ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

БД – база даних

ПЗ – програмне забезпечення

ПП – програмний продукт

СУБД – система управління базами даних

ТЗ – технічне завдання

ХО – хвилинний обсяг

ЧСС – частота серцевих скорочень

CLR – Common Language Runtime

IDE – Integrated Development Environment, інтегроване середовище розробки.

WAL – Write-Ahead Logging, протокол

ВСТУП

В час активного розвитку інноваційних технологій і появи великої кількості різноманітних гаджетів фізична активність людей стала знижуватись, так як життя дуже спростилося.

Але ні дивлячись ні на що, в XXI столітті дуже актуальним є здоровий спосіб життя. І щоб підтримувати своє здоров’я або досягати якихось

спортивних цілей люди займаються спортом. Одним з найпопулярніших видів спорту, котрий допомагає підтримувати форму є біг, так як він є бюджетним та ефективним видом спорту.

Легка атлетика − олімпійський вид спорту, який об’єднує в собі багато дисциплін(такі як біг, спортивна ходьба, легкоатлетичне багатоборство, стрибки та метання).ЇЇ прийнято називати "Королевою спорту", так як це один з найдавніших і наймасовіших видів.

Наявність сучасних технологій дає змогу зберігати та систематизувати інформацію про учасників і їх результати, дані про тренерів, місцях проведення змагань, регламент та протоколи та інші дані в єдиній базі, що значно полегшує організацію змагань.

Актуальність моєї роботи полягає в словесному проханні в наданні кафедрі фізичного виховання та валеології Одеського державного екологічного університету програми для підготовки та відстеження результатів спортсменів-легкоатлетів.

Метою даної дипломної роботи є моделювання і розробка системи, яка зможе оцінювати фізичну підготовленість спортсмена-легкоатлета до змагань та допоможе тренерам слідкувати за всіма результатами з одного приладу. Додаток повинен отримувати дані про результати спортсмена та зрівнювати їх з відповідними нормами для даної категорії, віку та статі атлета і виводити інформацію про те, готовий саме цей спортсмен до змагань чи ні.

Задачею дипломної роботи є проектування та розробка застосування для відстеження підготовки спортсменів-легкоатлетів до змагань та бази даних (БД) до нього.

За результатами роботи була опублікована стаття[1][1)].

Представлений дипломний проект містить 54 сторінки, 23 рисунків, 19 посилань, 3 таблиці та 15 листів додатків.

[1)] [1] Розробка системи контролю підготовки спортсменів-легкоатлетів до змагань. Збірник статей за матеріалами студентської наукової конференції Одеського державного екологічного університету (15-18 квітня 2019 р.). Одеса. 2019. С. 158–162.

1 АНАЛІТИЧНА ЧАСТИНА

1.1 Аналіз предметної області

Кінцевий програмний продукт – це програмний комплекс, який допомагає тренерам відстежувати спортивні показники атлетів і визначати їх готовність до змагань.

ПП включатиме в себе місце збереження всіх необхідних даних, а також програми-настільного застосування(desktop application), яке відображатиме ці дані і визначає готовність атлета до змагання.

Підставою для проведення розробки, яка є вмістом даної дипломної роботи було прохання кафедри фізичного виховання та валеології Одеського державного екологічного університету про створення програми, яка б допомагала тренеру відстежувати і контролювати ступінь підготовки спортсмена-легкоатлета до змагань.

Під час розробки ПП повинні виконуватися наступні вимоги:

- ПП повинен розраховувати підготовленість спортсменів до змагань у різні періоди підготовки ;
- ПП повинен мати графічний інтерфейс користувача ;
- ПП повинен мати БД, необхідну для комфортного користування ПП.

Актуальність цієї роботи полягає в тому, що всіх спортсменів-легкоатлетів можна поділити на 3 категорії:

- початківців;
- спортсмени середньої кваліфікації;
- спортсмени високого рівня.

Під час вивчення предметної області, було зроблено висновок, що у спортсмена середнього рівня є один чи два тренера без ретельного медичного супроводження.

Хоча у літературі показано багато показників за якими можна контролювати готовність організму спортсмена у різні періоди підготовки до змагань. З яких було виділено 13 показників.

Збільшення хвилинного обсягу (ХО) крові – у нормі величина хвилинного об'єму, за даними механокардіографічного методу, коливається в межах від 3л до 6л. В середньому нормальна величина ХО в спокої становить 3,5л–5,5л. При фізичних навантаженнях хвилинний обсяг серця може досягати 18л–28л і навіть 30л [2][1)].

Життєва ємність легенів – життєва ємність легень вимірюється спірометром. У спортсменів величина зможе коливатися від 4500мл до 8000мл у чоловіків і від 3500мл до 5300мл у жінок [3][2)].

Підвищення гемоглобіну – норма гемоглобіну в крові у чоловіків – 140грамів на літр (г/л) – 160г/л (16–18 років – 117г/л – 166г/л, 19–45 років – 132 г/л –173г/л), у жінок – 120г/л – 150г/л (15–18 років – 117г/л –153г/л, 19–45 років – 117г/л – 155г/л) [4][3)].

Частота серцевих скорочень (ЧСС) – у атлетів, що регулярно тренуються (бігунів, плавців, велосипедистів) пульс у спокої може бути нижче 60–50 і навіть нижче 40 ударів в хвилину (уд. / хв). Хорошим показником тренованості є ЧСС в стані спокою. Пульс в стані спокою, рівний 48 уд. / хв – 60 уд. / хв [5][4)].

Підвищується якість сили (перевіряється за допомогою еспандера).

Станова сила – збільшення кількості повторень і підходів, і можливе підвищення ваги.

Покращується показник вибухової сили (швидкий прояв руху в одиницю часу. Наприклад, стрибок з місця, тест Абалакова)

Високі результати проби Абалакова говорять про те, що в м'язах переважають сильні швидкі волокна, пристосовані до короткочасної силовий і

[1)] [2] Минутный объём кровообращения. URL: http://gemodinamika.ru/minutnyi-objem-krovoobraschenija.html (дата звернення 30.03.2019).

[2)] [3] Объём легких у спортсменов. URL: http://avtobaiki.ru/raznoe/obem-legkih-u-sportsmenov.html (дата звернення 01.04.2019).

[3)] [4] Норма гемоглобина. URL: https://news.sportbox.ru/doppelherz_2013/vbz/spbnews_NI389619_Zagadka-gemoglobina-Norma-visokiy-ili-nizkiy (дата звернення 17.01. 2019).

[4)] [5] Норма пульса на тренировке. Важнейшие сведения о пульсе человека. URL: http://ggym.ru/view_post.php?id=77 (дата звернення 23.03.2019).

швидкісно-силової роботи. Низькі результати проби говорять про протилежне – переважання повільних аеробних волокон, пристосованих до тривалої роботи на витривалість.

Таблиця 1 – Результати тесту Абалакова[6][1)]

Вік	Висота стрибку з місця(випригування: різниця між першою і другою відмітрою(см)	
	жінки	чоловіки
20-29	24-32	35-45
30-39	22-30	30-40
40-49	20-28	27-36
50-59	17-25	23-32
60-70	12-20	20-28

Уповільнення процесів окислення (перевіряється за допомогою лакмусового папірця – зсув РН в кислий бік менше).

Двадцятихвилинний біг (порівнюється час за кілометр) – щоб виконати цей тест, необхідно мати, як мінімум, два значення. Значення порівнюються між собою, якщо друге значення краще попереднього – тобто поліпшення, а якщо гірше – спортсмен не готовий за цим критерієм.

Тест Купера – 12 хвилинний біг, тест проводиться раз на місяць (див. таб.1). Цей тест був розроблений Кеннетом Купером для жінок та чоловіків в віці від 18 до 35 років.

Жим лежачи (на руки-динаміка зростання) – щоб виконати цю вправу і визначити чи є зростання чи ні, необхідно мати, як мінімум, два значення. Збільшення кількості повторень і підходів, і можливе підвищення ваги .

Жим лежачи (на ноги-динаміка зростання) – так само, як і жим лежачи на руки, треба мати, як мінімум два значення, для того, щоб можна було визначити є зростання чи ні.

[1)] [6] Тест на прыгучесть – Студопедия. URL: https://studopedia.ru/14_17385_test-na-priguchest.html (дата звернення 20.01.2019).

Таблиця 2 – Результати тесту Купера[7][1)]

Вік	стать	Дуже добрий	Добрий	Середній	Низький	Дуже низький
13-14	ч	2700+ м	2400-2700м	2200-2399м	2100-2199м	2100- м
	ж	2000+ м	1900-2000м	1600-1899м	1500-1599м	1500- м
15-16	ч	2800+ м	2500-2800м	2300-2499м	2200-2299м	2200- м
	ж	2100+ м	2000-2100м	1700-1999м	1600-1699м	1600- м
17-20	ч	3000+ м	2700-3000м	2500-2699м	2300-2499м	2300- м
	ж	2300+ м	2100-2300м	1800-2099м	1700-1799м	1700- м
20-29	ч	2800+ м	2400-2800м	2200-2399м	1600-2199м	1600-м
	ж	2700+ м	2200-2700м	1800-2199м	1500-1799м	1500- м

Критерій кругового тренування (8 секунд вправ-8 секунд відпочинку, 15 секунд вправ – 15 секунд відпочинку, 20 секунд вправ – 20 секунд відпочинку – параметри збільшення кількості повторів) – щоб виконати цю вправу і визначити чи є зростання чи ні, необхідно мати, як мінімум, два значення. Збільшення кількості повторень і буде зростанням спортсмена.

1.2 Аналіз ринку подібних проектів

Був проведений моніторинг існуючого програмного забезпечення (ПЗ), призначеного для допомоги спортсменам і їх тренерам, в їх роботі. В резуль-

[1)] [7] Тест Купера (Бег) : этапы теста, нормативы, таблицы. URL: http://kakbegat.com/24-test-kupera-beg.html (дата звернення 01.06. 2019).

таті, було виявлено, що є багато програм для групи початківців, прикладом таких програм можуть слугувати Fitness Point [9][1] та Runkeeper [8][2].

1.2.1 Розгляд застосування Fitness Point

Fitness Point – це сучасне застосування, для людей котрі займаються в спортивних залах.Орієнтований він на досвідченних спортсменів, або тих хто має серйозні наміри в тренуваннях.

На рис. 1.1 наведений загальний вигляд інтерфейсу застосування.

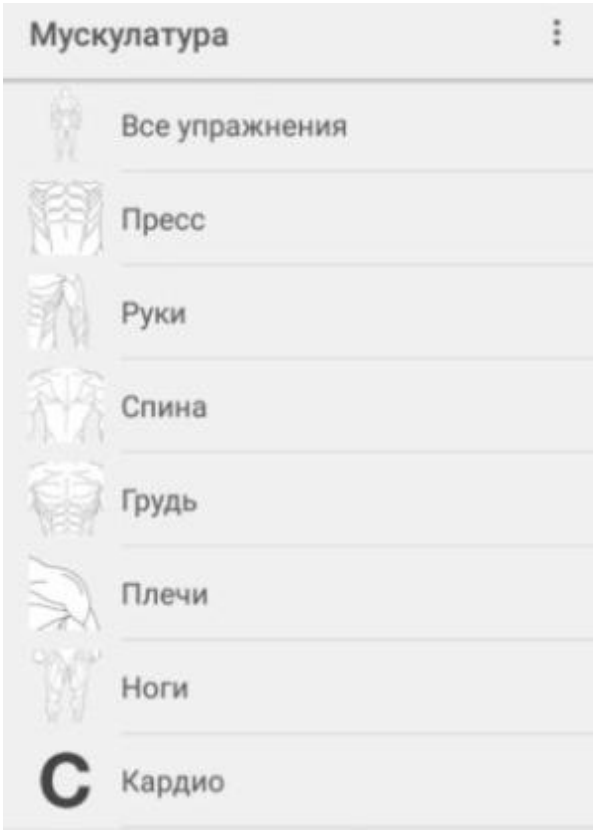

Рисунок 1.1 – Список вибору вправ

[1] [9] Скачать Fitness Point 2.6.1 для Android. URL: https://trashbox.ru/link/fitness-point-android (дата звернення 19.03.2019).

[2] [8] Runkeeper 9.9 для Android. URL: https://runkeeper.ru.uptodown.com/android (дата звернення 20.03.2019).

На другій вкладці (див. рис 1.2) можна складати, змінювати та видаляти план тренувань, або використати приклад занять

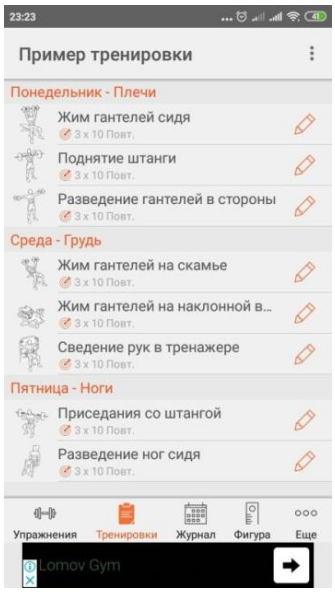

Рисунок 1.2 – Приклади тренувань

В третій вкладці (див. рис1.3) календар тренувань, в якому при натисканні на вибрану дату показується, які вправи були виконані.

Рисунок 1.3 – Календар тренувань

В четвертій вкладці (рис.1.4) спортсмен може записати параметри своєї фігури і слідкувати за її змінами.

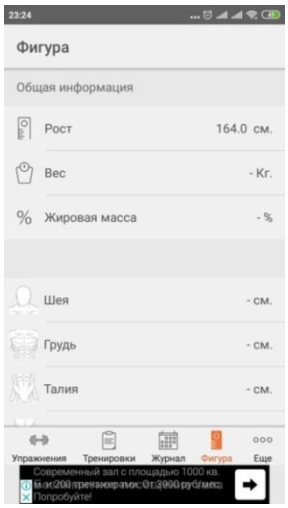

Рисунок 1.4 – Вікно антропоментричних замірів

В п’ятій вкладці (рис.1.5) користувачеві пропонують зареєструватися, ввійти в систему, або авторизуватися через соціальну мережу Facebook.

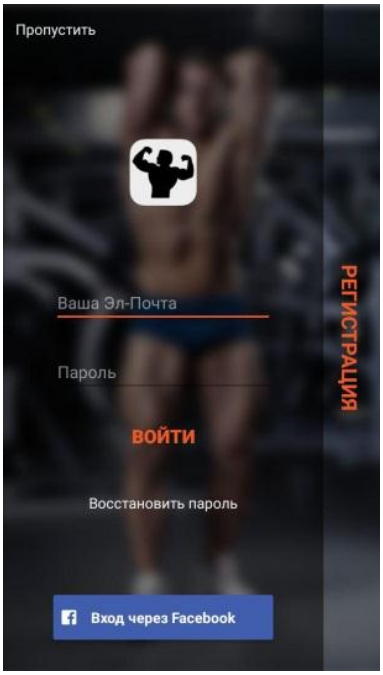

Рисунок 1.5 – Вікно авторизації устаткування

Із мінусів можна виділити велику кількість реклами, яка відволікає від основного призначення додатку та присутність платних функцій, що робить його малодоступним .

