МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерних наук

ЯВОРНИЦЬКИЙ Іван Михайлович

Веб-орієнтована система планування графіку відвідування спортивного залу ТНЕУ/ Web-based system for planning the visit schedule of the TNEU gym

напрям підготовки: 6.050103 - Програмна інженерія
фахове спрямування - Програмне забезпечення систем

Бакалаврська дипломна робота

Виконав студент групи ПЗС-42
І. М. Яворницький

Науковий керівник:
к.т.н., старший викладач
ВОЙТЮК І.Ф.

Бакалаврську дипломну роботу
допущено до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____ **А. В. Пукас**

РЕЗЮМЕ

Дипломна робота містить 81 сторінок, 20 таблиць, 35 рисунків, список використаних джерел із 15 найменувань.

Метою дипломної роботи є розробка веб-орієнтованої системи планування графіку відвідувань спортивного залу ТНЕУ.

Об’єктом дослідження є процес роботи спортзалу ТНЕУ.

Предметом дослідження є застосування сучасних інформаційних технологій для розробки веб-орієнтованої системи планування графіку відвідувань спортивного залу ТНЕУ.

Методи розробки базуються на технології ASP.NET в середовищі розробки Visual Studio 2013. Для організації баз даних використано Microsoft SQL server Management Studio 2016.

Одержані результати полягають в розробці веб-сайту для планування графіку відвідувань спортивного залу ТНЕУ, що дозволить переглядати та формувати розклад відвідувань.

Ключові слова: web-додаток, планування графіку відвідувань, спортивний зал.

SUMMARY

Thesis contains 81 pages, 20 tables, 35 figures, list of sources with 15 titles.

The aim of the thesis is develop web-based system for planning schedule visits to the gym TNEU.

Object of research is the process of work the gym.

The subject of research is developing web-based system for planning schedule visits to the gym TNEU.

Methods of developing technology based on ASP.NET, in the environment of development of Visual Studio 2013. For organization of databases used Microsoft SQL server Management Studio 2016.

The resulting is creating a web-application for planning schedule visits to the gym TNEU, that allow create and view schedule visits.

Keywords: web-site, planning schedule visits, gym.

ЗМІСТ

ВСТУП

У сучасних умовах відвідування спортивного залу є цікавим процесом, до якого людина часто докладає увагу та зусилля. На сьогодні проблеми планування графіку передбачають аналіз взаємодії комплексу чинників, що визначають поведінку особи під час підготовки до тренувань. Тому актуальною є розробка веб-орієнтованої системи, що дозволить ефективно розпоряджатись часом, виділеним для проведення спортивних тренувань та змагань як працівників ТНЕУ, так і студентів.

Метою роботи є розробка веб-орієнтованої системи планування графіку відвідувань спортивного залу ТНЕУ.

Об’єктом дослідження є процес роботи спортзалу ТНЕУ.

Предметом дослідження є застосування сучасних інформаційних технологій для розробки веб-орієнтованої системи планування графіку відвідувань спортивного залу ТНЕУ.

Розроблювана система має практичну цінність, адже може бути застосована для роботи спортзалу або інших, схожих за організацією спортзалів.

У роботі використано методи системного аналізу при дослідженні предметної області та визначенні бізнес-процесів, методи моделювання при дослідженні бізнес-процесів, методи об’єктно-орієнтованого аналізу та проектування при проектуванні системи; методи об’єктно-орієнтованого програмування при реалізації програми. Для розробки додатку з трирівневою архітектурою було використано технології ASP.NET для створення веб-застосунку і для реалізації реляційної бази даних – Microsoft SQL server Management Studio 2016.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Коротка характеристика об’єкту управління

Життя в ритмі спорту є запорукою здоров’я та підтримки відмінної фізичної форми на роки. Заняття спортом – це певна сукупність фізичних вправ та методичних прийомів, виконання котрих позитивно впливає на організм людини.

Спортивна інфраструктура надає безліч можливостей занять спортом і для аматорів, і для професійних спортсменів. Багато колишніх професійних спортсменів надалі пов’язують своє життя із спортом, тренуючи молоде покоління. Різноманітні спортивні секції та спеціальні школи направляють зусилля на формування та розвиток хорошого відношення до спорту, розпочинаючи з шахів та закінчуючи культуризмом. Основною метою діяльності офіційних спортивних організацій є проведення спортивних змагань на всеукраїнському рівні та на рівні університету.

У випадках, коли з професійним спортом з певних причин не склалося, то аматорський спорт і активний відпочинок є доступним для широкого кола всіх охочих вести здоровий спосіб життя. Для себе можна обрати будь-який вид спортивних занять, розпочинаючи з екстремальних, завершуючи щоденною ранковою гімнастикою. Також можна відвідувати спортивні клуби, спортзали, фітнес-центри, тренажерні зали або басейн, займатися з тренером чи обрати групові заняття.

Періодичне відвідування спортивних установ (тренажерний зал, спортзал, басейн і т.д.) або самостійне виконання фізичних вправ дозволяє не тільки виробити самодисципліну, але й сприяють всебічному розвитку, гартують організм і оздоровлюють організм. Заняття спортом призначені зберегти здоров’я, надати заряд бадьорості і гарний настрій.

Об’єктом управління є процес роботи спортивного залу ТНЕУ. Він має відношення до всього апарату управління ТНЕУ, а організацію та контроль

здорового способу життя працівників вузу та студентів проводить кафедра фізичної культури та спорту.

Діяльність кафедри здійснюється за трьома напрямками:

- розвиток спорту вищих досягнень (підготовка спортсменів високого класу);

- розвиток масового спорту (підготовка студентів для участі у внутрішніх спортивних заходах навчального закладу та обласних студентських змагань);

- впровадження та популяризація новітніх оздоровчих програм[5].

Працівники цих підрозділів відповідають за спортзали та результати їхньої діяльності. Також вони формують відвідуваність об’єктів, які належать даній кафедрі та стежать за матеріальними речами та їх використанням.

Для кращого розуміння взаємодії об’єкту управління та кафедри фізкультури та спорту які є структурною частиною ТНЕУ, на рисунку 1.1. представлено діаграму організаційної структури ТНЕУ.

