith = ages[iter, 3];
var weightTo = ages[iter, 4];
var model = service.GetAllParticipents()
.Where(x => x.AgeTo <= ageTo && x.AgeTo >= ageWith
&&       x.Weight >= weightWith && x.Weight <= weightTo && x.Status == true)
.ToList();
ViewBag.MyString = $" у віковій категорії {ageWith}-{ageTo} років {weightTo}кг."; return View(model);
}
}
}
```

MyParticipantsController

```
using Diplom.Services.Abstraction;
using Diplom.Services.Implementation;
using Microsoft.AspNet.Identity;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using PagedList;
using Diplom.Models;
using System.Net;
using System.IO;
using System.Globalization;
namespace Diplom.Controllers
{
[Authorize]
public class MyParticipantsController : Controller
{
ISimpleService service = new SimpleService();
int itemsPerPage=5;
// GET: MyParticipents
public ActionResult Index(int? i = 1)
{
return View(service.GetParticipentsByCoach(User.Identity.GetUserId()).ToPagedList(i
??1, itemsPerPage));
}
public ActionResult AddNew()
{
return View(new ParticipantModel());
}
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult AddNew([Bind(
Include = "ID, ImageUpload, ImagePath, Category, " +"Country, DateOfBirth, Dojo, Grade, Initsials, Organisation, " +
"Region, Age, Weight, Day, Mount, Year")] ParticipantModel participant)
{
if (participant.ImageUpload == null)
{
ModelState.AddModelError("ImageUpload", "Це поле є обов'язковим.");
}
if (ModelState.IsValid)
{
var birthDate = new DateTime(participant.Year, participant.Mount, participant.Day);
var age = DateTime.Now.Year - birthDate.Year;
var model = new ParticipantModel
{
ID = participant.ID,
```

```
ImageUpload = participant.ImageUpload,
ImagePath = participant.ImagePath,
Category = $"{age} років, {participant.Weight} кг",
Country = participant.Country,
DateOfBirth = birthDate,
Dojo = participant.Dojo,
Grade = participant.Grade,
Initsials = participant.Initsials,
Organisation = participant.Organisation,
Region = participant.Region,
AgeTo = age,
Weight = participant.Weight,
Status = false
};
service.AddParticipent(model, User.Identity.GetUserId()); return RedirectToAction("Index");
}
return View(participant);
}
public ActionResult Edit(int? id)
{
if (id == null)
{
return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
}
var part =service.ParticipentByID(id.Value);
if (part == null)
{
return new HttpNotFoundResult();
}
var model = new ParticipantModel
{
ID = part.Id,
Category = part.Category,
Country = part.Country,
DateOfBirth = part.DateOfBirth,
Dojo = part.Dojo,
Grade = part.Grade, ImagePath = part.ImagePath,
Initsials = part.Initsials, Organisation = part.Organisation, Region =
part.Region,
ImageDisplayUrl = part.ImagePath
};
return View(model);
}
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Edit(int id, ParticipantModel model)
{
var validImageTypes = new string[]
{
"image/gif",
"image/jpeg",
"image/pjpeg",
"image/png"
};
if (model.ImageUpload != null)
{
if (model.ImageUpload.ContentLength > 0)
{
if (!validImageTypes.Contains(model.ImageUpload.ContentType))
ModelState.AddModelError("ImageUpload", "Виберіть будь-яке GIF, JPG or
PNG зображення.");
}
```

```
}
if (ModelState.IsValid)
{
var image = service.ParticipentByID(id);
if (image == null)
{
return new HttpNotFoundResult();
}
if (model.ImageUpload != null && model.ImageUpload.ContentLength > 0)
{
var uploadDir = "~/images";
var imagePath = Path.Combine(Server.MapPath(uploadDir), mod-el.ImageUpload.FileName);
var imageUrl = Path.Combine(uploadDir, model.ImageUpload.FileName); model.ImageUpload.SaveAs(imagePath);
model.ImagePath = imageUrl;
}
else
{
model.ImagePath = image.ImagePath;
}
service.ChangeParticipent(model, model.ID);
return RedirectToAction("Index");
}
return View(model);
}
public ActionResult Delete(int? id)
{
if (id == null)
{
return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
}
Participent participent = service.GetById(id.Value);
if (participent == null)
{
return HttpNotFound();
}
return View(participent);
}
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public ActionResult DeleteConfirmed(int id)
{
Participent participent = service.GetById(id);
if (participent == null)
{
return HttpNotFound();
}
try
{
string sFName = HttpContext.Server.MapPath(participent.ImagePath); System.IO.File.Delete(sFName);
}
catch (Exception exc)
{
throw new HttpException(500, exc.Message);
}
service.DeleteByID(id);
return RedirectToAction("Index");
}
}
}
```

ParticipantsController

```
using Diplom.Services.Abstraction;
using Diplom.Services.Implementation;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Diplom.Models;
using Microsoft.AspNet.Identity;
using PagedList;
namespace Diplom.Controllers
{
public class ParticipantsController : Controller {
ISimpleService service = new SimpleService();
int itemsPerPage = 5;
public ActionResult Index(int? i = 1)
{
return View(service.GetAllParticipents().ToPagedList(i ?? 1, itemsPerPage));
}
}
```

ProfileControlle

```
using Diplom.Services.Abstraction;
using System.Web;
using System.Web.Mvc;
using Diplom.Models;
using Diplom.Services.Implementation;
namespace Diplom.Controllers
{
public class ProfileController : Controller
{
ISimpleService service = new SimpleService();
public ActionResult Detail(string id)
{
var user = service.GetUserById(id);
var model = new ProfileViewModel
{
Id = user.Id,
FirstName = user.FirstName,
SecondName = user.SecondName,
Telephone = user.PhoneNumber,
ImgUrl = user.ImgUrl,
Email = user.Email
};
return View(model);
}
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Edit(ProfileViewModel user, HttpPostedFileBase image)
{
if (image != null)
{
var userId = user.Id;
string fileName = System.IO.Path.GetFileName(image.FileName); image.SaveAs(Server.MapPath("/Images/Profile/" +
fileName)); service.UpdateUserDescription(user, fileName);
}
return RedirectToAction("Index", "MyParticipants");
}
public ActionResult About()
{
return View();
```

```
}
public ActionResult Contact()
{
return View();
}
public ActionResult Help()
{
return View();
}
}
}
```

TournamentGridController

```
using Diplom.Services.Abstraction;
using Diplom.Services.Implementation;
using System;
using System.Web.Mvc;
using System.IO;
using Diplom.Models;
namespace Diplom.Controllers
{
public class TournamentGridController : Controller {
ISimpleService service = new SimpleService();
public ActionResult Index(int id=1)
{
var model = service.TournamentGetById(id);
if (model.Status == false)
{
return View("EmptyIndex");
}
return View(model);
}
}
}
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Диплом_ doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом_.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація_.ppt	Презентація кваліфікаційної роботи