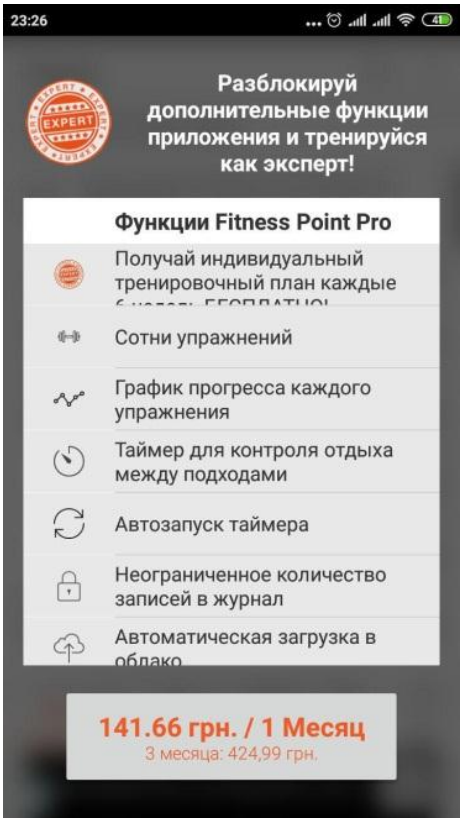

Рисунок 1.6 – Приклад реклами в устаткуванні

1.2.2 Опис застосування Runkeeper

Runkeeper – застосування створенне для любителів бігу,професійних атлетів та для людей полюбляючих їзду на велосипеді.

Перший недолік цього додатку, що їм не можна користуватись без авторизації. Але як і в минулому прикладі можно авторизуватися через соц. мережу Facebook.

На рис. 1.7 представлене вікно авторизації користувача в застосуванні, на якому розміщений мотиваційний заклик, щоб люди не стояли на місці та займалися спортом.

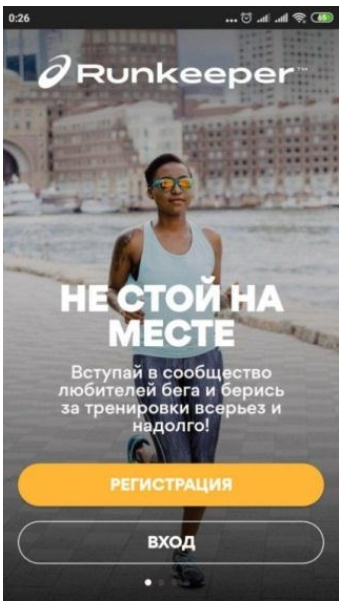

Рисунок 1.7 – Вікно авторизації в Runkeeper

Тут ми бачимо складніший та яскравий інтерфєйс (див. рис.1.8). Так само, як і в Fitness Point, можна заповнити дані про себе. Можна ставити цілі на вибранний період,створювати групу зі своїми друзями і виконувати одну ціль, або ж мірятись силами. Підримує майже всі види пульсометрів.

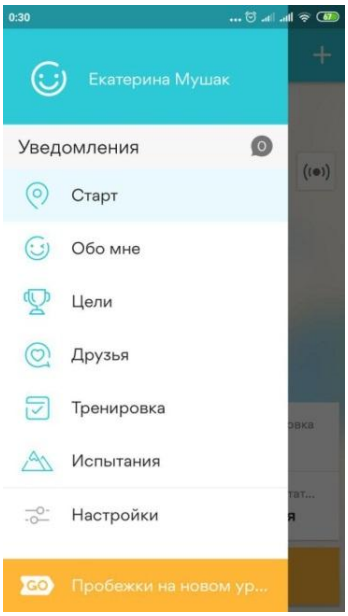

Рисунок 1.8 – Вікно користувача

Можна подивитися скільки кілометрів користувач подолав за обраний період часу(див. рис.1.9).

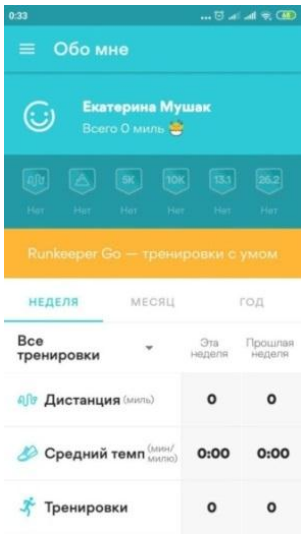

Рисунок 1.9 – Режим тренування

На рисунку 1.10 зображений вибір одного з запропонованих тренувать(такий режим підійде для новачків), або виконувати своє(для досвідченних спортсменів).

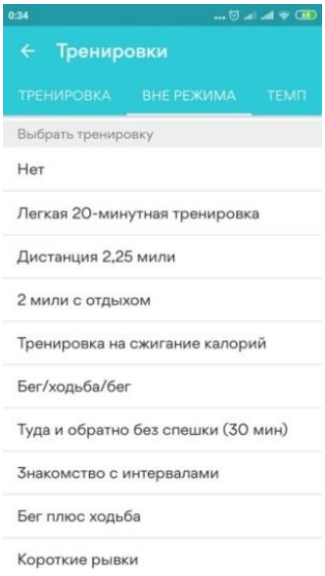

Рисунок 1.10 – Вибір тренування

Як і в Fitness Point, Runkeeper має багато реклами та платних функцій (див. рис.1.11). Але тут можна підключити режим прослуховування музики та аудіоозвучення подробиць про дистанцію та темп бігу.

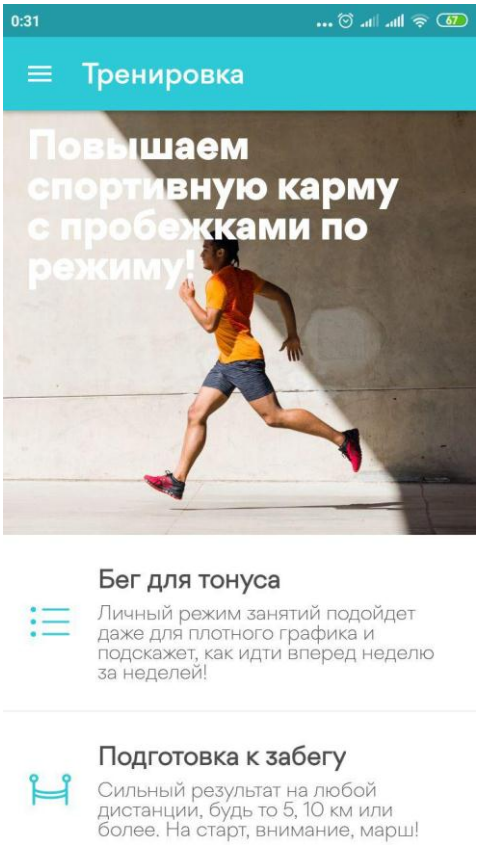

Рисунок 1.11 – Режими тренування

Всі ці застосування мають один великий мінус, вони сугубо індивідуальні і практично не дають змогу тренерам проводити контроль ступеню підготовки спорптсмена.

З іншого боку, спортсменів вищої кваліфікації, при підготовки до чемпіонатів Європи, Світу та Олімпійських Ігор, оточують команди тренерів, цілі групи лікарів. Нажаль, в вільному доступі немає того ПЗ, яким користуються всі ці фахівці, тому ми не можемо зробити їх ретельний аналіз.

Нажаль, така ж сама ситуація з відсутністю відповідного ПЗ, складається з спортсменів середнього рівня. Тому метою нашої роботи є створення є ПЗ для них.

1.3 Аналіз інструментів розробки

Як згадувалося у вступі (стор.6), основними завданнями при розробці ПП були:

- огляд платформи, на якій буде розроблятися ПП;
- СУБД для створення БД ПП;
- середовище розробки(IDE), у якому буде проводитися розробка ПП.

1.4 Вибір платформи розробки

При виборі платформи були розглянуті та проведений порівняльний аналіз віртуальних машин Java і .NET.

1.4.1 Опис Java

Програмна платформа Java – ряд програмних продуктів і специфікацій компанії Sun Microsystems, раніше незалежної компанії, а нині дочірньої компанії корпорації Oracle, які спільно надають систему для розробки прикладного програмного забезпечення та вбудовування її в будь-який кросплатформенне програмне забезпечення. Java використовується в самих різних комп'ютерних платформах від вбудованих пристроїв і мобільних телефонів в нижньому ціновому сегменті, до корпоративних серверів і суперкомп'ютерів у вищому ціновому сегменті.

Технологія Java-аплетів стала рідко використовуваної в настільних комп'ютерах, проте вона іноді використовується для поліпшення функціональності і підвищення безпеки при перегляді всесвітньої павутини.

Програмний код, написаний на Java, віртуальна машина Java виконує байт-код Java. Однак є компілятори байт-коду для інших мов програмування, таких як Ada, JavaScript, Python, і Ruby. Також є кілька нових мов програмування, розроблених для роботи з віртуальною машиною Java. Це

такі мови як Scala, Clojure and Groovy. Синтаксис Java (англ.) В основному запозичений з Сі і C++, але об'єктно-орієнтовані можливості засновані на моделі, використовуваної в мовах програмування Smalltalk і Objective-C. В Java відсутні певні низькорівневі конструкції, такі як покажчики, також Java має дуже просту модель пам'яті, де кожен об'єкт розташований в купі і всі змінні об'єктного типу є посиланнями. Управління пам'яттю здійснюється за допомогою інтегрованої автоматичної збірки сміття, яку виконує JVM.

Компанія Sun Microsystems зробила велику частину своєї реалізації Java доступною відповідно до GNU General Public License (GPL), хоча деякі частини поставляються в скомпільованому вигляді через питання авторського права з кодом, на який має ліцензію, але не право власності, компанія Sun Microsystems.

Програмна платформа Java – це ім'я для пакета програм компанії Sun, які дозволяють розробляти і запускати програми, написані на мові програмування Java. Ця програмна платформа не є специфічною для якого-небудь одного процесора або операційної системи, але механізм виконання (званий віртуальною машиною) і компілятор з набором бібліотек, які реалізовані для різного апаратного забезпечення і різних операційних систем, щоб Java-програми могли працювати скрізь однаково.

По-перше, існує технологія Java Card, яка дозволяє невеликим Java-застосуванням (аплетам) надійно працювати на смарт-картах та інших подібних пристроїв с малим об'ємом пам'яті.

По-друге, є підмножина платформи Java – Java ME, яка включає в себе кілька різних наборів бібліотек (відомих як профілі) для пристроїв з обмеженим обсягом місця для зберігання, невеликим розміром дисплея і батареї.Часто використовується для розробки додатків для мобільних пристроїв, КПК, ресиверів цифрового телебачення і принтерів.

По-третє, існує базова платформа програмування Java – Java SE, яка використовується на настільних ПК, серверах і іншому подібному обладнанні.

В решті-решт, Java EE – це Java SE плюс API, корисне для багаторівневих клієнт-серверних бізнес-додатків[11][1)].

1.4.2 Опис .NET Framework

Платформа .NET Framework – це керована виконавча для ОС Windows, що надає різноманітні служби для запуску в ній додатків. Вона складається з двох основних компонентів: середовища CLR – механізму, керуючого виконуються додатками, і бібліотеки класів .NET Framework – бібліотеки перевіреного коду, призначеного для повторного використання, який розробники можуть викликати зі своїх додатків. Нижче перераховані служби, які надає .NET Framework виконуваних у ній додатків.

По-перше, у багатьох мовах програмісти повинні передбачати виділення і звільнення пам'яті, а також керувати часом життя об'єктів. Управління пам'яттю у додатках .NET Framework виконує середа CLR.

По-друге, у традиційних мовах програмування базові типи визначаються компілятором, що ускладнює взаємодію між мовами. В .NET Framework базові типи визначаються системою типів .NET Framework, при цьому для всіх мов .NET Framework використовуються одні й ті ж базові типи.

По-третє, розробникам не потрібно писати код для виконання стандартних низькорівневих операцій програмування, так як вони використовують зручну бібліотеку типів і членів, що входить в бібліотеку класів .NET Framework.

По-четверте, платформа .NET Framework включає бібліотеки для конкретних областей розробки додатків, наприклад ASP.NET для веб-додатків, ADO.NET для доступу до даних, Windows Communication Foundation для додатків, орієнтованих на служби, а також Windows Presentation Foundation для класичних додатків Windows.

[1)] [11] Java программная платформа – Википедия. URL: https://ru.wikipedia.org/wiki/Java_(программная_платформа) (дата звернення 30.01.2019).

Крім того, мовні компілятори, орієнтовані на .NET Framework, видають проміжний код, званий мовою CIL (Common Intermediate Language), який в свою чергу компілюється під час виконання середовищем CLR. За допомогою цієї функції підпрограми, написані однією мовою, доступні в інших мовах, тому розробники можуть створювати додатки на бажаних мовами.

Можна сказати, що додатки, розроблені на основі конкретної версії платформи .NET Framework, можуть виконуватися без доробок і на більш пізніх версіях платформи.

Оскільки, платформа .NET Framework допомагає вирішувати конфлікти версій, оскільки на комп'ютері можуть бути встановлені кілька версій середовища CLR. Це означає, що кілька версій додатків можуть співіснувати і що додаток може виконуватися на версії платформи .NET Framework, для якої воно було створено. Паралельне виконання застосовується до груп версій .NET Framework 1.0 / 1.1, 2.0 / 3.0 / 3.5 і 4 / 4.5.x / 4.6.x / 4.7.x / 4.8.

В заключення, під час налаштування відповідно до стандарту .NET розробники створюють бібліотеки класів, які працюють на різних платформах .NET Framework, підтримуваних відповідної версією стандарту. Наприклад, бібліотеки, розроблені відповідно до стандарту .NET 2.0, можуть використовуватися додатками, орієнтованими на платформи .NET Framework 4.6.1, .NET Core 2.0 і UWP 10.0.16299[10] [1)].

За результатами проведення аналізу було прийнято рішення, розробляти ПП використовуючи платформу .NET Framework і з нею мову C#. Вибір платформи обґрунтовано тим, що кінцевому користувачу на потрібно встановлювати додаткове ПЗ, таке як JVM, на відміну від .NET Framework, яке з «коробки» йде разом із CLR при встановленні ОС Windows.

[1)] [10] Начало работы с .NET Framework | Microsoft Docs. URL: https://docs.microsoft.com/ru-ru/dotnet/framework/get-started/ (дата звернення 30.01.2019).

1.5 Вибір СУБД

1.5.1 Опис Microsoft SQL Server

Microsoft SQL Server – комерційна система керування базами даних, що розповсюджується корпорацією Microsoft. Мова, що використовується для запитів – Transact-SQL, створена спільно Microsoft та Sybase. Transact-SQL є реалізацією стандарту ANSI / ISO щодо структурованої мови запитів SQL із розширеннями. Використовується як для невеликих і середніх за розміром баз даних, так і для великих баз даних масштабу підприємства. Багато років вдало конкурує з іншими системами керування базами даних.

Microsoft SQL Server як мову запитів використовує версію SQL, що отримала назву Transact-SQL (скорочено T-SQL), яка є реалізацією SQL-92 (стандарт ISO для SQL) з багатьма розширеннями. T-SQL дозволяє використовувати додатковий синтаксис процедур, що зберігаються і забезпечує підтримку транзакцій (взаємодія бази даних з керуючим застосунком). Microsoft SQL Server та Sybase ASE для взаємодії з мережею використовують протокол рівня застосунка під назвою Tabular Data Stream (TDS, протокол передачі табличних даних).

Microsoft SQL Server також підтримує Open Database Connectivity (ODBC) − інтерфейс взаємодії застосунків з СУБД. Версія SQL Server 2005 надає можливість підключення користувачів через веб-сервер-сервіси, що використовують протокол SOAP. Це дозволяє клієнтським програмам, не призначеним для Windows, кроссплатформенно з'єднуватися з SQL Server. Microsoft також випустила сертифікований драйвер JDBC, що дозволяє застосункам під керування Java (таким як BEA і IBM Websphere) з'єднуватися з Microsoft SQL Server 2000 і 2005.