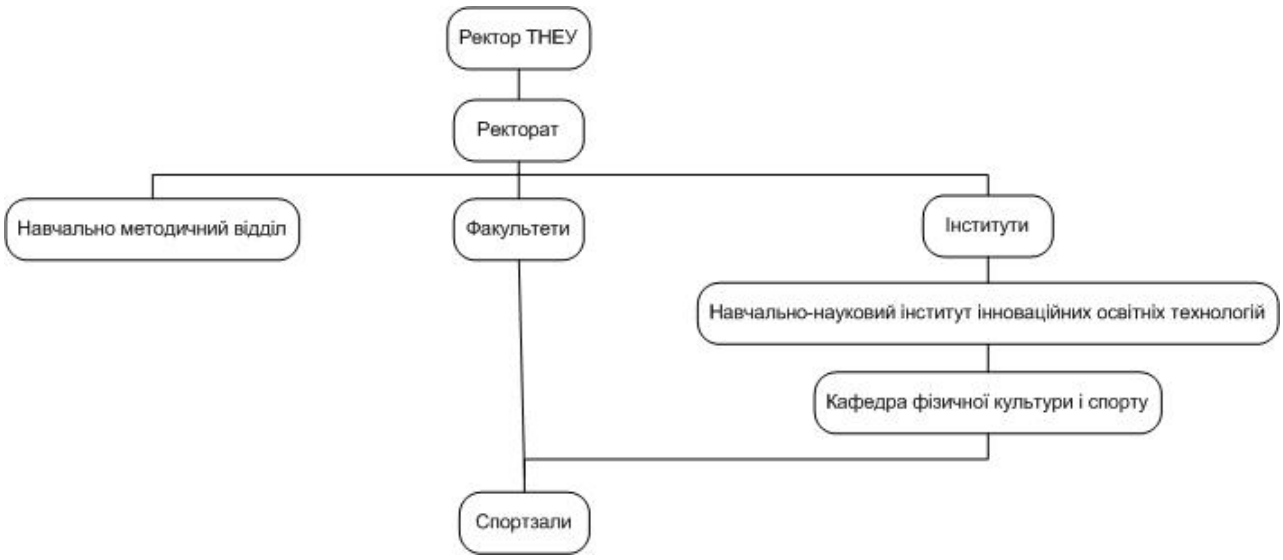


Рис.1.1. Організаційна структура ТНЕУ

При формуванні графіку часто виникають конфлікти та виникає багато проблем пов’язаних з управлінням ресурсів спортзалу, як матеріальних (інвентар), так і організаційних (наявність працівників спортзалу та їх

завантаженості). Тому необхідно покращити систему планування відвідувань та забезпечити реалізацію таких бізнес-процесів:

- введення інформації, пов’язаної із спортивним залом;
- редагування графіку відвідувань;
- зберігання та зручний перегляд графіку;
- автоматичне планування графіків відповідно до виду спорту, секції, тощо;
- автоматизоване створення та друк звітної документації.

1.2. Опис предметної області

Комп’ютеризація у всіх сферах діяльності людини залежить від масштабу та структури того, що необхідно комп’ютеризувати. Головною ціллю є створення єдиної інформаційної структури, що б давала можливість людям легко і правильно заносити інформацію до системи управління, швидко знаходити необхідну інформацію та зручно нею обмінюватись. Для досягнення мети будь-які процеси взаємодії людей стають зручними, зрозумілими та більш контрольованими, займаючи на порядок менше затраченого часу. Тому гроші, що витрачаються на комп’ютеризацію, необхідно обґрунтувати саме тими цілями, досягнення яких зробить той чи інший процес максимально ефективними.

Предметною областю дипломної роботи є автоматизація планування графіку відвідувань спортзалу, яка пришвидшить процес внесення нових подій спортзалу, формування та перегляд графіку відвідувань. Додаток розроблений в розрахунку на різні рівні управління роботою спортивного залу:

- можливість узагальнення даних;
- забезпечення єдиного підходу до порядку організації інформації;
- можливість постійного оперативного впливу на процес роботи спортивного залу;
- значне підвищення доступу до аналітичної інформації.

Для того, щоб повністю представити функціонал додатку, виділено основні бізнес-процеси, які представлені на рисунку 1.2.

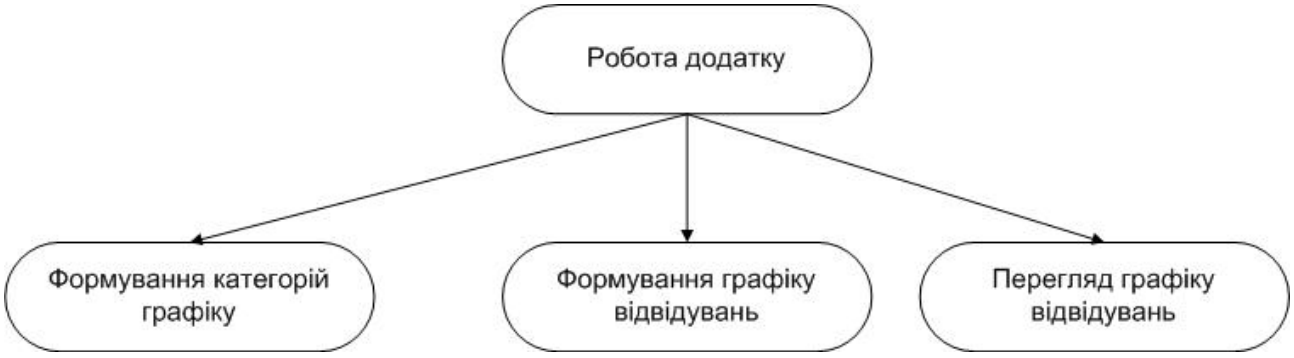


Рис.1.2. Діаграма бізнес-процесів розроблюваного програмного продукту

Тепер детальніше розглянемо кожен з вище представлених бізнес-процесів. На рисунку 1.3. зображено процес формування категорій графіку.

Рис.1.3. Діаграма функцій процесу формування категорій графіку

Перше, що потрібно зробити адміністратору, – це сформувати категорію, для подальшого додавання її в БД. Формування категорій поділяється на такі особливості:

- Вибір типу – вибір виду відвідування, наприклад, пари з фізкультури для груп факультетів, секції для кафедри спорту, тренажерний зал.
- Опис характеристик, ними можуть виступати для прикладу відповідальний за подію, кількість місць у залі.

Характеристику бізнес-процесу формування категорій графіку адміністратором наведено в таблиці 1.1.

Таблиця 1.1

Характеристика бізнес-процесу формування категорій

Назва характеристики	Значення характеристики
Ім’я бізнес-процесу	Формування категорій графіку
Основні учасники	Адміністратор
Вхідна подія	Запит на формування категорій
Вхідні документи	Інформація про викладача чи тренера, що проводить пару з фізкультури чи тренування
Вихідна подія	Перегляд сформованих категорій
Вихідні документи	Сформовані категорії відвідування
Клієнт бізнес-процесу	-

На рисунку 1.3. бачимо діаграму функцій процесу формування графіку відвідувань (виконання завдання).

Рис.1.4. Діаграма функцій процесу формування графіку відвідувань

Наступне, що потрібно зробити адміністратору, – це додати до категорій, записи для відвідувань і додавання їх в БД. Це найголовніший процес, адже в ньому формується графік відвідувань. Формування графіку поділяється на такі особливості:

- Вибір часу проведення – це вибір часу проведення та періодичність.
- Опис події – містить місце проведення події та відповідальних за її проведення чи тренерів (якщо це тренування спортивної секції)

- Внесення в графік відвідування – це проведення самого внесення про відвідування в БД.

Характеристику бізнес-процесу формування графіку відвідувань адміністратором наведено в таблиці 1.2.

Таблиця 1.2.

Характеристика бізнес-процесу формування графіку відвідувань

Назва характеристики	Значення характеристики
Ім’я бізнес-процесу	Формування графіку відвідувань
Основні учасники	Адміністратор
Вхідна подія	Запит на внесення нового відвідування
Вхідні документи	Інформація про інвентар і актуальний графік відвідування
Вихідна подія	Перегляд графіку відвідувань
Вихідні документи	Оновлений графік відвідування
Клієнт бізнес-процесу	Процес формування категорій графіку

На рисунку 1.5 наведемо діаграму функцій процесу перегляду графіку відвідувань спортивного залу ТНЕУ.

Рисунок 1.5. Діаграма функцій процесу перегляду графіку відвідувань

Процес перегляду графіку відвідувань є останнім кроком і служить для зручного відображення графіку для користувачів системи. Він поділяється на такі частини:

- за часом проведення;

- за категорією відвідування ;
- за кількістю вільних місць.

Характеристику бізнес-процесу перегляду графіку відвідувань наведено в таблиці 1.3.

Таблиця 1.3.

Характеристика бізнес-процесу перегляду графіку відвідувань

Назва характеристики	Значення характеристики
Ім’я бізнес-процесу	Перегляд графіку відвідування
Основні учасники	Користувач, адміністратор.
Вхідна подія	Запит на здійснення перегляду
Вхідні документи	Запит на перегляд
Вихідна подія	Відображення графіка по необхідній категорії
Вихідні документи	Актуальний графіку
Клієнт бізнес-процесу	Процес додавання формування графіку відвідувань

Автоматизація роботи спортивного залу сприяє підтриманню бази даних в актуальному стані і отриманню статистично достовірної інформації. Це має значно підвищувати швидкість створення та доступ до інформації щодо графіку відвідувань і дає можливість постійного оперативного впливу на процес управління.

Додаток має зручний інтерфейс системи, що надає змогу легко та швидко працювати в його середовищі. Основним користувачем системи є працівники спортзалу відповідальні за створення графіку. Усі ці можливості позбавляють даних працівників спортзалу роботи по заповненню документів, плутанини в паперовій документації та помилок.

1.3. Огляд і аналіз існуючих аналогів

На сьогодні є достатньо програмних систем, за допомогою яких можна здійснити планування будь-яких подій. Проте систем за допомогою яких можна здійснити планування відвідувань спортивного залу на порядок менше.

Першим, розглянуто програмний продукт «Програма для спортзалу», який можна завантажити з сайту usu.kz. Головна сторінка цієї програми представлена на рисунку 1.6.

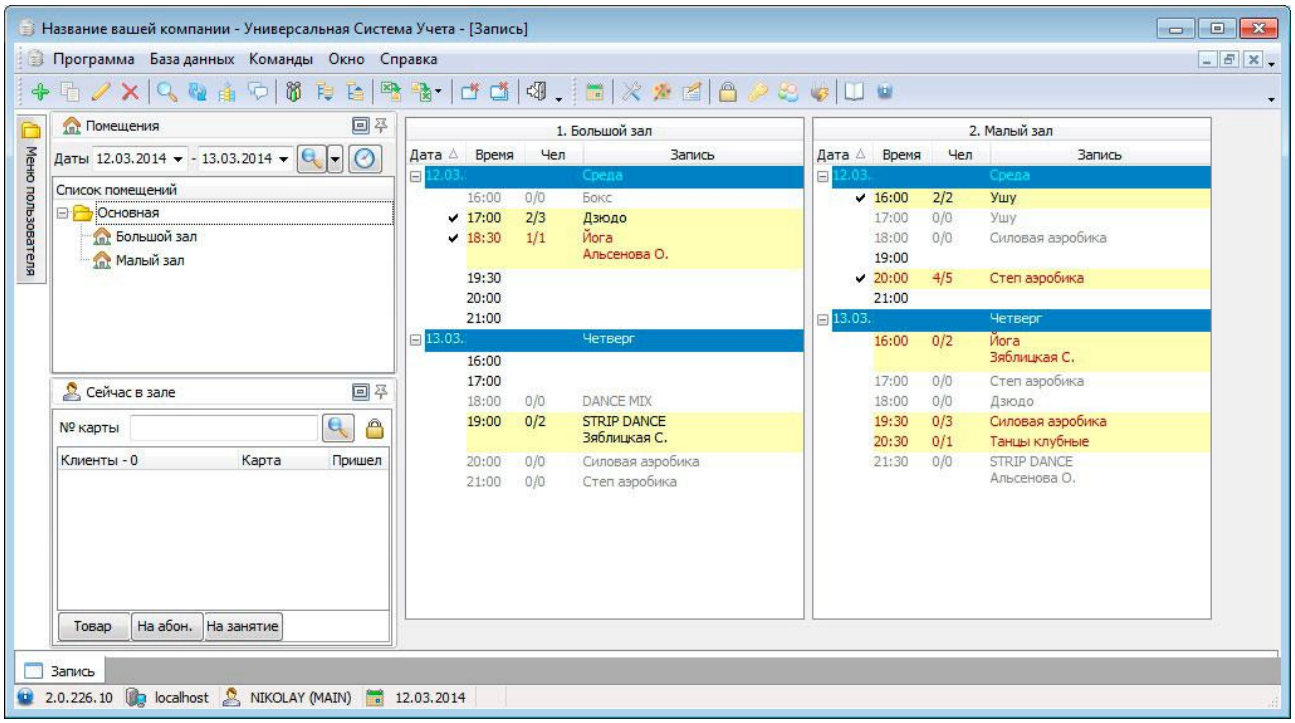


Рис.1.6. Головна сторінка «Програми для спортзалу»

Дана програма має дуже простий і достатньо зручний інтерфейс, за допомогою якого користувач може досить легко зорієнтуватися в подальших діях. Для здійснення обліку користувачу не потрібно проходити реєстрацію. Проте, щоб використовувати додаткові функції, потрібно купити програмний продукт.