SQL Server підтримує дзеркалювання та кластеризацію баз даних. Кластер серверу SQL − це сукупність однаково конфігурованих серверів; така схема допомагає розподілити робоче навантаження між декількома серверами. Усі сервери мають одне віртуальне ім'я, а дані розподіляються за

IP-адресами машин кластеру протягом робочого циклу. Також у разі відмови або збою на одному з серверів кластеру доступне автоматичне перенесення навантаження на інший сервер.

SQL Server підтримує надлишкове дублювання даних за трьома сценаріями:

– знімок − виконується «знімок» бази даних, який сервер відправляє одержувачам;
– історія змін − всі зміни бази даних безперервно передаються користувачам;
– синхронізація з іншими серверами − бази даних декількох серверів синхронізуються між собою. Зміни усіх баз даних відбуваються незалежно на кожному сервері, а під час синхронізації відбувається звірка даних. Дублювання такого типу передбачає можливість вирішення протиріч між базами даних.

SQL Server 2005 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.

Microsoft та інші компанії пропонують велику кількість програмних засобів розробки, які дозволяють розробляти застосунки для бізнесу з використанням баз даних Microsoft SQL Server. Microsoft SQL Server 2005 включає також Common Language Runtime (CLR) Microsoft .NET, що дозволяє застосункам, розробленим на мовах платформи .NET (наприклад, VB.NET або C#), реалізовувати процедури, що зберігаються та різні функції.

Попередні версії засобів розробки Microsoft використовували лише API для надання функціонального доступу до Microsoft SQL Server[14][1)].

1.5.2 Опис MySQL

MySQL – це реляційна система управління базами даних з відкритим вихідним кодом. В даний час ця СУБД одна з найбільш популярних в веб-додатках – переважна більшість CMS використовує саме MySQL (часто тільки її, без альтернатив), а майже всі веб-фреймворки підтримують MySQL вже на рівні базової конфігурації (без додаткових модулів).

З переваг СУБД MySQL слід відзначити простоту використання, гнучкість, низьку вартість володіння (щодо платних СУБД), а також масштабованість і продуктивність.

MySQL дозволяє зберігати цілочисельні значення зі знаком і беззнакові, довжиною в 1, 2, 3, 4 і 8 байтів, працює із строковими і текстовими даними фіксованої і змінної довжини, дозволяє здійснювати SQL-команди SELECT, DELETE, INSERT, REPLACE і UPDATE, забезпечує повну підтримку операторів і функцій в SELECT- і WHERE- частинах запитів, працює з GROUP BY і ORDER BY, підтримує групові функції COUNT, AVG, STD, SUM, MAX і MIN, дозволяє використовувати JOIN в запитах, в т.ч. LEFT OUTER JOIN і RIGHT OUTER JOIN, підтримує реплікацію, транзакції, роботу з зовнішніми ключами і каскадні зміни на їх основі, а також забезпечує багато інших функціональні можливості.

Гнучкість СУБД MySQL забезпечується підтримкою великої кількості типів таблиць: користувачі можуть вибрати як таблиці типу MyISAM, що підтримують повнотекстовий пошук, так і таблиці InnoDB, що підтримують транзакції на рівні окремих записів. Є й інші типи таблиць, розроблені спільнотою.

[1)] [14] Microsoft SQL Server – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Microsoft_SQL_Server (дата звернення 05.02.2019)

СУБД MySQL з'явилася в 1995. Написана на C і C ++, протестована на безлічі різних компіляторів і працює на різних платформах. С 2010 року розроблення та підтримку MySQL здійснює корпорація Oracle. Продукт поширюється як під GNU GPL, так і під власною комерційною ліцензією. Однак за умовами GPL, якщо яка-небудь програма включає вихідні коди MySQL, то і ця програма теж повинна розповсюджуватися за ліцензією GPL. Для небажаючих відкривати вихідні тексти своїх програм якраз передбачена комерційна ліцензія, яка, на додаток до можливості розробки під «закритою» ліцензією, забезпечує якісну сервісну підтримку. Спільнотою розробників MySQL створені різні відгалуження - Drizzle, OurDelta, Percona Server і MariaDB, всі ці відгалуження вже існували на момент отримання прав на MySQL корпорацією Oracle. Зараз MySQL разом з Форком MariaDB займають почесне перше місце, а слідом за ними йде PostgreSQL. Решта СУБД в веб-проектах використовуються значно рідше[12][1)].

1.5.3 Опис SQLite

SQLite – компактна вбудована реляційна база даних. Вихідний код бібліотеки переданий в суспільне надбання. Є чисто реляційної базою даних.

Слово «вбудовується» означає, що SQLite не використовує парадигму клієнт-сервер. Тобто движок SQLite не є окремо працюючим процесом, з яким взаємодіє програма, а надає бібліотеку, з якої програма компонується і движок стає складовою частиною програми. Таким чином, в якості протоколу обміну використовуються виклики функцій (API) бібліотеки SQLite. Такий підхід зменшує накладні витрати, час відгуку і спрощує програму. SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси і дані) в єдиному стандартному файлі на тому комп'ютері, на якому виповнюється програма. Простота реалізації досягається за рахунок того, що

[1)] [12] MySql – система управления базы данных – « Веб Креатор». URL: https://web-creator.ru/articles/mysql (дата звернення 30.01.2019).

перед початком виконання транзакції записи весь файл, який зберігає базу даних, блокується; ACID – функції досягаються в тому числі за рахунок створення файлу журналу.

Кілька процесів або потоків можуть одночасно без будь-яких проблем читати дані з однієї бази. Запис в базу можна здійснити тільки в тому випадку, якщо ніяких інших запитів в даний момент не обслуговується; в іншому випадку спроба запису закінчується невдачею, і в програму повертається код помилки. Іншим варіантом розвитку подій є автоматичне повторення спроб запису протягом заданого інтервалу часу. Можна, також, ввести таймаут операцій. Тоді підключення, зіткнувшись із зайнятістю БД, чекатиме N секунду до того, як відвалитися з помилкою SQLITE_BUSY.

Також з версії 3.7.0 присутній режим WAL (Write-Ahead Logging), за допомогою якого можна іспольовать одну і ту ж базу кількома додатками, як на читання, так і на запис.

Завдяки архітектурі можливо використовувати SQLite як на вбудовуваних системах, так і на виділених машинах з гігабайтними масивами даних.

Формат файлу бази даних є крос-платформних, що дозволяє без проблем використовувати одну і ту ж базу на декількох операційних системах. Також присутня можливість зберігання бази в пам'яті, без її записи на диск. Цей варіант використовується за умовчанням для консольної утиліти sqlite3, якщо не вказано ім'я файлу[13][1)] .

В результаті вибору СУБД можна зробити висновок, що SQLite краще підходить для розобки застосування ніж MS SQL Server, тому що база даних перебуває ніби на локальному сервер і коли користувач коли запускає додаток йому вилетить помилка, що не може знайти сервер баз даних. У разі якщо працювати з SQLite, то у нас буде кишенькова БД, якій і сервер баз даних не потрібний. Щоб використовувати MS SQL Server, треба буде

[1)] [13] SQLite – базы данных. URL: https://lecturesdb.readthedocs.io/databases/sqlite.html#id9 (дата звернення 30.01.2019).

розмістити БД на сервері баз даних і всі програми підключені в одну локальну (чи іншу) мережу.

1.6 Вибір IDE

1.6.1 Розгляд MonoDevelop

MonoDevelop (також відомий як Xamarin Studio) – це комплексне середовище розробки з відкритим кодом для Linux, MacOS і Windows. Основна увага приділяється розробці проектів, що використовують рамки Mono та. MonoDevelop об'єднує функції, подібні до функцій NetBeans і Microsoft Visual Studio, такі як автоматичне завершення коду, керування джерелами, графічний інтерфейс користувача та веб-дизайнер. MonoDevelop об'єднує дизайнера графічного інтерфейсу Gtk# під назвою Stetic. Він підтримує Boo, C, C++, C#, CIL, D, F#, Java, Oxygene, Vala, JavaScript, TypeScript і Visual Basic.NET.

MonoDevelop можна використовувати на Windows, MacOS і Linux. Офіційно підтримувані дистрибутиви Linux включають CentOS, Debian, Fedora, openSUSE, SUSE Linux Enterprise, Red Hat Enterprise Linux і Ubuntu, причому багато інших дистрибутивів забезпечують власні неофіційні збірки MonoDevelop у своїх сховищах. MacOS і Windows офіційно підтримуються з версії 2.2.

MonoDevelop включила компілятор C# (альтернативу MSBuild і CSC) ще з найдавніших версій. В даний час він включає компілятор, який підтримує C# 1.0, C# 2.0, C# 3.0, C# 4.0, C# 5.0 і C# 6.0.

Індивідуальна версія MonoDevelop раніше поставлялася з версіями Windows і Mac Unity, ігровим движком Unity Technologies. Це дозволило розширити сценарії C#, які використовувалися для компіляції

міжплатформових відеоігор компілятором Unity. З тих пір вона була замінена спільнотою Visual Studio, за винятком версій Linux[15][1)] .

1.6.2 Розгляд Eclipse

Eclipse – вільне модульнеінтегроване середовище розробки програмного забезпечення. Розробляється і підтримується Eclipse Foundation і включає проекти, такі як платформа Eclipse, набір інструментів для програмістів на мові Java, системи контролю версій, конструктори GUI тощо. Написаний в основному на Java, може бути використаний для розробки застосунків на Java і, за допомогою різних плагінів, на інших мовах програмування, включаючи Ada, C, C++, COBOL, Fortran, Perl, PHP, Python, R, Ruby, Scala, Clojure та Scheme. Середовища розробки зокрема включають Eclipse ADT (Ada Development Toolkit) для Ada, Eclipse CDT для C/C++, Eclipse JDT для Java, Eclipse PDT для PHP. Eclipse насамперед повноцінна Java IDE, націлена на групову розробку, має засоби роботи з системами контролю версій (підтримка CVS входить у поставку Eclipse, активно розвиваються кілька варіантів SVN модулів, існує підтримка VSS та інших). З огляду на безкоштовність, у багатьох організаціях Eclipse – корпоративний стандарт для розробки ПЗ на Java.

Друге призначення Eclipse – служити платформою для нових розширень. Такими стали C/C++ Development Tools (CDT), розроблювані інженерами QNX разом із IBM, засоби для підтримки інших мов різних розробників. Безліч розширень доповнює Eclipse менеджерами для роботи з базами даних, серверами застосунків та інших.

З версії 3.0 Eclipse став не монолітною IDE, яка підтримує розширення, а набором розширень. У основі лежать фреймворки OSGi, і SWT/JFace, на основі яких розроблений наступний шар – платформа і засоби розробки

[1)] [15] MonoDevelop – Wikipedia. URL: https://en.wikipedia.org/wiki/MonoDevelop (дата звернення 05.02.2019).

повноцінних клієнтських застосунків RCP (Rich Client Platform). Платформа RCP є базою для розробки різних RCP програм як торент-клієнт Azareus чи File Arranger.

Eclipse написана на Java, тому є платформонезалежним продуктом, крім бібліотеки графічного інтерфейсу SWT, яка розробляється окремо для більшості поширених платформ[16][1)].

1.6.3 Розгляд Microsoft Visual Studio

Microsoft Visual Studio є інтегрованим середовищем розробки (IDE) від Microsoft. Він використовується для розробки комп'ютерних програм, а також веб-сайтів, веб-додатків, веб-служб і мобільних додатків.

Visual Studio використовує платформи розробки програмного забезпечення Microsoft, такі як Windows API, Windows Forms, Windows Presentation Foundation, Магазин Windows і Microsoft Silverlight. Він може виробляти як власний код, так і керований код.

Visual Studio включає в себе редактор коду, що підтримує IntelliSense (компонент завершення коду), а також рефакторинг коду. Інтегрований відладчик працює як відладчик вихідного рівня, так і відладчик на рівні машини. Інші вбудовані інструменти включають в себе профайлер коду, дизайнер форм для створення GUI-додатків, веб-дизайнер, дизайнер класів і дизайнер схеми бази даних. Він приймає плагіни, що підвищують функціональність практично на кожному рівні, включаючи додавання підтримки систем керування джерелами (наприклад, Subversion і Git) і додавання нових наборів інструментів, таких як редактори та візуальні дизайнери для доменних мов або наборів програм для інших аспектів розробки Життєвий цикл (наприклад, клієнт Team Foundation Server: Team Explorer).

[1)] [16] Eclipse – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Eclipse(дата звернення 05.02.2019).

Visual Studio підтримує 36 різних мов програмування і дозволяє редактору коду і відладчику підтримувати (в різному ступені) практично будь-яку мову програмування, за умови наявності спеціальної мовної служби. Вбудовані мови включають C, C++, C++/CLI, Visual Basic .NET, C#, F#, JavaScript, TypeScript, XML, XSLT, HTML і CSS. Підтримка інших мов, таких як Python, Ruby, Node.js і M серед інших, доступна через плагіни. Java (і J#) підтримувалися в минулому.

Найбільш базове видання Visual Studio, видання Community, доступне безкоштовно. Гаслом для Visual Studio Community edition є "Безкоштовна, повнофункціональна IDE для студентів, відкритих джерел та окремих розробників".

В даний час підтримується версія Visual Studio – 2019[17][1)].

В результаті розгляду, середовище Visual Studio було обрано для розробки ПП, нажаль продукт крадений, на майбутнє передбачений перехід, ПП, що розробляється на систему MonoDevelop, яка є крос-платформною і дозволяє створювати однакові програми, як для операційної системи (ОС) Windows, так і для Unix.

[1)] [17] Microsoft Visual Studio – Wikipedia. URL: https://en.wikipedia.org/wiki/Microsoft_Visual_Studio (дата звернення 25.02.2019).

2 ПРОЕКТНА ЧАСТИНА

2.1 Постановка завдань

За результатами огляду і аналізу предметної області, ринку подібних проектів і вимог до ПП, було виділено ряд завдань, які потребують вирішення, щоб успішно виконати проект дипломної роботи:

- спроектувати і розробити БД ПП;
- спроектувати і розробити структуру ПП;
- спроектувати і розробити зовнішній вигляд ПП;
- спроектувати і розробити оцінювальну систему готовності атлета до змагання.

2.2 Побудова функціональної моделі

IDEF0 – Function Modeling – методологія функціонального моделювання і графічного описання процесів, призначена для формалізації і опису бізнес-процесів. Особливістю IDEF0 є її акцент на ієрархічне представлення об'єктів, що значно полегшує розуміння предметної області. В IDEF0 розглядаються логічні зв'язки між роботами, а не послідовність їх виконання в часі (WorkFlow) [18][1)] .

Головною функцією системи, що моделюється, є підготовка спортсмена до змагань.

Дані: спортсмен, готовність, норматив, змагання, стадіон, календар, тренування, тренер, лікар, додаток, устаткування, спортсмен, який виступив.

Функції: пройти перехідний період, підготовчий період, перезмагальний період, змагальний період та підготовити спортсмена.

Мета створення інформаційної системи: організувати підготовку спортсмена-легкоатлета до змагань.

[1)] [18] IDEF0 – Вікіпедія. URL: https://uk.wikipedia.org/wiki/IDEF0 (дата звернення 30.03.2019).