Дана програмна система має наступні функції:

- облік графіку відвідувань;
- історія графіку відвідувань;
- смс розсилка;
- можливість перенесення занять.

Також, для порівняння, розглянемо програмний продукт «Контроль відвідувань спортзалу». Головна сторінка програми представлена на рисунку 1.7.

Сторінку списку клієнтів наведено на рисунку 1.8., а сторінку аналізу відвідувань наведено на рисунку 1.9.

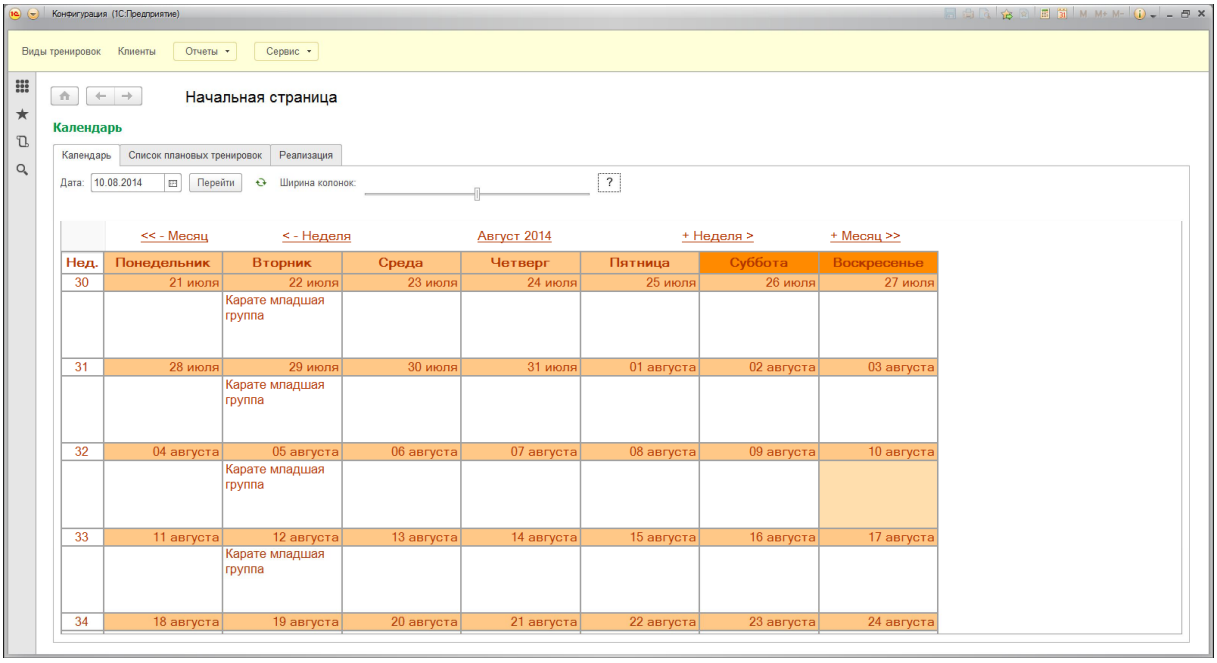


Рис.1.7. Головна сторінка програми «Контроль відвідувань спортзалу»

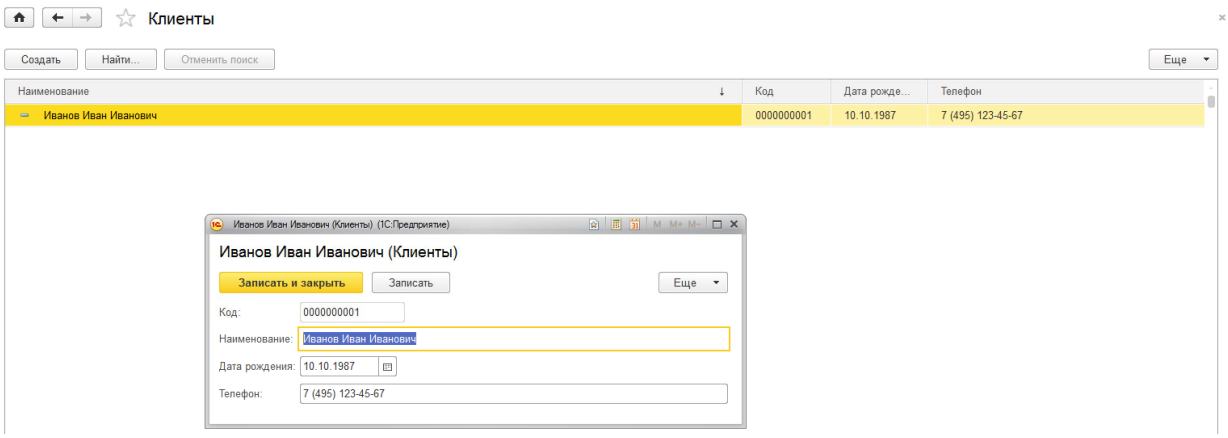


Рис.1.8. Сторінка списку клієнтів програми «Контроль відвідувань спортзалу»

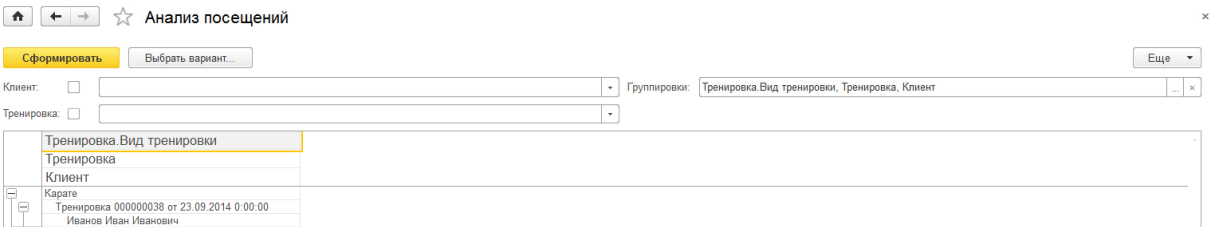


Рис.1.9. Сторінка аналізу відвідувань програми «Контроль відвідувань спортзалу»

Даний продукт дозволяє проводити облік відвідувань, реєструвати їх та проводити зміни. Цей продукт має багато різноманітних функцій, таких як:

- планування тренувань;
- ведення списку клієнтів;
- контроль відвідуваності;
- контроль оплат;
- пошук і групування даних.