Питання:

- що потрібно для того ,щоб підготувати спортсмена?
- які є змагання,в якому місті і для якого року?
- які обстеження потрібні для підготовки спортсмена?
- як визначити готовність спорсмена?
- що потрібно для виконання нормативу?
- як буде працювати додаток?
- скільки лікарів потрібно для підготовки спортсмена?
- від чого залежить результат легкоатлета?
- що входить в перехідний,підготовчий,передзмагальний і змагальний періоди?
- скільки може бути тренерів в одного спорсмена?

Методологія IDEF0 складається з контекстної діаграми та діаграм декомпозиції.Всі діаграми складються з робіт (прямокутники) та стрілок (даних).

В контекстній діаграмі всього одна робота і вона є роботою, що виконує система в цілому – підготовити спортмена.

Стрілки на контекстній діаграмі служать для опису взаємодії системи з навколишнім світом. Вони можуть починатися біля рамки діаграми і закінчуватися біля роботи, або навпаки. Такі стрілки називаються граничними[19][1)].

Існує 5 типів граничних стрілок:

- стрілка вхід(зліва від блоку робота) – показує, що необхідно, щоб виконалась головна функція(спортсмен);
- вихід(справа від блоку робота) – результат нашої функції(спортсмен, який виступив) ;
- управління(зверху від блоку робота) – правила, стандарти і процедури ,якими керується наша робота(готовність, норматив, змагання, стадіон, календар, тренування) ;

[1)] [19] Принципи побудови моделі IDEF0 – Реферати та учбові матеріали на um.co.ua. URL: http://um.co.ua/10/10-17/10-171815.html (дата звернення 30.03.2019).

– механізм(знизу від блока робота) – ресурси, що виконують роботу(тренер, лікар, додаток, устаткування) ;
– виклик – використовується, щоб показати, що деяка робота використовується за межами моделі.

На рисунку 2.1 показана контекстна діаграма нульового рівня.

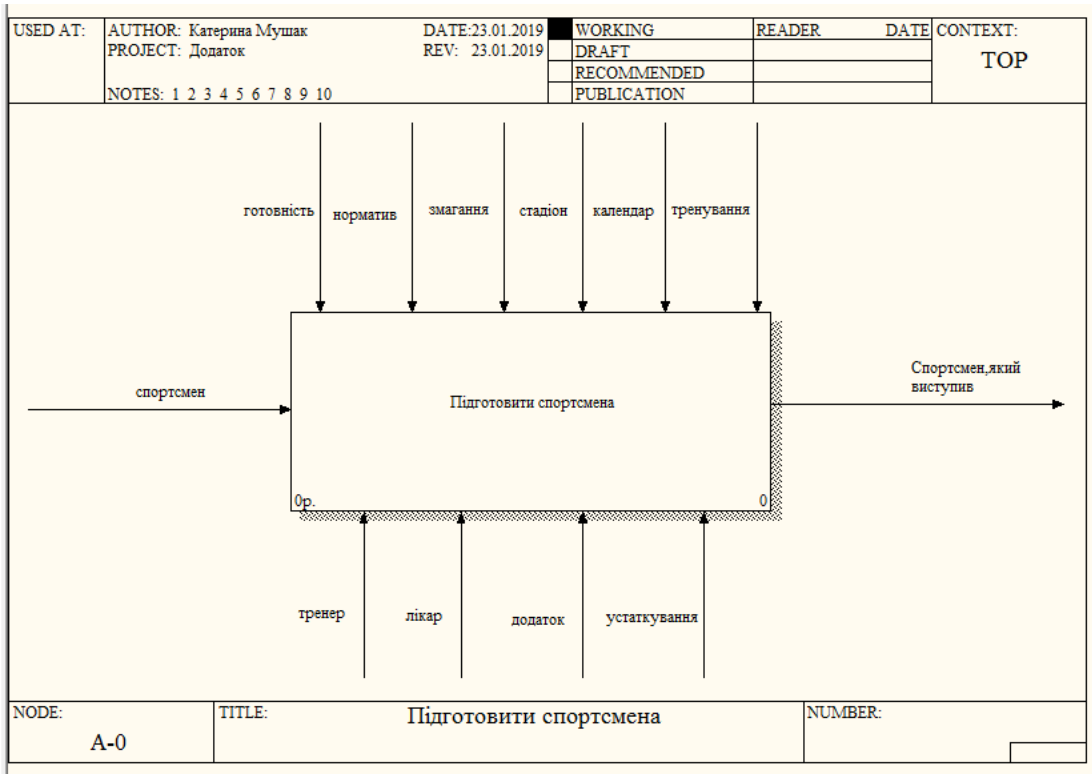

Рисунок 2.1 – Контекстна діаграма IDEF0

Діаграма декомпозиції – це опис і взаємодія фрагментів, розбитих з контекстної діаграми. Тут ми бачимо вже 4 блоки-роботи (4 періоди).Зв’язані вони між собою внутрішніми стрілками (див. рис.2.2).

Існує 5 типів внутрішніх стрілок:

– зв'язок по входу (output-input) – починаючий підготовку спортсмен, підготовлений спортсмен, готовий до змагань спортсмен **;**
– зв'язок з управління (output-control) – в нашому випадку такого зв’язку немає **;**

– зворотній зв'язок по входу (output-input feedback) – непідготовленний спортсмен **;**
– зворотній зв'язок з управління (output-control feedback) – в нашому випадку такого зв’язку немає**;**
– зв'язок вихід-механізм (output-mechanism) – в нашому випадку такого зв’язку немає**;**

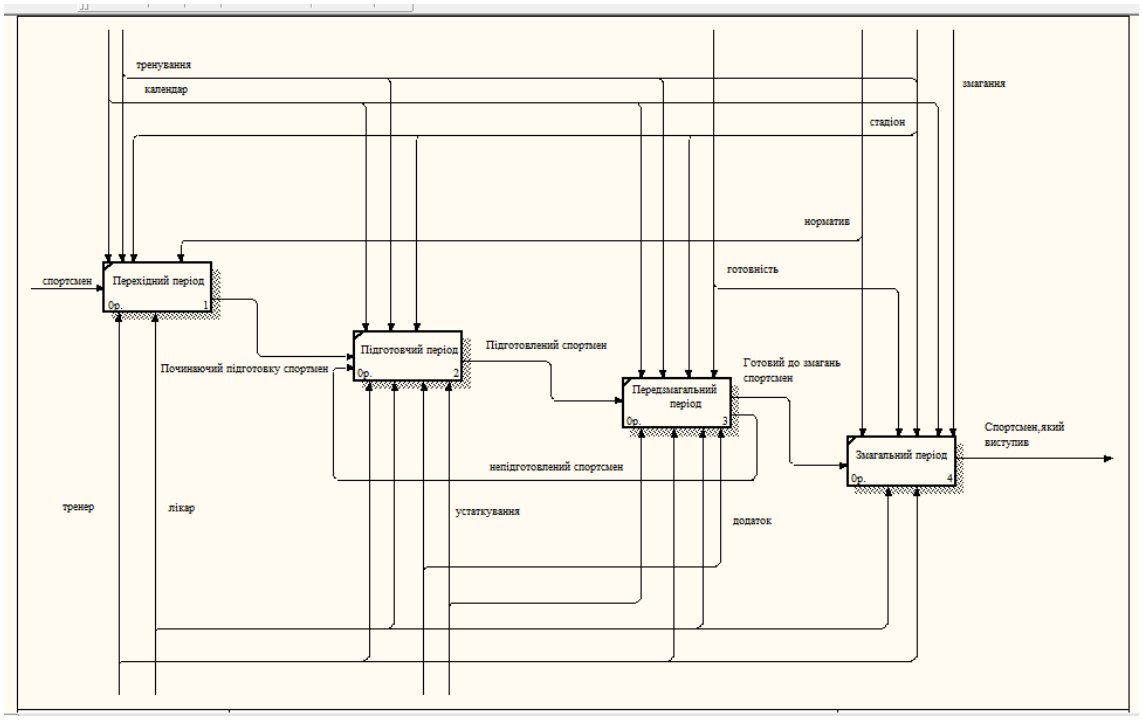

Рисунок 2.2 – Діаграма декомпозиції IDEF0

2.3 Проектування структури БД ПП

Кожна БД складається з певної кількості об’єктів, – сутностей. Сутність – це реальний або представляється тип об'єкта, інформація про який повинна зберігатися і бути доступна. У діаграмах сутність представляється у вигляді прямокутника, що містить ім'я сутності. При цьому ім'я сутності − це ім'я типу, а не деякого конкретного примірника цього типу. Приклади сутностей: ФАКУЛЬТЕТ, ГРУПА, СТУДЕНТ. В БД, що проектується, можна виділити сутності, описані нижче. Зв’язки сутностей показані на рис. 2.3.

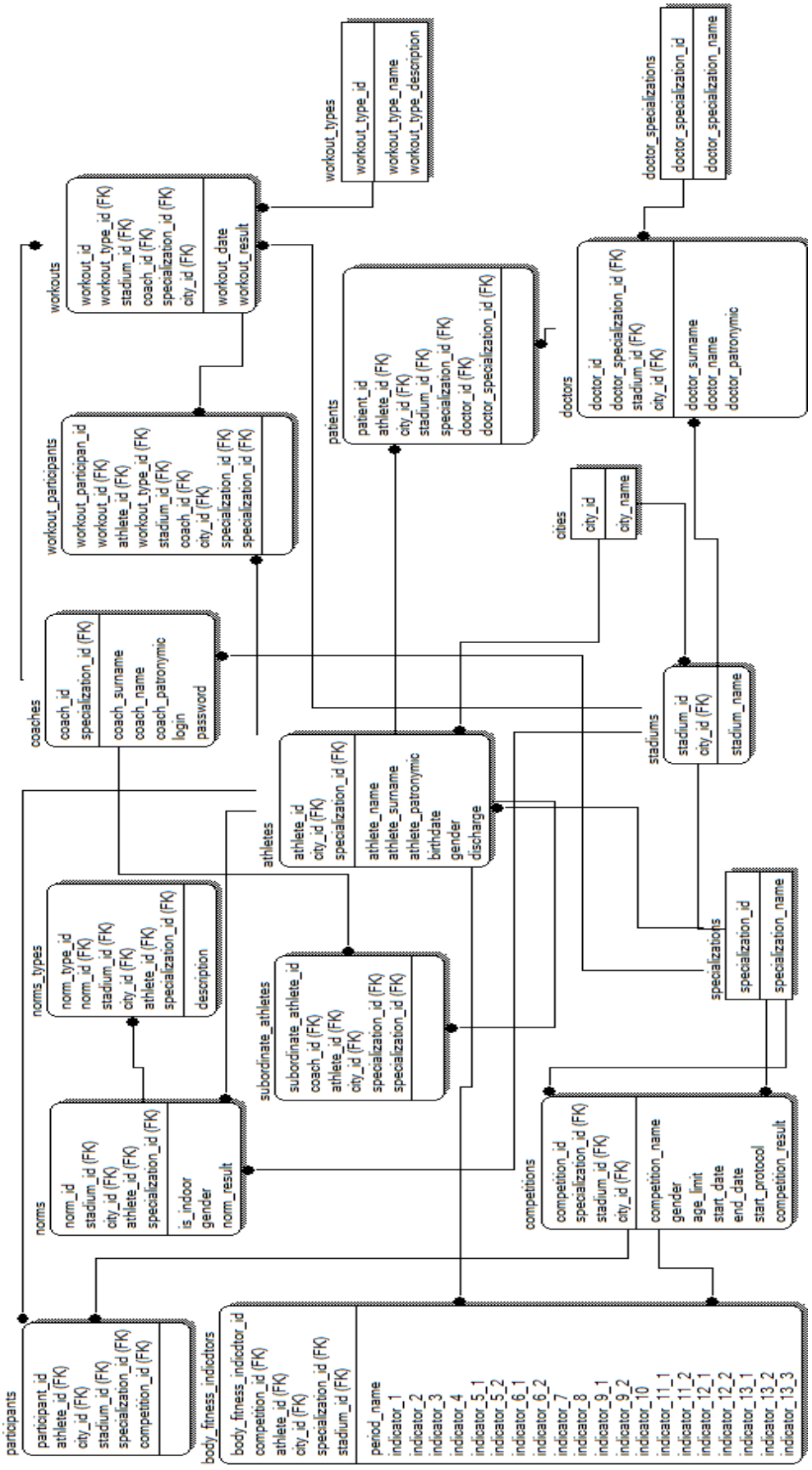

Рисунок 2.3 – Схема БД застосування

Перша сутність – атлети (athletes), має наступні атрибути (властивості): прізвище (athlete_surname), ім’я (athlete_name), ім’я по батькові (athlete_patronymic), дату народження (birthdate), стать (gender), місто (виражено у якості посилання на номер запису у таблиці міст – city_id), спеціалізацію (виражено у якості посилання на номер запису у таблиці спеціалізації − specialization_id), розряд (discharge).

Друга сутність – показники тренованості (body fitness indicators), має наступні атрибути (властивості): назву періоду (period_name), назви 13 показників (indicator_1 – indicator_13_3), спеціалізацію (виражено у якості посилання на номер запису у таблиці спеціалізації − specialization_id), місто (виражено у якості посилання на номер запису у таблиці міст – city_id), атлетів (виражено у якості посилання на номер запису у таблиці атлетів – athletes), стадіону (виражено у якості посилання на номер запису у таблиці стадіонів – stadiums), змагань (виражено у якості посилання на номер запису у таблиці змагань – competitions) .

Третя сутність – міста (cities), має наступні атрибути (властивості): назва міста (city_name) .

Четверта сутність – тренери (coaches), має наступні атрибути (властивості): прізвище (coach_surname), ім’я (coach_name), ім’я по-батькові (coach_patronymic), спеціалізацію (виражено у якості посилання на номер запису у таблиці спеціалізації − specialization_id), логін (login) та пароль (password) .

П’ята сутність – змагання (competitions), має наступні атрибути (властивості): назву змагань (competition_name), стать (gender), вікове обмеження (age_limit), початок змагань (start_date), кінець змагань (end_date), стартовий протокол (start_protocol), результати змагань (competition_result), спеціалізацію (виражено у якості посилання на номер запису у таблиці спеціалізації − specialization_id), стадіону (виражено у якості посилання на номер запису у таблиці стадіонів – stadiums), місто (виражено у якості посилання на номер запису у таблиці міст – city_id) .

Шоста сутність – спеціалізації лікарів (doctor_specializations), має наступні атрибути (властивості): назву спеціалізації лікаря (doctor_specialization_name) .

Сьома сутність – лікарі (doctors), має наступні атрибути (властивості): прізвище (doctor_surname), ім’я (doctor_name), ім’я по- батькові (doctor_patronymic), стадіону (виражено у якості посилання на номер запису у таблиці стадіонів – stadiums), місто (виражено у якості посилання на номер запису у таблиці міст – city_id), спеціалізації лікарів (виражено у якості посилання на номер запису у таблиці спеціалізації лікарів –doctor_specializations) .

Восьма сутність – нормативи (norms), має наступні атрибути (властивості): стать (gender), результати норм (norm_result), на повітрі проведення чи в приміщенні (is_indoor), стадіону (виражено у якості посилання на номер запису у таблиці стадіонів – stadiums), місто (виражено у якості посилання на номер запису у таблиці міст – city_id), атлетів (виражено у якості посилання на номер запису у таблиці атлетів – athletes), спеціалізацію (виражено у якості посилання на номер запису у таблиці спеціалізації − specialization_id) .