Розглянемо ще один онлайн сервіс «Отмечалка». Головну сторінку показано на рисунку 1.10.

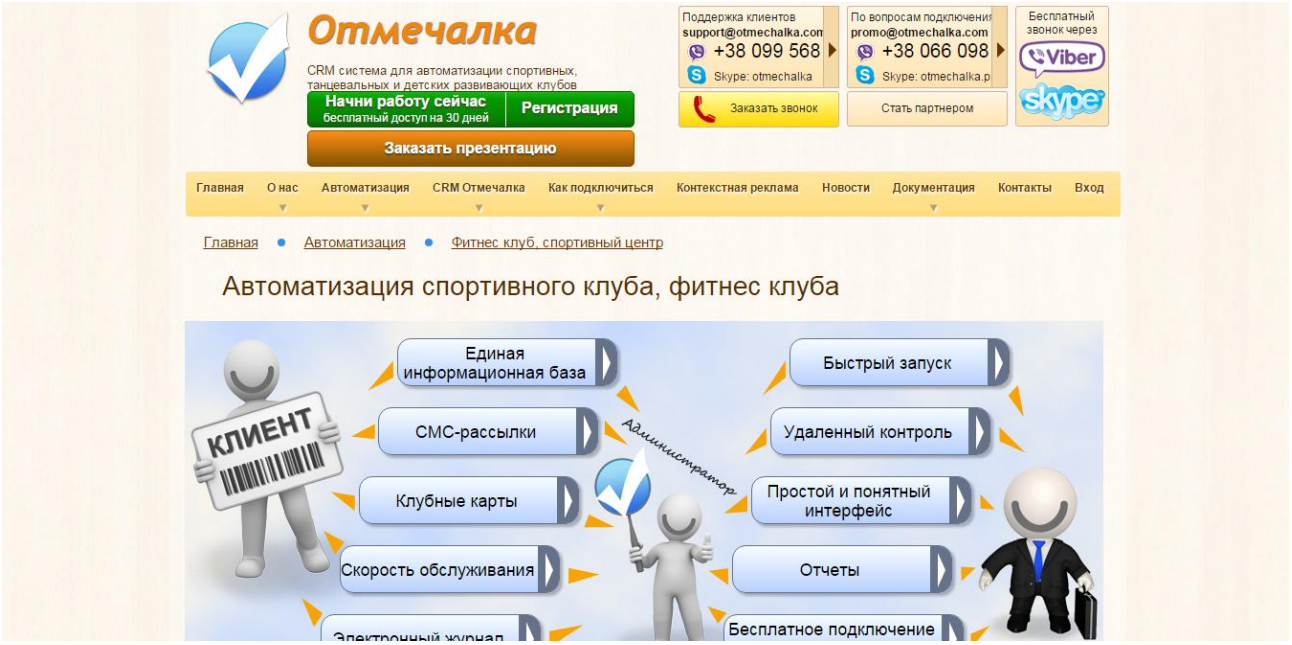


Рис.1.10. Головна сторінка ресурсу «Отмечалка»

Інтерфейс даного сервісу є дуже зрозумілий і цікавий для користувача. Перевагою є онлайн режим.

Основними функціями є:

- планування тренувань;
- ведення списку клієнтів;
- контроль відвідуваності;
- контроль оплат;

- пошук і групування даних;
- облік графіку відвідувань;
- історія графіку відвідувань.

Хоча програмний продукт і має велику кількість хороших функцій, він доступний лише у платній версії.

Здійснивши аналіз альтернатив, створено таблицю, де відображається порівняльна характеристика аналогів.

Таблиця 1.4.

Порівняльна характеристика програмних продуктів

Фірма-розробник	УСУ – Універсальная Система Учета	Фізична особа	«Отмечалка»
Назва програмного продукту	«Програма для спортзалу»	Контроль відвідувань спортзалу	Автоматизація спортивного клуба, фітнес клуба
Допомога користувачу	Відео-огляд про програму та презентація	Наведений опис програми і вигляді захоплень екранів	Сайт з допоміжними фото та зауваженнями
Ціна	Існує платна і демо-версія	Безплатна версія	Тільки платна версія
Сайт, де можна скачати	http://usu.kz/programma_dlya_zala.php	http://forum-1c.ru/index.php?topic=41989.0	https://otmechalka.com/index/page/avtomatizaciya-fitnes-centrov-i-sport-klubov

Оглянувши та зробивши аналіз існуючих аналогів можна зробити висновок, що вище перелічені програмні системи є хорошими для проведення планування відвідувань приватних спортивних залів. Кожна програма функціонує досить добре. Але, переглянувши функції кожного програмного

продукту, можна побачити, що кожна система різна і орієнтована лише на російськомовних користувачів. Окрім цього для планування графіку відвідувань спортивного залу ТНЕУ необхідна більш вузькоспеціалізована система, яка не буде загромаджена зайвим функціоналом, – орієнтована на організаційну структуру університету.

1.4. Специфікація вимог до системи

Специфікація вимог до програмної системи – це повний опис поведінки системи, яка розробляється. Вона включає в себе множину прецедентів, що описують всі взаємодії, що користувачі мають з програмним забезпеченням. Прецеденти відомі також як функціональні вимоги. Окрім прецедентів також включає нефункціональні вимоги. Нефункціональні вимоги – вимоги що накладають обмеження на проект чи його реалізацію, що не пов’язані з його основним функціоналом. Специфікація вимог до системи включає: глосарій проекту, опис варіантів використання.