Дев’ята сутність – типи нормативів (norms_types), має наступні атрибути (властивості): опис нормативів (description), атлетів (виражено у якості посилання на номер запису у таблиці атлетів – athletes), спеціалізацію (виражено у якості посилання на номер запису у таблиці спеціалізації − specialization_id), стадіону (виражено у якості посилання на номер запису у таблиці стадіонів – stadiums), місто (виражено у якості посилання на номер запису у таблиці міст – city_id), норми (виражено у якості посилання на номер запису у таблиці норм – norm_id) .

Десята сутність – учасники (participants), має наступні атрибути (властивості): атлетів (виражено у якості посилання на номер запису у таблиці атлетів – athletes), спеціалізацію (виражено у якості посилання на номер запису у таблиці спеціалізації − specialization_id), стадіону (виражено у якості посилання на номер запису у таблиці стадіонів – stadiums), місто (виражено у яко-

сті посилання на номер запису у таблиці міст – city_id), змагань (виражено у якості посилання на номер запису у таблиці змагань – competitions) .

Одинадцята сутність – спеціалізації (specializations), має наступні атрибути(властивості): назву спеціалізації (specialization_name).

Дванадцята сутність – стадіони (stadiums), має наступні атрибути (властивості): назву стадіону (stadium_name), місто (виражено у якості посилання на номер запису у таблиці міст – city_id) .

Тринадцята сутність – підпорядкуванні атлети (subordinate athletes), має наступні атрибути (властивості): атлетів (виражено у якості посилання на номер запису у таблиці атлетів – athletes), спеціалізацію (виражено у якості посилання на номер запису у таблиці спеціалізації − specialization_id), місто (виражено у якості посилання на номер запису у таблиці міст – city_id), тренерів (виражено у якості посилання на номер запису у таблиці тренерів – coach_id).

Чотирнадцята сутність – тренування (workouts), має наступні атрибути (властивості): дати тренування (workout_date), результати змагання (workout_result), спеціалізацію (виражено у якості посилання на номер запису у таблиці спеціалізації − specialization_id), місто (виражено у якості посилання на номер запису у таблиці міст – city_id), тренери (виражено у якості посилання на номер запису у таблиці тренерів – coach_id), стадіону (виражено у якості посилання на номер запису у таблиці стадіонів – stadiums), типу тренування (виражено у якості посилання на номер запису у таблиці типів тренування – workouts_types) .

П’ятнадцята сутність – учасники тренування (workouts_participants), має наступні атрибути (властивості): спеціалізацію (виражено у якості посилання на номер запису у таблиці спеціалізації − specialization_id), місто (виражено у якості посилання на номер запису у таблиці міст – city_id), тренерів (виражено у якості посилання на номер запису у таблиці тренерів – coach_id), стадіону (виражено у якості посилання на номер запису у таблиці стадіонів – stadiums), атлетів (виражено у якості посилання на номер запису у таблиці

атлетів – athletes), типу тренування (виражено у якості посилання на номер запису у таблиці типів тренування – workouts_types), тренування (виражено у якості посилання на номер запису у таблиці тренування – workout) .

Шістнадцята сутність – типи тренування (workouts_types), має наступні атрибути (властивості): назви типу тренування (workout_type_name), опису типу тренування (workout_type_description) .

БД ПП буде зберігати лише дані екземплярів сутностей, а уявлення таблиць і процедури будуть реалізовані у функціях програми-настільного додатку.

2.4 Проектування елементів відображання даних БД ПП

У ПП будуть два види елементів відображення: головний і допоміжний.

У головному елементі будуть відображатися таблиці з даними атлетів, тренувань та їх типів, змагань, лікарів, нормативів та їх типів. Також у головному елементі будуть знаходитися відповідні елементи модифікації даних таблиць. Приблизний вигляд головного елементу продемонстровано на рис. 2.4.

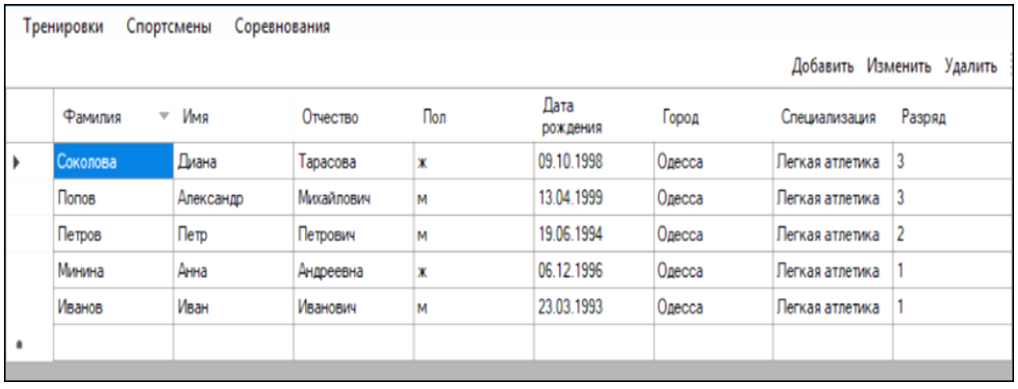

Рисунок 2.4 – Вкладка Спортсмени

При відкритті застосування користувач має змогу ввійти в систему, або зареєструватися (див. рис.2.5-2.6).

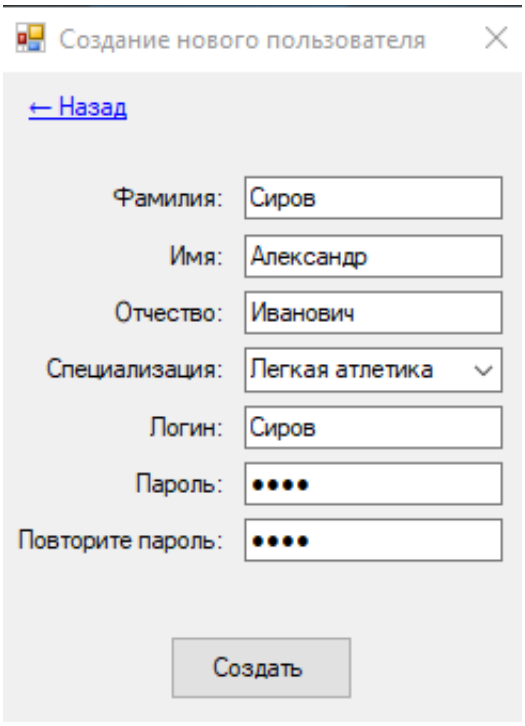

Рисунок 2.5 – Вікно створення нового користувача

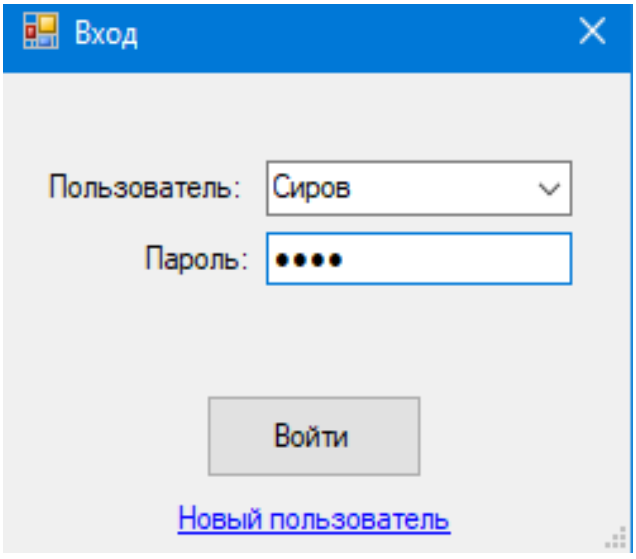

Рисунок 2.6 – Вікно авторизації

Після вдалого проходження авторизації тренер може передивитися своїх підопічних та їх тренування (рис 2.7). Він має змогу додавати, змінювати чи видаляти тренування для своїх спортсменів.

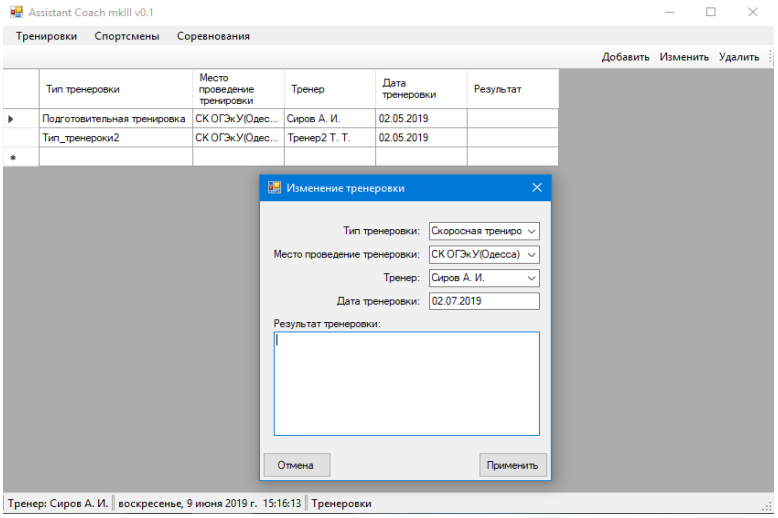

Рисунок 2.7 –Вікно зміни тренування

На вкладці змагання, користувач може подивитися, які є змагання, додати ,змінити чи видалити їх (див. рис.2.8).

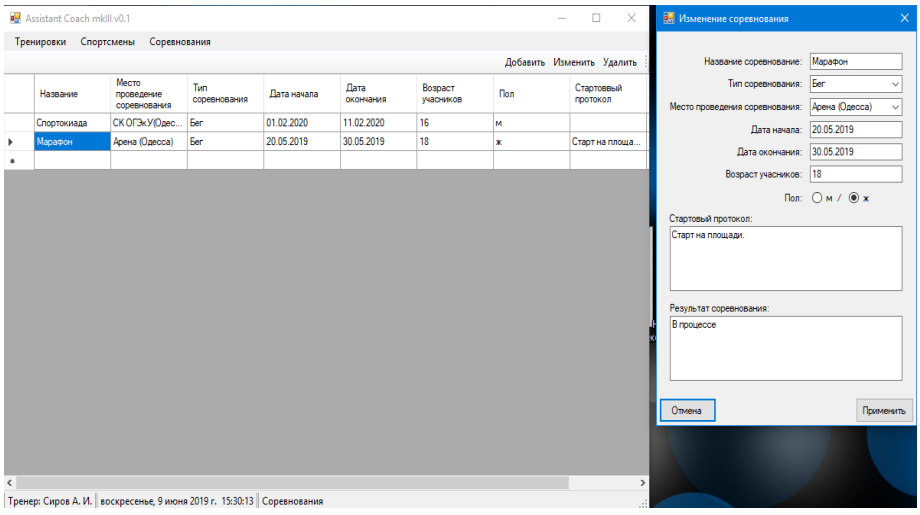

Рисунок 2.8 – Вікно зміни змагань

Допоміжні елементи екрану будуть відображати форми для зміни значень рядку даних з певної таблиці. Для різних таблиць будуть створені різні форми. Приблизний вигляд допоміжних елементів продемонстровано на рис. 2.9–2.10.

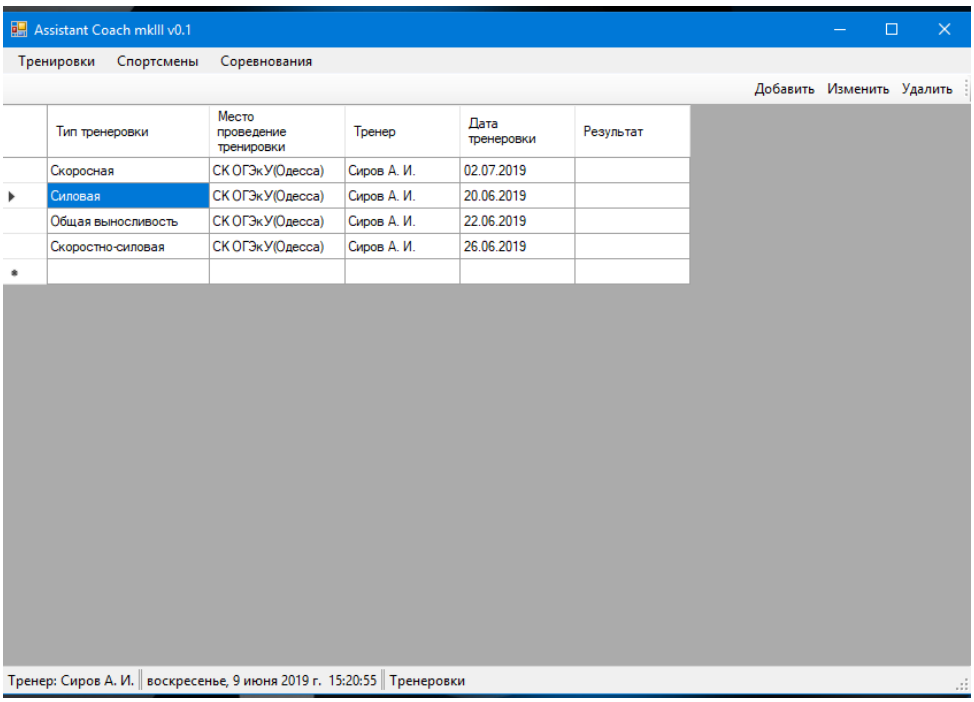

Рисунок 2.9 – Тип тренування

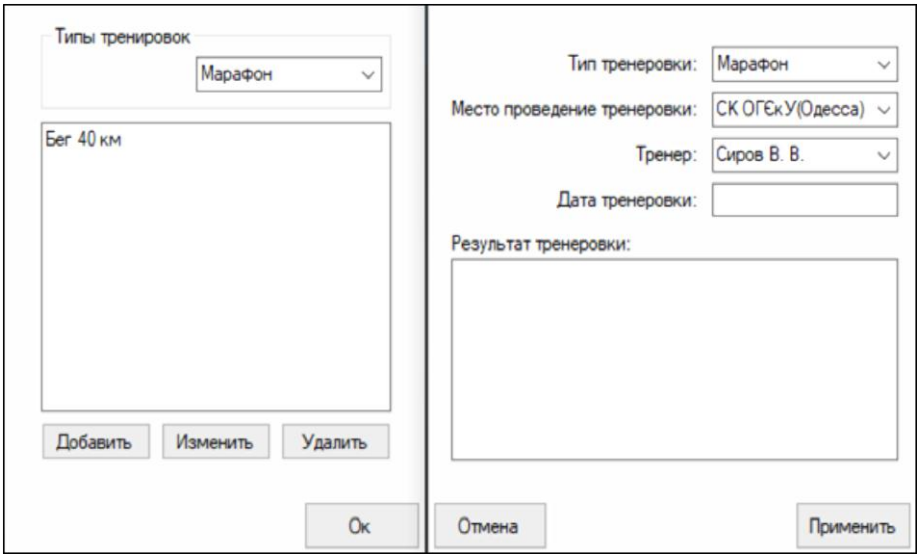

Рисунок 2.10 – Типи тренувань

Окремо стоїть форма перегляду показників готовності атлета до змагання, приблизний вигляд якої зображено на рис. 2.11–2.12 .

Рисунок 2.11 – Заповнення норм, для розрахунку готовності спортсмена до змагань

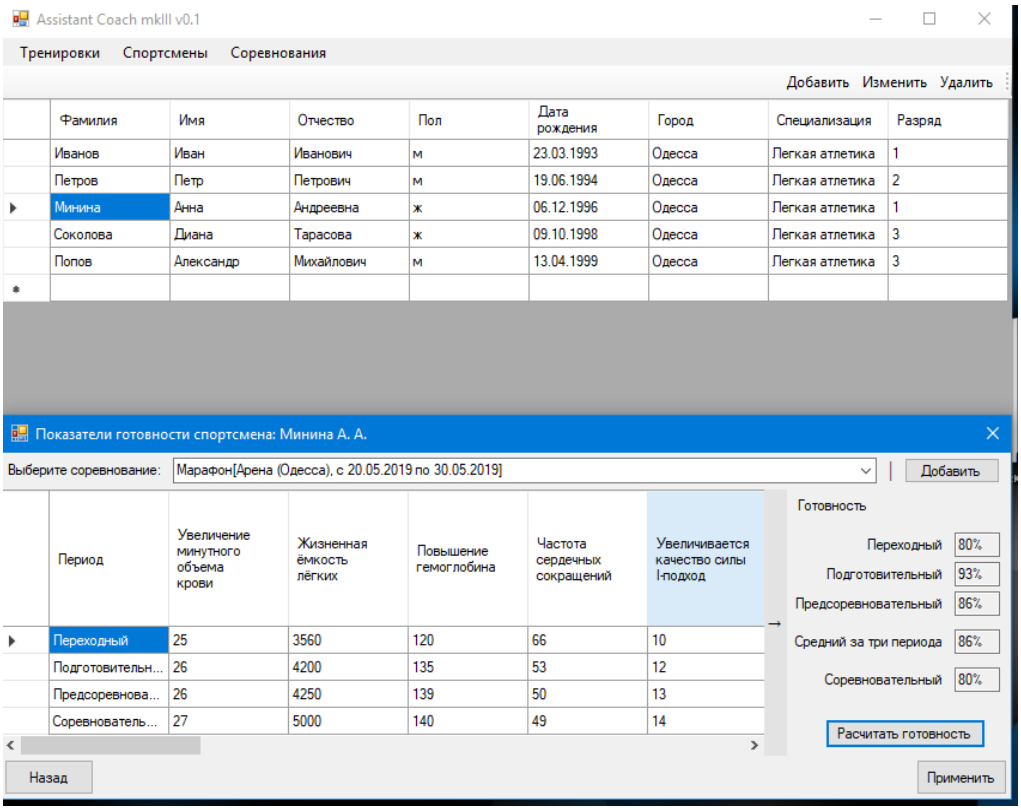

Рисунок 2.12 – Розрахунок готовності

3 ОПИС ОКРЕМИХ ПРОГРАМНИХ ЕЛЕМЕНТІВ РОЗРОБКИ

3.1 Розробка інструментів «спілкування» з БД ПП

Щоб прискорити час розробки ПП, будемо використовувати наступні бібліотеки: SQLite.Core і Dapper. Ці бібліотеки мають всі необхідні методи для вирішення питань, які можуть виникнути у процесі розробки інструментів «спілкування» з БД ПП.

Для отримання тих чи інших даних з БД ПП, треба вирішити ряд питань, серед яких такі:

- отримання уявлення таблиці БД;
- отримання значення з певного осередку таблиці;
- отримання значень з певного стовпця деякої таблиці;
- виконання поцедури.

Методи, що вирішують ці питання зберігатимуться у класі SQLiteDataAccess і повинні будити публічними та статичними, тобто мати модифікатори public і static.

Розглянемо метод, який реалізує перше питання. Дані, що будуть отримані з таблиці БД, відображаються у елементі керування dataGridView, який повинен мати джерело даних. Для отримання цього джерела, створимо метод, що приймає SQL-запит, який міститиме уявлення таблиці.

```
public static DataSet GetView(string sql)
{
    using (IDbConnection cnn = new SQLiteConnec-
tion(LoadConnectionString()))
    {
        SQLiteDataAdapter adapter = new SQLiteDataAdapter(sql,
(SQLiteConnection)cnn);
        DataSet dataSet = new DataSet();
        adapter.Fill(dataSet);
        return dataSet;
    }
}
```

Метод, використовуючи з’єднання з БД, створить набір даних, заповнить його необхідними даними, які були вказані у SQL-запиті. Виклик методу можна переглянути нижче.

```
dataGridView.DataSource = SQLiteDataAccess.GetView(
    "SELECT " +
    "a.athlete_surname AS[Фамилия], " +
    "a.athlete_name AS[Имя], " +
    "a.athlete_patronymic AS[Отчество], " +
    "a.gender AS[Пол], " +
    "a.birthdate AS[Дата рождения], " +
    "c.city_name AS[Город], " +
    "s.specialization_name AS[Специализация], " +
    "a.discharge AS[Разряд], " +
    "a.athlete_id, a.city_id, a.specialization_id " +
    "FROM athletes a JOIN subordinate_athletes sa JOIN cities c
JOIN specializations s " +
    "WHERE a.athlete_id = sa.athlete_id AND a.city_id =
c.city_id AND a.specialization_id = " +
    "s.specialization_id AND sa.coach_id = " +
coach_id).Tables[0];
```

Перейдемо до вирішення другого питання. Для отримання значення з певного осередку таблиці БД, треба знати назву таблиці БД, назву стовпця таблиці до якого належить осередок, назву стовпця-ідентифікатору, по якому буде шукатися необхідний осередок, і його значення. Ці параметри потрібні для створення SQL-запиту, який поверне необхідне значення. Так як SQL-запит уявляє собою строку типу string, то і всі параметри потрібно буде призвести до цього ж типу.

```
public static string GetCellValue(string table, string column,
string id_name, string id_value)
{
    using (IDbConnection cnn = new SQLiteConnec-
tion(LoadConnectionString()))
    {
        string query = $"SELECT {column} FROM {table} WHERE
{id_name} = {id_value}";
        return cnn.Query<string>(query, new DynamicParame-
ters()).AsList()[0];
    }
}
```

Окрім цієї реалізації створимо ще один варіант цього методу, який матиме на вході лише один вхідний параметр, а саме SQL-запит.

```
public static string GetCellValue(string sql)
{
    using (IDbConnection cnn = new SQLiteConnec-
tion(LoadConnectionString()))
    {
```

```
        return cnn.Query<string>(sql, new DynamicParame-
ters()).AsList()[0];
    }
}
```

Обидві реалізації повертають строкове уявлення значення осередку таблиці БД.

Вирішення третього питання має декілька задач. Перша – це отримання безпосередньо списку даних з певного стовпця таблиці, а друга – заповнення даними елементи керування типу ComboBox і ListBox.

Метод, що вирішує першу задачу, буде приймати два параметри: назву таблиці і назву стовпця, і повертати список значень певного типу, що буде вказано при виклику метода.

```
public static List<T> GetList<T>(string table, string column)
{
    using (IDbConnection cnn = new SQLiteConnec-
tion(LoadConnectionString()))
    {
        return cnn.Query<T>($"SELECT {column} FROM {ta-
ble}").AsList();
    }
}
```

Метод, що вирішує другу задачу має лише на вході чотири параметра: SQL-запит, елемент керування, який потрібно заповнити, рядок, що буде відображатися, значення, яке буде мати рядок, що відображається. Після заповнення даними, метод поверне елемент керування, який було передано.

Щоб не реалізовувати два однакових методи, метод буде отримувати і повертати елемент курування типу ListControl від якого наслідуються вже ComboBox і ListBox.

```
public static ListControl FillBox(string sql, ListControl box,
string displayMember, string valueMember)
{
    using (IDbConnection cnn = new SQLiteConnec-
tion(LoadConnectionString()))
    {
        SQLiteDataAdapter adapter = new SQLiteDataAdapter(sql,
(SQLiteConnection)cnn);
        DataTable dt = new DataTable();
        adapter.Fill(dt);
```

```
        box.DataSource = dt;
        box.DisplayMember = displayMember;
        box.ValueMember = valueMember;
        return box;
    }
}
```

Знайдемо рішення четвертого питання. Для виконання певних дій з даними будемо використовувати зручний метод Execute(string sql), що є бібліотеці Dapper. Але щоб кожного разу не встановлювати власноруч з'єднання з БД, створимо метод обгортку з тим же ім'ям і параметрами, який можна переглянути нижче.

```
public static void Execute(string sql)
{
    using (IDbConnection cnn = new SQLiteConnec-
tion(LoadConnectionString()))
    {
        cnn.Execute(sql);
    }
}
```

При виклику цього методу в якості вхідного параметру виступає SQL-запит, у який можна буде помістити команди на додавання, зміни і видалення даних з таблиць БД.

В усіх методах фігурує виклик метода LoadConnectionString(). Цей метод, на відміну від інших, не публічний і зчитує з файлу конфігурацій програми-настільного додатку App.config рядок з'єднання з БД ПП. Отримавши рядок він його повертає у змінній типу string.

3.2 Технічні рішення при розробці елементів відображання

Головний елемент відображення складається з шести елементів керування, це:

- таблиця відображення даних(DataGridView);
- контекстного меню(ContextMenuStrip);
- панель інструментів(ToolStrip);
- кнопки додавання(ToolStripButton);

- кнопки зміни(ToolStripButton);
- кнопки видалення(ToolStripButton).

Так як зовнішній вигляд однаковий для усіх таблиць БД, то треба при створенні елементу вказувати, які дані у таблицю завантажувати і які дії будуть виконуватися при натисканні кнопок.

Задля цієї мети був створений клас TableViewUC, що наслідує клас UserControl. В цьому класі були створені п’ять методів ініціалізації компонентів, що були перераховані вище. Кожен метод робить наступні дії:

- заповнює таблицю(dataGridView) даними з відповідної до метода таблиці БД;
- встановлює для кнопок і контекстного меню методи-обробники подій.

Для зручності цей клас було розширено на шість окремих файлів, це можливо, якщо вказати модифікатор partial. Таким чином маємо великий потенціал для розширення можливостей відображення і взаємодії з даними майбутніх таблиць БД ПП. Розглянемо, що містить головний файл(TableViewUC.cs) і один файл розширення, наприклад для випадку перегляду таблиці атлетів(TableViewUC.AthleteView.cs).

У класі TableViewUC, що у одноіменному файлі, оголошено поле, яке міститиме номер запису у БД таблиці coaches, того тренера, що авторизований. Також у класі міститься оголошення перелічення enum назв тих таблиць, що треба заповнити і два конструктори. Нижче наведено конструктори класу.

```
public TableViewUC()
{
    InitializeComponent();
}

public TableViewUC(string coach_id, TableView tableView) :
this()
{
    this.coach_id = coach_id;

    switch (tableView)
    {
        case TableView.Athlete: InitializeAthleteViewCompo-
nent();
```

```
            break;
        case TableView.Workout: InitializeWorkoutViewCompo-
nent();
            break;
        case TableView.Competition: InitializeCompetitonViewCom-
ponent();
            break;
        case TableView.Norm: InitializeNormViewComponent();
            break;
        case TableView.Doctor: InitializeDoctorViewComponent();
            break;
    }
}
```

Перший є конструктором за замовчуванням і ініціалізує всі компоненти, що є у класі, шляхом виклику методу InitializeComponent(). Другий конструктор, так би мовити, завершує ініціалізацію компонентів, викликаючи методи заповнення таблиць і встановлення методів-обробників подій, в залежності від переданого параметру tableView. Також ініціалізується поле coach_id.

Розглянемо частину класу TableViewUC, що у файлі TableViewUC.AthleteView.cs. В цій частині реалізовано чотири методи:

- InitializeAthleteViewComponent() – заповнює таблицю і встановлює обробники подій;
- LoadAthleteView() – викликає метод GetView з класу SQLiteDataAccess;
- AthleteViewToolButton_Click(object sender, EventArgs e) – обробник подій натиску на кнопки, що розташовані на панелі задач;
- ShowIndicators_Click(object sender, EventArgs e) – обробник подій натиску на елементі контекстного меню.

Подібні методи присутні в інших файлах, що розширюють клас TableViewUC.

ВИСНОВКИ

У результаті роботи, був виконаний аналіз предметної галузі. Були виявлені аналогічні ПП, проаналізовані їх недоліки і переваги.

Крім того виявлені 13 показників, які характеризують ступінь підготовки спортсмена до змагань.

Було проведено моделювання і обрані мова програмування та середовище розробки. З розглянутих середовищ розробки, було обрано Visual Studio і .Net.

Було виконане моделювання БД, логічна та фізична схема БД.

В результаті розробки, одержаний ПП. У подальшому передбачається переведення ПП на Mono Develop, з метою одержання платформно-незалежного ПП.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Розробка системи контролю підготовки спортсменів-легкоатлетів до змагань. Збірник статей за матеріалами студентської наукової конференції Одеського державного екологічного університету (15-18 квітня 2019 р.). Одеса. 2019. С. 158–162.
2. Минутный объём кровообращения URL: http://gemodinamika.ru/minutnyi-objem-krovoobraschenija.html (дата звернення 30.03.2019).
3. Объём легких у спортсменов URL: http://avtobaiki.ru/raznoe/obem-legkih-u-sportsmenov.html (дата звернення 01.04.2019).
4. Норма гемоглобина URL: https://news.sportbox.ru/doppelherz_2013/vbz/spbnews_NI389619_Zagadka-gemoglobina-Norma-visokiy-ili-nizkiy (дата звернення 17.01. 2019).
5. Норма пульса на тренировке. Важнейшие сведения о пульсе человека URL: http://ggym.ru/view_post.php?id=77 (дата звернення 23.03.2019).
6. Тест на прыгучесть – Студопедия URL: https://studopedia.ru/14_17385_test-na-priguchest.html (дата звернення 20.01. 2019).
7. Тест Купера (Бег) : этапы теста, нормативы, таблицы URL: http://kakbegat.com/24-test-kupera-beg.html (дата звернення 01.06. 2019).
8. Runkeeper 9.9 для Android URL: https://runkeeper.ru.uptodown.com/android (дата звернення 20.03.2019).
9. Скачать Fitness Point 2.6.1 для Android URL: https://trashbox.ru/link/fitness-point-android (дата звернення 19.03.2019).
10. Начало работы с .NET Framework | Microsoft Docs URL: https://docs.microsoft.com/ru-ru/dotnet/framework/get-started/ (дата звернення 30.01.2019).

11. Java программная платформа — Википедия URL: https://ru.wikipedia.org/wiki/Java_(программная_платформа) (дата звернення 30.01.2019).
12. MySql – система управления базы данных —« Веб Креатор» URL: https://web-creator.ru/articles/mysql (дата звернення 30.01.2019).
13. SQLite – базы данных URL: https://lecturesdb.readthedocs.io/databases/sqlite.html#id9 (дата звернення 30.01.2019).
14. Microsoft SQL Server – Вікіпедія URL: https://uk.wikipedia.org/wiki/Microsoft_SQL_Server (дата звернення 05.02.2019).
15. MonoDevelop – Wikipedia URL: https://en.wikipedia.org/wiki/MonoDevelop (дата звернення 05.02.2019).
16. Eclipse – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Eclipse(дата звернення 05.02.2019).
17. Microsoft Visual Studio – Wikipedia URL: https://en.wikipedia.org/wiki/Microsoft_Visual_Studio (дата звернення 25.02.2019).
18. IDEF0 – Вікіпедія URL: https://uk.wikipedia.org/wiki/IDEF0 (дата звернення 30.03.2019).
19. Принципи побудови моделі IDEF0 – Реферати та учбові матеріали на um.co.ua URL: http://um.co.ua/10/10-17/10-171815.html (дата звернення 30.03.2019).