У таблиці 1.5. подано глосарій основних використовуваних термінів при створенні системи планування відвідувань спортзалу.

Таблиця 1.5.

Глосарій основних термінів

Термін	Опис терміну
1. Основні поняття та категорії предметної області та проекту	
Спортзал	Спортивний зал індивідуального або колективного відвідування
Клієнт	Відвідувач спортзалу або керівник спортивного гуртка
Секція	Періодичне відвідування спортзалу групою під керуванням тренера протягом певного часу
Графік відвідувань	сукупність елементів, що становлять систематизацію відвідувань спортзалу

Продовження таблиці 1.5.

Абонемент	Форма яка дозволяє відвідувати тренажерний зал протягом певного періоду
Одноразове відвідування	Форма на відвідування спорт залу, яка передбачає одиничне відвідування
Інтерфейс	Засіб зручної взаємодії користувача з інформаційною системою, що дозволяє взаємодіяти з системою з високою ефективністю
Тип відвідування	Об’єднує в собі всі види секцій, абонемент і одноразове відвідування.
2. Користувачі системи	
Замовник	Особа, що використовує систему планування відвідувань спортзалу щоб отримати доступ до спортзалу в певний час
Користувач	Особа що хоче переглянути графік
Адміністратор	Користувач системи, який здійснює підтримку системи для обліку системи замовлень
3. Вхідні та вихідні документи	
Заяка	Комерційний документ у процесі замовлення, який видається клієнтом адміністратору й зазначає тип, тривалість, періодичність та іншу інформацію про необхідний запис у графіку
Звіт	Письмове повідомлення про виконання певної роботи
База даних	Впорядкований набір логічно взаємопов’язаних даних, що використовуються спільно та призначені для задоволення інформаційних потреб користувачів

Далі, на рисунку 1.10. наведемо діаграму варіантів використання для акторів системи (замовник, адміністратор, користувач)

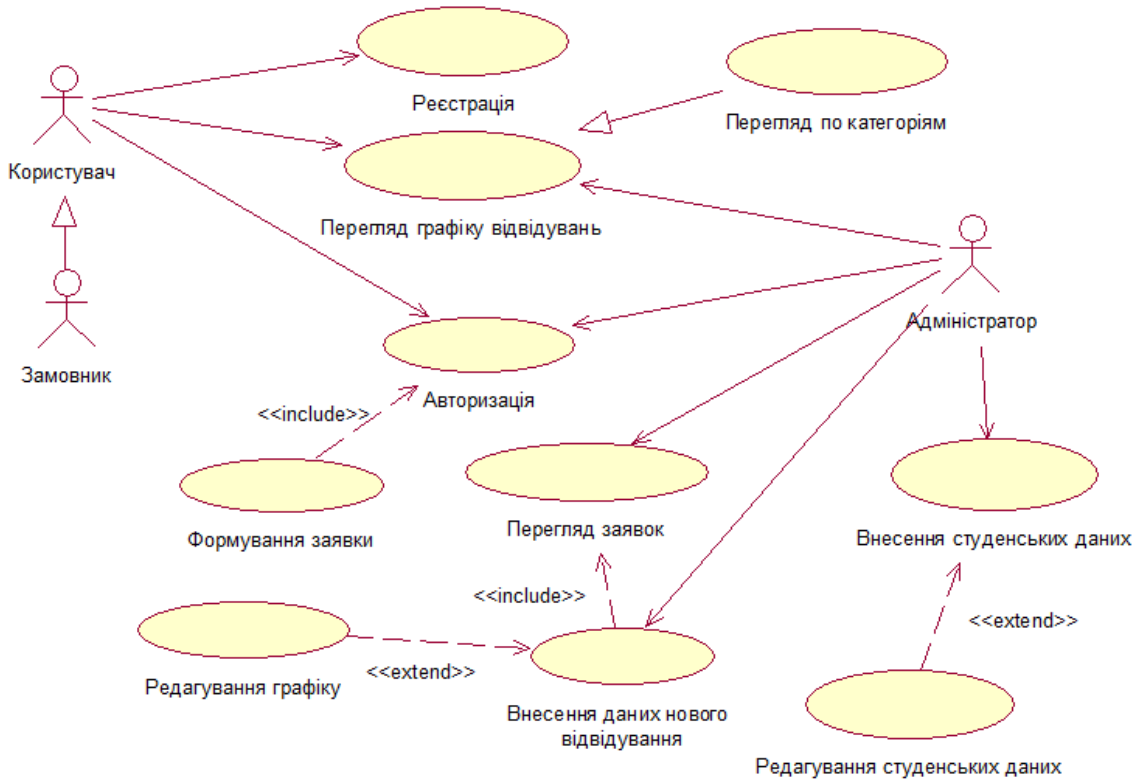


Рис.1.11. Діаграма варіантів використання

Наступним кроком у визначенні специфікацій вимог до системи буде опис варіантів використання.

Варіанти використання представленні у таблицях 1.6-1.13.

Варіант використання «Реєстрація» подано у таблиці 1.6.

Таблиця 1.6.

Опис варіанту використання «Реєстрація»

Контекст використання	Реєстрація
Дійові особи	Не зареєстрований користувач
Передумова	Користувач знаходиться на сторінці реєстрації
Тригер	Відкрита сторінка для реєстрації
Сценарій	1.Заповнюємо поле «Введіть логін»; 2.Заповнюємо поле «Введіть пароль»; 3.Заповнюємо поле «Введіть пароль ще раз»; 4.Натискаємо кнопку «Зареєструвати»
Пост-умова	Користувач зареєстрований

Варіант використання «Авторизація» подано у таблиці 1.7.

Таблиця 1.7.

Опис варіанту використання «Авторизація»

Контекст використання	Авторизація
Дійові особи	Не авторизований користувач, адміністратор
Передумова	Користувач знаходиться на сторінці авторизації
Тригер	Відкрита сторінка для авторизації
Сценарій	1.Заповнюємо поле «Введіть логін»; 2.Заповнюємо поле «Введіть пароль»; 3.Натискаємо кнопку «Увійти»
Пост-умова	Користувач, адміністратор авторизований

Варіант використання «Перегляд графіку відвідувань» подано у таблиці 1.8.

Таблиця 1.8.