ДОДАТОК А

Лістинг класу SQLiteDataAccess

```
using Dapper;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Data.SQLite;
using System.Windows.Forms;

namespace AssistantCoachUI_mkIII.sqlite
{
    public class SQLiteDataAccess
    {
        private static string LoadConnectionString(string name = "Default")
        {
            return ConfigurationManag-
er.ConnectionStrings[name].ConnectionString;
        }

        public static string GetCellValue(string sql)
        {
            using (IDbConnection cnn = new SQLiteConnec-
tion(LoadConnectionString()))
            {
                return cnn.Query<string>(sql, new DynamicParame-
ters()).AsList()[0];
            }
        }
        public static string GetCellValue(string table, string column, string
id_name, string id_value)
        {
            using (IDbConnection cnn = new SQLiteConnec-
tion(LoadConnectionString()))
            {
                string query = $"SELECT {column} FROM {table} WHERE {id_name}
= {id_value}";
                return cnn.Query<string>(query, new DynamicParame-
ters()).AsList()[0];
            }
        }

        public static List<T> GetList<T>(string table, string column)
        {
            using (IDbConnection cnn = new SQLiteConnec-
tion(LoadConnectionString()))
            {
                return cnn.Query<T>($"SELECT {column} FROM {ta-
ble}").AsList();
            }
        }
        public static DataSet GetView(string sql)
        {
            using (IDbConnection cnn = new SQLiteConnec-
tion(LoadConnectionString()))
            {
                SQLiteDataAdapter adapter = new SQLiteDataAdapter(sql,
(SQLiteConnection)cnn);
                DataSet dataSet = new DataSet();
                adapter.Fill(dataSet);
                return dataSet;
            }
        }

        public static ListControl FillBox(string sql, ListControl box, string
displayMember, string valueMember)
        {
```

```
            using (IDbConnection cnn = new SQLiteConnec-
tion(LoadConnectionString()))
            {
                SQLiteDataAdapter adapter = new SQLiteDataAdapter(sql,
(SQLiteConnection)cnn);
                DataTable dt = new DataTable();
                adapter.Fill(dt);
                box.DataSource = dt;
                box.DisplayMember = displayMember;
                box.ValueMember = valueMember;
                return box;
            }
        }

        public static void Execute(string sql)
        {
            using (IDbConnection cnn = new SQLiteConnec-
tion(LoadConnectionString()))
            {
                cnn.Execute(sql);
            }
        }
    }
}
```