Опис варіанту використання «Перегляд графіку відвідувань»

Контекст використання	Перегляд графіку
Дійові особи	Користувач, адміністратор, гість
Передумова	Користувач знаходиться на головній сторінці
Тригер	Відкрита головна сторінка
Сценарій	1.Клік на вкладку календаря
Пост-умова	Відкрита головна сторінка

Варіант використання «Внесення даних відвідування» подано у таблиці 1.9.

Таблиця 1.9.

Опис варіанту використання «Внесення даних нового відвідування»

Контекст використання	Додавання відвідування
Дійові особи	Адміністратор
Передумова	Авторизація адміністратора
Тригер	Відкрита особиста сторінка
Сценарій	1. Обираємо дату; 2. Обираємо тип відвідування; 3. Обираємо час; 4. Якщо не суперечить актуальному розкладу, – натискаємо оформити.
Пост-умова	Збереження даних в БД

Варіант використання «Формування заявки» подано у таблиці 1.10.

Таблиця 1.10.

Опис варіанту використання «Формування заявки»

Контекст використання	Формування заявки
Дійові особи	Користувач
Передумова	Авторизація користувача
Тригер	Відкрита особиста сторінка
Сценарій	1. Обираємо тип відвідування; 3. Обираємо час; 4. Обираємо тривалість; 5. Натискаємо кнопку «Подати заявку»
Пост-умова	Збереження даних в БД

Варіант використання «Внесення студентських даних» подано у таблиці 1.11.

Таблиця 1.11.

Опис варіанту використання «Внесення студентських даних»

Контекст використання	Внесення студентських даних
Дійові особи	Адміністратор
Передумова	Авторизація Адміністратора
Тригер	Відкрита сторінка студентських даних
Сценарій	1. Додаємо необхідний факультет; 2. Додаємо необхідний групу; 3. Додаємо необхідного студента; 4. Вносимо дані про студента.
Пост-умова	Збереження даних в БД

Варіант використання «Перегляд по категоріям» подано у таблиці 1.12.

Таблиця 1.12

Опис варіанту використання «Перегляд по категоріям»

Контекст використання	Перегляд по категоріям
Дійові особи	Користувач, адміністратор
Передумова	Користувач знаходиться на головній сторінці
Тригер	Відкрита особиста сторінка
Сценарій	1. Обраємо категорію відвідування
Пост-умова	Відкрита головна сторінка

Варіант використання «Перегляд заявок» подано у таблиці 1.13.

Таблиця 1.13

Опис варіанту використання «Перегляд заявок»

Контекст використання	Перегляд заявок
Дійові особи	Адміністратор
Передумова	Авторизація адміністратора
Тригер	Відкрита особиста сторінка
Сценарій	1. Обираємо заявку 2. Натискаємо на неї 3. Переглядаємо зміст заявки
Пост-умова	Відкриття варіанту використання «Внесення даних нового відвідування»

Для представлення варіантів використання було здійснено розкадровку – створення прототипів системи для показу основної функціональності системи та вимоги до неї.