ДОДАТОК Б

Лістинг класів форм

```
using AssistantCoachUI_mkIII.sqlite;
using System;
using System.Windows.Forms;

namespace AssistantCoachUI_mkIII.forms
{
    public partial class LoginForm : Form
    {
        public string CoachID;

        public LoginForm()
        {
            InitializeComponent();

            userBox = (ComboBox)SQLiteDataAccess.FillBox("SELECT login,
coach_id FROM coaches ", userBox, "login", "coach_id");
            specializtionBox = (ComboBox)SQLiteDataAccess.FillBox("SELECT *
FROM specializations", specializtionBox, "specialization_name", "specializa-
tion_id");

            passwordBox.KeyPress += PasswordBoxes_KeyPress;
            passwordBox_1.KeyPress += PasswordBoxes_KeyPress;
            passwordBox_2.KeyPress += PasswordBoxes_KeyPress;
            passwordBox.TextChanged += PasswordBoxes_TextChanged;
        }
        private void Clear()
        {
            surnameBox.Clear();
            nameBox.Clear();
            patronymicBox.Clear();
            loginBox.Clear();
            passwordBox_1.Clear();
            passwordBox_2.Clear();
            passwordBox.Clear();
        }

        private void CreateBtn_Click(object sender, EventArgs e)
        {
            if (surnameBox.TextLength > 1 && nameBox.TextLength > 1 && patro-
nymicBox.TextLength > 1 && loginBox.TextLength > 1)
            {
                int password_1 = Convert.ToInt32(passwordBox_1.Text),
                    password_2 = Convert.ToInt32(passwordBox_2.Text);

                if (password_1 == password_2)
                {
                    SQLiteDataAccess.Execute("INSERT INTO coach-
es(coach_surname, coach_name, coach_patronymic, specialization_id, login,
password) " +
                        $"VALUES('{surnameBox.Text}', '{nameBox.Text}',
'{patronymicBox.Text}', {specializtionBox.SelectedValue}, '{loginBox.Text}',
{passwordBox_1.Text})");
                    userBox = (ComboBox)SQLiteDataAccess.FillBox("SELECT log-
in, coach_id FROM coaches ", userBox, "login", "coach_id");
                    createUserPanel.Visible = false;
                    Text = "Вход";
                    MaximumSize = new System.Drawing.Size(Width, 220);
                    Height = MaximumSize.Height;
                    Clear();
                }
                else
                {
                    messageLabel2.Visible = true;
                }
            }
```

```
        }

        private void GoBtn_Click(object sender, EventArgs e)
        {
            int password = Con-
vert.ToInt32(SQLiteDataAccess.GetCellValue($"SELECT password FROM coaches
WHERE coach_id = {userBox.SelectedValue}"));

            try
            {
                if (password == Convert.ToInt32(passwordBox.Text))
                {
                    CoachID = userBox.SelectedValue.ToString();
                    DialogResult = DialogResult.OK;
                }
                else
                {
                    messageLabel.Visible = true;
                    passwordBox.Clear();
                }
            }
            catch (FormatException) { messageLabel.Visible = true; }
        }

        private void NewUserLink_LinkClicked(object sender, LinkLabel-
LinkClickedEventArgs e)
        {
            createUserPanel.Visible = true;
            Text = "Создание нового пользователя";
            MaximumSize = new System.Drawing.Size(Width, 340);
            Height = MaximumSize.Height;
        }

        private void PasswordBoxes_TextChanged(object sender, EventArgs e)
        {
            TextBox textBox = (TextBox)sender;

            if (textBox.TextLength > 0)
            {
                messageLabel.Visible = false;
                messageLabel2.Visible = false;
            }
        }

        private void PasswordBoxes_KeyPress(object sender, KeyPressEventArgs
e)
        {
            if ((e.KeyChar >= '0') && (e.KeyChar <= '9'))
            {
                return;
            }
            if (char.IsControl(e.KeyChar))
            {
                if (e.KeyChar == (char)Keys.Back)
                {
                    return;
                }
            }
            e.Handled = true;
        }

        private void BackLink_LinkClicked(object sender, LinkLabel-
LinkClickedEventArgs e)
        {
            createUserPanel.Visible = false;
            Text = "Вход";
            MaximumSize = new System.Drawing.Size(Width, 220);
            Height = MaximumSize.Height;
            Clear();
        }
    }
}
```

```
using AssistantCoachUI_mkIII.sqlite;
using AssistantCoachUI_mkIII.uc;
using System;
using System.Windows.Forms;

namespace AssistantCoachUI_mkIII.forms
{
    public partial class MainForm : Form
    {
        public Control Content
        {
            get
            {
                return contentPanel.Controls[0];
            }
            set
            {
                if (Content == value) return;
                value.Dock = DockStyle.Fill;
                contentPanel.Controls.RemoveAt(0);
                contentPanel.Controls.Add(value);
            }
        }
        public string CoachID { get; set; }

        public MainForm()
        {
            LoginForm loginForm = new LoginForm();
            loginForm.ShowDialog();

            if (loginForm.DialogResult == DialogResult.Cancel)
            {
                Application.Exit();
                Console.WriteLine("loginForm.DialogResult = " +
loginForm.DialogResult);
                Close();
            }

            if (loginForm.DialogResult == DialogResult.OK)
            {
                InitializeComponent();
                contentPanel.Controls.Add(new Control());

                CoachID = loginForm.CoachID;

                currentCoahLabel.Text = "Тренер: " + SQLiteDataAc-
cess.GetCellValue(
                    "SELECT(coach_surname || ' ' || substr(coach_name, 1, 1)
|| '. ' || substr(coach_patronymic, 1, 1) || '.') " +
                    $"FROM coaches WHERE coach_id = {CoachID}");

                timer.Start();
            }
        }

        private void MenuItems_Click(object sender, EventArgs e)
        {
            ToolStripMenuItem item = (ToolStripMenuItem)sender;

            switch (item.Name)
            {
                case "showAthletes":
                    Content = new TableViewUC(CoachID,
TableViewUC.TableView.Athlete);
                    statusLabel.Text = "Спортсмены";
                    break;
                case "showWorkouts":
                    Content = new TableViewUC(CoachID,
TableViewUC.TableView.Workout);
                    statusLabel.Text = "Тренеровки";
```

```
                    break;
                case "showCompetitions":
                    Content = new TableViewUC(CoachID,
TableViewUC.TableView.Competition);
                    statusLabel.Text = "Соревнования";
                    break;
                case "showNorms":
                    Content = new TableViewUC(CoachID,
TableViewUC.TableView.Norm);
                    statusLabel.Text = "Тренеровки → Нормативы";
                    break;
                case "showDoctors":
                    Content = new TableViewUC(CoachID,
TableViewUC.TableView.Doctor);
                    statusLabel.Text = "Спортсмены → Лечащие врачи";
                    break;
                case "showWorkoutsTypes":
                    new ListViewForm(CoachID,
(TableViewUC)contentPanel.Controls[0], ListView-
Form.TableListVew.WorkoutsTypes).Show();
                    break;
                case "showNormsTypes":
                    new ListViewForm(CoachID,
(TableViewUC)contentPanel.Controls[0], ListView-
Form.TableListVew.NormsTypes).Show();
                    break;
            }
        }

        private void Timer_Tick(object sender, EventArgs e)
        {
            timeLabel.Text = $"{DateTime.Now.ToLongDateString()}
{DateTime.Now.ToLongTimeString()}";
        }
    }
}


using System;
using System.Windows.Forms;

namespace AssistantCoachUI_mkIII.forms
{
    public partial class DialogForm : Form
    {
        public WorkMode Mode { get; set; }
        protected string coach_id;
        protected string idToUpdate;

        public enum WorkMode
        {
            Insert, Update
        }

        public DialogForm()
        {
            InitializeComponent();
        }

        public DialogForm(string coach_id) : this()
        {
            this.coach_id = coach_id;
        }

        public DialogForm(string coach_id, string idToUpdate) :
this(coach_id)
        {
            this.idToUpdate = idToUpdate;
        }

        private void Cancel_Click(object sender, EventArgs e)
        {
```

```
            Close();
        }


    }
}

using System.Windows.Forms;

namespace AssistantCoachUI_mkIII.uc
{
    public partial class TableViewUC : UserControl
    {
        public int SelectedRowIndex { get; set; }

        private readonly string coach_id;

        public enum TableView
        {
           Athlete, Workout, Competition, Norm, Doctor, WorkoutType, NormType
        }

        public TableViewUC()
        {
            InitializeComponent();
        }

        public TableViewUC(string coach_id, TableView tableView) : this()
        {
            this.coach_id = coach_id;

            switch (tableView)
            {
                case TableView.Athlete: InitializeAthleteViewComponent();
                    break;
                case TableView.Workout: InitializeWorkoutViewComponent();
                    break;
                case TableView.Competition: InitializeCompetitonViewCompo-
nent();
                    break;
                case TableView.Norm: InitializeNormViewComponent();
                    break;
                case TableView.Doctor: InitializeDoctorViewComponent();
                    break;
            }

        }

        private void DataGridView_CellEnter(object sender, DataGridView-
CellEventArgs e)
        {
            SelectedRowIndex = e.RowIndex;
        }

        public void RefreshDataGridView(TableView tableView)
        {
            switch (tableView)
            {
                case TableView.Workout: LoadWorkoutView(); break;
                case TableView.Norm: LoadNormView(); break;
            }
        }
    }
}
using AssistantCoachUI_mkIII.forms;
using AssistantCoachUI_mkIII.sqlite;
using System;
using System.Windows.Forms;

namespace AssistantCoachUI_mkIII.uc
{
    public partial class TableViewUC
```

```
    {
        private void InitializeAthleteViewComponent()
        {
            LoadAthleteView();

            addToolBtn.Click += AthleteViewToolButton_Click;
            editToolBtn.Click += AthleteViewToolButton_Click;
            deleteToolBtn.Click += AthleteViewToolButton_Click;

            contextMenuStrip.Items.Add("Посмотреть готовность");
            contextMenuStrip.Items[0].Click += ShowIndicators_Click;
        }

        private void LoadAthleteView()
        {
            dataGridView.DataSource = SQLiteDataAccess.GetView(
               "SELECT " +
                   "a.athlete_surname AS[Фамилия], " +
                   "a.athlete_name AS[Имя], " +
                   "a.athlete_patronymic AS[Отчество], " +
                   "a.gender AS[Пол], " +
                   "a.birthdate AS[Дата рождения], " +
                   "c.city_name AS[Город], " +
                   "s.specialization_name AS[Специализация], " +
                   "a.discharge AS[Разряд], " +
                   "a.athlete_id, a.city_id, a.specialization_id " +
               "FROM athletes a JOIN subordinate_athletes sa JOIN cities c
JOIN specializations s " +
               "WHERE a.athlete_id = sa.athlete_id AND a.city_id = c.city_id
AND a.specialization_id = " +
                   "s.specialization_id AND sa.coach_id = " +
coach_id).Tables[0];
            dataGridView.Columns[8].Visible = false;
            dataGridView.Columns[9].Visible = false;
            dataGridView.Columns[10].Visible = false;
        }

        private void AthleteViewToolButton_Click(object sender, EventArgs e)
        {
            ToolStripButton toolBtn = (ToolStripButton)sender;

            switch (toolBtn.Name)
            {
                case "addToolBtn":
                    new AthleteDialog(coach_id) { Text = "Добавление нового
спортсмена", Mode = DialogForm.WorkMode.Insert }.ShowDialog();
                    break;
                case "editToolBtn":
                    new AthleteDialog(coach_id, data-
GridView.Rows[SelectedRowIndex].Cells[8].Value.ToString()) {
                        Text = "Изменение данных спортсмена", Mode = Dialog-
Form.WorkMode.Update }.ShowDialog();
                    break;
                case "deleteToolBtn":
                    SQLiteDataAccess.Execute(
                        $"DELETE FROM body_fitness_indicators WHERE ath-
lete_id = {dataGridView.Rows[SelectedRowIndex].Cells[8].Value}; " +
                        $"DELETE FROM participants WHERE athlete_id = {data-
GridView.Rows[SelectedRowIndex].Cells[8].Value}; " +
                        $"DELETE FROM patients WHERE athlete_id = {data-
GridView.Rows[SelectedRowIndex].Cells[8].Value}; " +
                        $"DELETE FROM workouts_participants WHERE athlete_id
= {dataGridView.Rows[SelectedRowIndex].Cells[8].Value}; " +
                        $"DELETE FROM subordinate_athletes WHERE athlete_id =
{dataGridView.Rows[SelectedRowIndex].Cells[8].Value}; " +
                        $"DELETE FROM norms WHERE athlete_id = {data-
GridView.Rows[SelectedRowIndex].Cells[8].Value}; " +
                        $"DELETE FROM athletes WHERE athlete_id = {data-
GridView.Rows[SelectedRowIndex].Cells[8].Value}");
                    break;
            }
```

```
            LoadAthleteView();
        }

        private void ShowIndicators_Click(object sender, EventArgs e)
        {
            new IndicatorsViewDia-
log(dataGridView.Rows[SelectedRowIndex].Cells[8].Value.ToString()).ShowDialog
();
        }
    }
}


using AssistantCoachUI_mkIII.forms;
using AssistantCoachUI_mkIII.sqlite;
using System;
using System.Windows.Forms;

namespace AssistantCoachUI_mkIII.uc
{
    public partial class TableViewUC
    {
        private void InitializeCompetitonViewComponent()
        {
            LoadCompetitionView();

            contextMenuStrip.Items.Add("Учасники соревнования");
            contextMenuStrip.Items[0].Click += ShowParticipants;

            addToolBtn.Click += CompetitionViewToolButton_Click;
            editToolBtn.Click += CompetitionViewToolButton_Click;
            deleteToolBtn.Click += CompetitionViewToolButton_Click;
        }

        private void LoadCompetitionView()
        {
            dataGridView.DataSource = SQLiteDataAccess.GetView(
               "SELECT " +
                   "c.competition_name AS[Название], " +
                   "(st.stadium_name || '(' || ct.city_name || ')') AS[Место
проведение соревнования], " +
                   "s.specialization_name AS[Тип соревнования], " +
                   "c.start_date AS[Дата начала], " +
                   "c.end_date AS[Дата окончания], " +
                   "c.age_limit AS[Возраст учасников], " +
                   "c.gender AS[Пол], " +
                   "c.start_protocol AS[Стартоввый протокол], " +
                   "c.competition_result AS[Результат соревнования], " +
                   "c.competition_id, c.specialization_id, c.stadium_id " +
               "FROM competitions c JOIN stadiums st JOIN cities ct JOIN spe-
cializations s " +
               "WHERE c.stadium_id = st.stadium_id AND c.specialization_id =
s.specialization_id AND st.city_id = ct.city_id").Tables[0];
            dataGridView.Columns[9].Visible = false;
            dataGridView.Columns[10].Visible = false;
            dataGridView.Columns[11].Visible = false;
        }

        private void CompetitionViewToolButton_Click(object sender, EventArgs
e)
        {
            ToolStripButton toolBtn = (ToolStripButton)sender;

            switch (toolBtn.Name)
            {
                case "addToolBtn":
                    new CompetitionDialog() { Text = "Добавить новое
соревнование", Mode = DialogForm.WorkMode.Insert }.ShowDialog();
                    break;
                case "editToolBtn":
                    new CompetitionDi-
alog(dataGridView.Rows[SelectedRowIndex].Cells[9].Value.ToString()) {
```

```
                        Text = "Изменение соревнования", Mode = Dialog-
Form.WorkMode.Update }.ShowDialog();
                    break;
                case "deleteToolBtn":
                    SQLiteDataAccess.Execute(
                        $"DELETE FROM participants WHERE competition_id =
{dataGridView.Rows[SelectedRowIndex].Cells[9].Value}; " +
                        $"DELETE FROM competitions WHERE competition_id =
{dataGridView.Rows[SelectedRowIndex].Cells[9].Value}");
                    break;
            }

            LoadCompetitionView();
        }

        private void ShowParticipants(object sender, EventArgs e)
        {
            new ListViewForm(coach_id, data-
GridView.Rows[SelectedRowIndex].Cells[9].Value.ToString(), ListView-
Form.TableListVew.Participants).ShowDialog();
        }
    }
}


using AssistantCoachUI_mkIII.forms;
using AssistantCoachUI_mkIII.sqlite;
using System;
using System.Windows.Forms;

namespace AssistantCoachUI_mkIII.uc
{
    public partial class TableViewUC
    {
        private void InitializeDoctorViewComponent()
        {
            LoadDoctorView();

            addToolBtn.Click += DoctorViewToolButton_Click;
            editToolBtn.Click += DoctorViewToolButton_Click;
            deleteToolBtn.Click += DoctorViewToolButton_Click;

            contextMenuStrip.Items.Add("Посмотреть пациентов");
            contextMenuStrip.Items[0].Click += ShowPatients_Click;
        }

        private void LoadDoctorView()
        {
            dataGridView.DataSource = SQLiteDataAccess.GetView(
               "SELECT " +
                    "d.doctor_surname AS[Фамилия], " +
                    "d.doctor_name AS[Имя], " +
                    "d.doctor_patronymic AS[Отчество], " +
                    "ds.doctor_specialization_name AS[Специализация], " +
                    "(s.stadium_name || '(' || c.city_name || ')') AS[Место
работы], " +
                    "d.doctor_id, d.doctor_specialization_id, d.stadium_id "
+
                "FROM doctors d JOIN doctor_specializations ds JOIN stadiums s
JOIN cities c " +
                "WHERE ds.doctor_specialization_id =
d.doctor_specialization_id AND c.city_id = s.city_id AND s.stadium_id =
d.stadium_id").Tables[0];
            dataGridView.Columns[5].Visible = false;
            dataGridView.Columns[6].Visible = false;
            dataGridView.Columns[7].Visible = false;
        }
        private void DoctorViewToolButton_Click(object sender, EventArgs e)
        {
            ToolStripButton toolBtn = (ToolStripButton)sender;

            switch (toolBtn.Name)
```

```
            {
                case "addToolBtn":
                    new DoctorDialog() { Text = "Добавление нового варча",
Mode = DialogForm.WorkMode.Insert }.ShowDialog();
                    break;
                case "editToolBtn":
                    new Doctor-
Dialog(dataGridView.Rows[SelectedRowIndex].Cells[5].Value.ToString()) { Text
= "Изменение врача", Mode = DialogForm.WorkMode.Update }.ShowDialog();
                    break;
                case "deleteToolBtn":
                    SQLiteDataAccess.Execute(
                        $"DELETE FROM patients WHERE doctor_id = {data-
GridView.Rows[SelectedRowIndex].Cells[5].Value}; " +
                        $"DELETE FROM doctors WHERE doctor_id = {data-
GridView.Rows[SelectedRowIndex].Cells[5].Value}");
                    break;
            }

            LoadDoctorView();
        }

        private void ShowPatients_Click(object sender, EventArgs e)
        {
            new ListViewForm(coach_id, data-
GridView.Rows[SelectedRowIndex].Cells[5].Value.ToString(), ListView-
Form.TableListVew.Patients).ShowDialog();
        }

    }
}


using AssistantCoachUI_mkIII.forms;
using AssistantCoachUI_mkIII.sqlite;
using System;
using System.Windows.Forms;

namespace AssistantCoachUI_mkIII.uc
{
    public partial class TableViewUC
    {
        private void InitializeNormViewComponent()
        {
            LoadNormView();

            addToolBtn.Click += NormViewToolButton_Click;
            editToolBtn.Click += NormViewToolButton_Click;
            deleteToolBtn.Click += NormViewToolButton_Click;
        }

        private void LoadNormView()
        {
            dataGridView.DataSource = SQLiteDataAccess.GetView(
                "SELECT " +
                    "nt.norm_type_name AS[Норматив], " +
                    "(a.athlete_surname || ' ' || substr(a.athlete_name, 1,
1) || '. ' || substr(a.athlete_patronymic, 1, 1) || '.') AS[Спортсмен], " +
                    "n.gender AS[Пол], " +
                    "s.stadium_name AS[Место здачи норматива], " +
                    "n.is_indoor AS[В помещении], " +
                    "n.norm_result AS[Результат], " +
                    "n.norm_id, n.athlete_id, n.stadium_id, n.norm_type_id "
+
                "FROM norms n JOIN subordinate_athletes sa JOIN norms_types
nt JOIN athletes a JOIN stadiums s " +
                "WHERE nt.norm_type_id = n.norm_type_id AND a.athlete_id =
n.athlete_id AND s.stadium_id = n.stadium_id AND n.athlete_id = sa.athlete_id
AND " +
                $"sa.coach_id = {coach_id}").Tables[0];
            dataGridView.Columns[6].Visible = false;
            dataGridView.Columns[7].Visible = false;
```

```
            dataGridView.Columns[8].Visible = false;
            dataGridView.Columns[9].Visible = false;
        }

        private void NormViewToolButton_Click(object sender, EventArgs e)
        {
            ToolStripButton toolBtn = (ToolStripButton)sender;

            switch (toolBtn.Name)
            {
                case "addToolBtn":
                    new NormDialog(coach_id) { Text = "Добавление нового
норматива", Mode = DialogForm.WorkMode.Insert }.ShowDialog();
                    break;
                case "editToolBtn":
                    new NormDialog(coach_id, data-
GridView.Rows[SelectedRowIndex].Cells[6].Value.ToString()) { Text =
"Изменение норматива", Mode = DialogForm.WorkMode.Update }.ShowDialog();
                    break;
                case "deleteToolBtn":
                    SQLiteDataAccess.Execute($"DELETE FROM norms WHERE
norm_id = {dataGridView.Rows[SelectedRowIndex].Cells[6].Value}");
                    break;
            }

            LoadNormView();
        }
    }
}
 using AssistantCoachUI_mkIII.forms;
using AssistantCoachUI_mkIII.sqlite;
using System;
using System.Windows.Forms;

namespace AssistantCoachUI_mkIII.uc
{
    public partial class TableViewUC
    {
        private void InitializeWorkoutViewComponent()
        {
            LoadWorkoutView();

            contextMenuStrip.Items.Add("Учасники тренировки");
            contextMenuStrip.Items[0].Click += ShowWorkoutParticipants;

            addToolBtn.Click += WorkoutViewToolButton_Click;
            editToolBtn.Click += WorkoutViewToolButton_Click;
            deleteToolBtn.Click += WorkoutViewToolButton_Click;
        }

        private void LoadWorkoutView()
        {
            dataGridView.DataSource = SQLiteDataAccess.GetView(
                "SELECT " +
                    "wt.workout_type_name AS[Тип тренеровки], " +
                    "(s.stadium_name || \"(\" || ct.city_name || \")\")
AS[Место проведение тренировки], " +
                    "(coach_surname || \" \" || substr(coach_name, 1, 1) ||
\". \" || substr(coach_patronymic, 1, 1) || \".\") AS[Тренер], " +
                    "w.workout_date AS[Дата тренеровки], " +
                    "w.workout_result AS[Результат], " +
                    "w.workout_id, w.stadium_id, w.coach_id " +
                "FROM workouts w JOIN workouts_types wt JOIN stadiums s JOIN
coaches c JOIN cities ct " +
                "WHERE wt.workout_type_id = w.workout_type_id AND
s.stadium_id = w.stadium_id AND c.coach_id = w.coach_id AND ct.city_id =
s.city_id").Tables[0];
            dataGridView.Columns[5].Visible = false;
            dataGridView.Columns[6].Visible = false;
            dataGridView.Columns[7].Visible = false;
        }
```

```
        private void WorkoutViewToolButton_Click(object sender, EventArgs e)
        {
            ToolStripButton toolBtn = (ToolStripButton)sender;

            switch (toolBtn.Name)
            {
                case "addToolBtn":
                    new WorkoutDialog() { Text = "Добавление новой
тренровки", Mode = DialogForm.WorkMode.Insert }.ShowDialog();
                    break;
                case "editToolBtn":
                    new WorkoutDia-
log(dataGridView.Rows[SelectedRowIndex].Cells[5].Value.ToString()) {
                        Text = "Изменение тренеровки", Mode = Dialog-
Form.WorkMode.Update }.ShowDialog();
                    break;
                case "deleteToolBtn":
                    SQLiteDataAccess.Execute(
                        $"DELETE FROM workouts_participants WHERE workout_id
= {dataGridView.Rows[SelectedRowIndex].Cells[5].Value};" +
                        $"DELETE FROM workouts WHERE workout_id = {data-
GridView.Rows[SelectedRowIndex].Cells[5].Value}");
                    break;
            }

            LoadWorkoutView();
        }

        private void ShowWorkoutParticipants(object sender, EventArgs e)
        {
            new ListViewForm(coach_id, data-
GridView.Rows[SelectedRowIndex].Cells[5].Value.ToString(), ListView-
Form.TableListVew.WorkoutParticipants).ShowDialog();
        }
    }
}


using System;
using System.Linq;
using System.Runtime.InteropServices;

namespace AssistantCoachUI_mkIII.support_classes
{
    public class Calculator
    {
        // функция должна рассситать процент готовности и вернуть его
        // gender = true, для пацанов, иначе для девченок
        private static int CalculateReadiness(int[] indicators, bool gender)
        {
            int[] indicatorsReadiness = new int[15]; // сюда будем записывать
порцент готовности каждого показателя

            for (int i = 0; i < indicatorsReadiness.Length; i++)
                indicatorsReadiness[i] = -1;

            // 0 строка - мужской пол, 1 строка - женский пол, столбцы - по-
казатели
            // последовательность паказателей: 1, 2, 3, 4, 7, 10
            Interval[,] standarts = new Interval[2, 6]
            {
                { new Interval(25, 30), new Interval(4500, 8000), new Inter-
val(117, 173), new Interval(48, 74), new Interval(35, 45), new Interval(2400,
3000) },
                { new Interval(25, 30), new Interval(3500, 5300), new Inter-
val(117, 153), new Interval(48, 74), new Interval(24, 32), new Interval(2100,
2700) }
            };

            // считаем интервальные показатели 1, 2, 3, 4, 7, 10
            int[] pointers = { 0, 1, 2, 3, 6, 9 };
            int[] pointers_2 = { 0, 1, 2, 3, 8, 12 };
```

```
            for (int i = 0; i < pointers.Length; i++)
            {
                if ((gender ? standarts[0, i].start : standarts[1, i].start)
< indicators[pointers_2[i]] &&
                    indicators[pointers_2[i]] < (gender ? standarts[0, i].end
: standarts[1, i].end))
                {
                    indicatorsReadiness[pointers[i]] = 100;
                }
                else
                {
                    indicatorsReadiness[pointers[i]] = 0;
                }
            }

            // считаем показатели 5, 6, 11, 12, 13.1, 13.2, 13.3
            pointers = new int[] { 4, 5, 6, 7, 13, 14, 15, 16, 17, 18, 19,
20, 21, 22 };
            pointers_2 = new int[] { 4, 5, 10, 11, 12, 13, 14 };

            for (int i = 0, j = 0; i < pointers.Length; i += 2)
            {
                if (indicators[pointers[i]] < indicators[pointers[i + 1]])
                {
                    indicatorsReadiness[pointers_2[j]] = 100;
                    j++;
                }
                else
                {
                    indicatorsReadiness[pointers_2[j]] = 0;
                    j++;
                }
            }

            // считаем 9 показатель
            if (indicators[10] > indicators[11])
            {
                indicatorsReadiness[8] = 100;
            }

            // считаем 8 показатель
            indicatorsReadiness[7] = indicators[9] == 1 ? 100 : 0;

            return (int)indicatorsReadiness.Average();
        }

        public static int[] GetReadiness(int[,] indicators, bool gender)
        {
            int[] readiness = new int[4];

            for (int i = 0; i < readiness.Length; i++)
            {
                readiness[i] = CalculateReadiness(GetRow(indicators, i), gen-
der);
            }

            return readiness;
        }


        private static T[] GetRow<T>(T[,] array, int row)
        {
            if (!typeof(T).IsPrimitive)
                throw new InvalidOperationException("Not supported for man-
aged types.");

            if (array == null)
                throw new ArgumentNullException("array");

            int cols = array.GetUpperBound(1) + 1;
            T[] result = new T[cols];
            int size = Marshal.SizeOf<T>();
```

```
            Buffer.BlockCopy(array, row * cols * size, result, 0, cols *
size);

            return result;
        }
        private class Interval
        {
            public int start;
            public int end;

            public Interval(int start, int end)
            {
                this.start = start;
                this.end = end;
            }
        }
    }
}
```