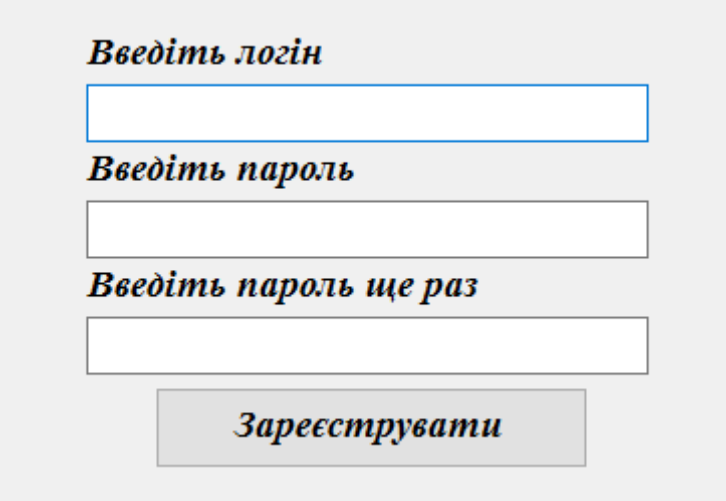


Рис.1.12. Вікно реєстрації

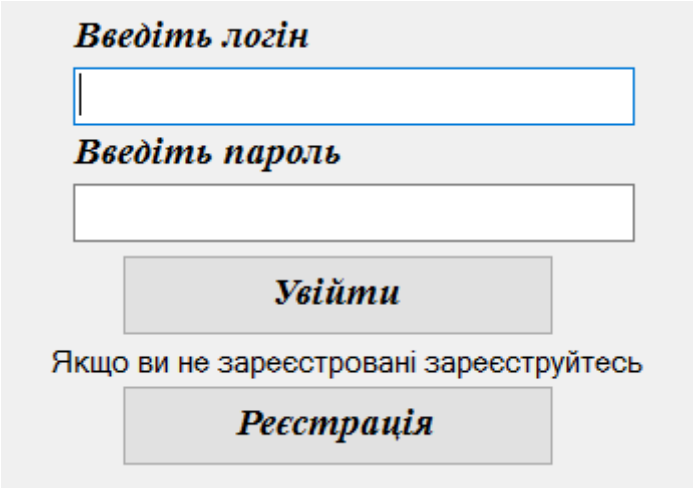


Рис.1.13. Вікно авторизації

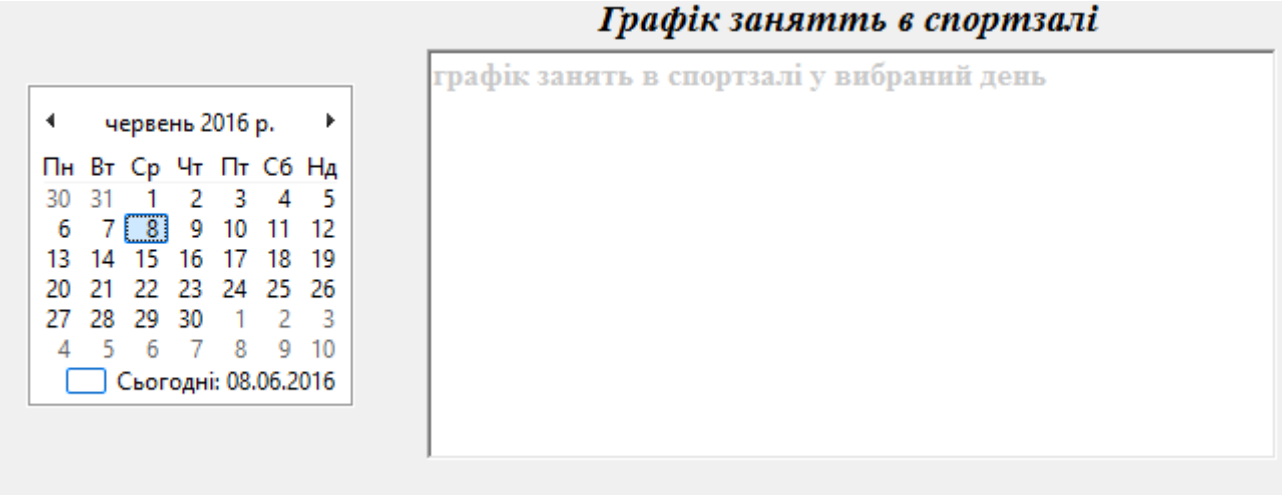


Рис. 1.14. Перегляд графіку відвідувань

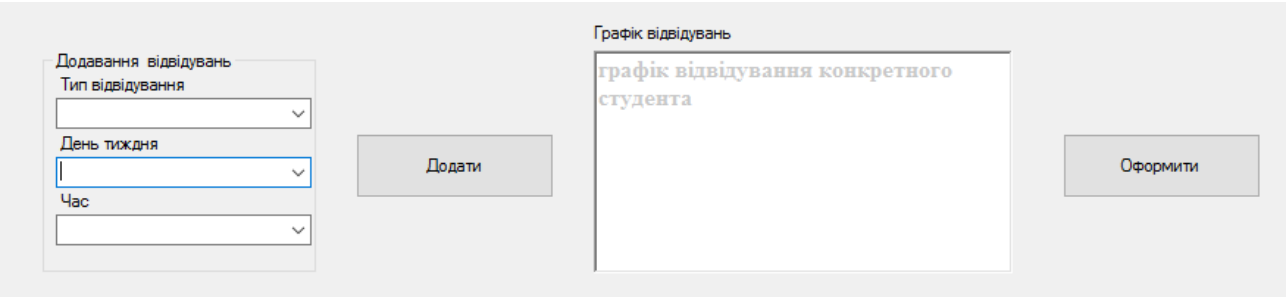


Рис.1.15. Обліковий запис адміністратора. Внесення даних нового відвідування

Рис.1.16. Обліковий запис користувача. Формування заявки

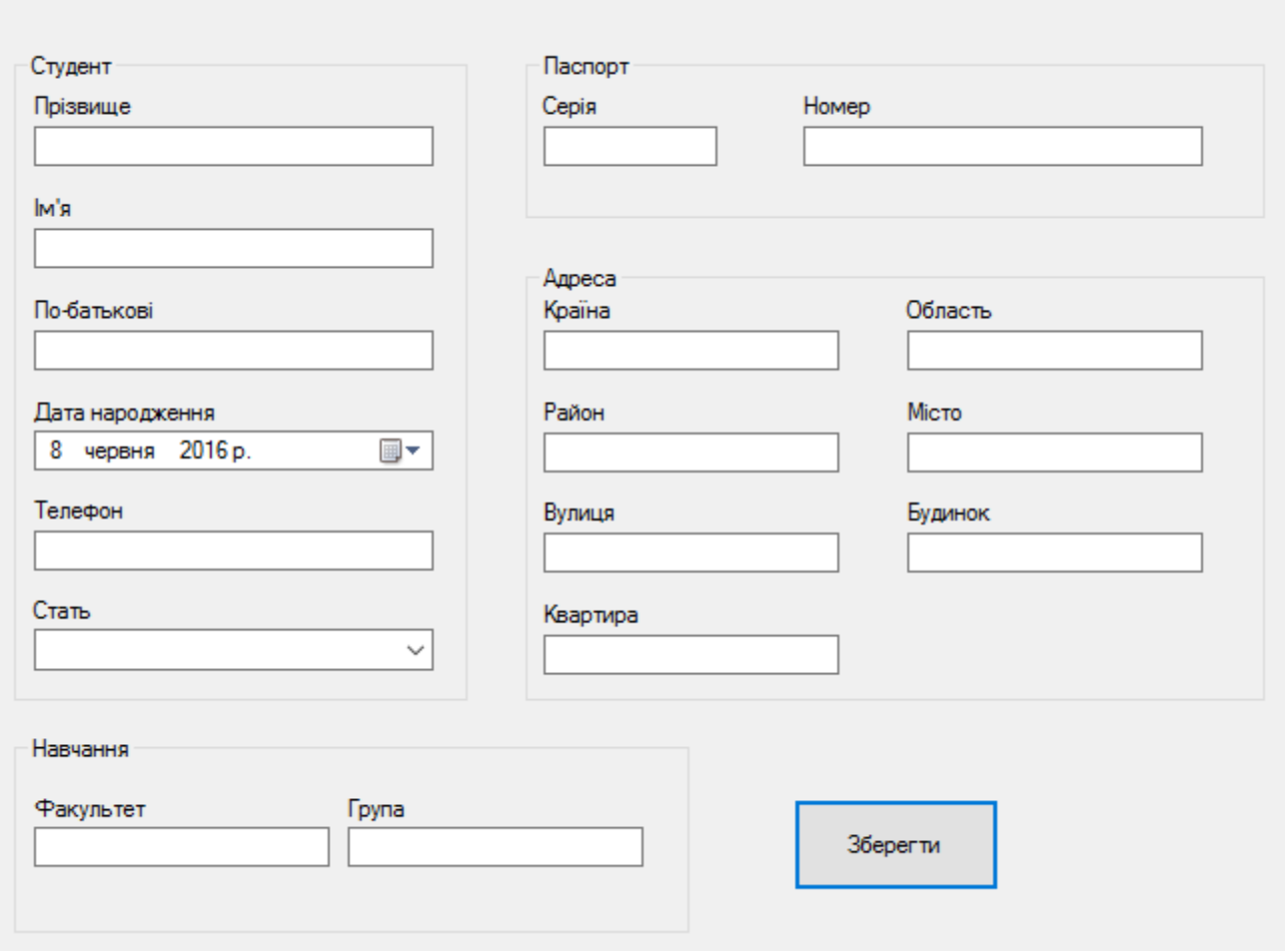


Рис.1.17.Внесення студентських даних

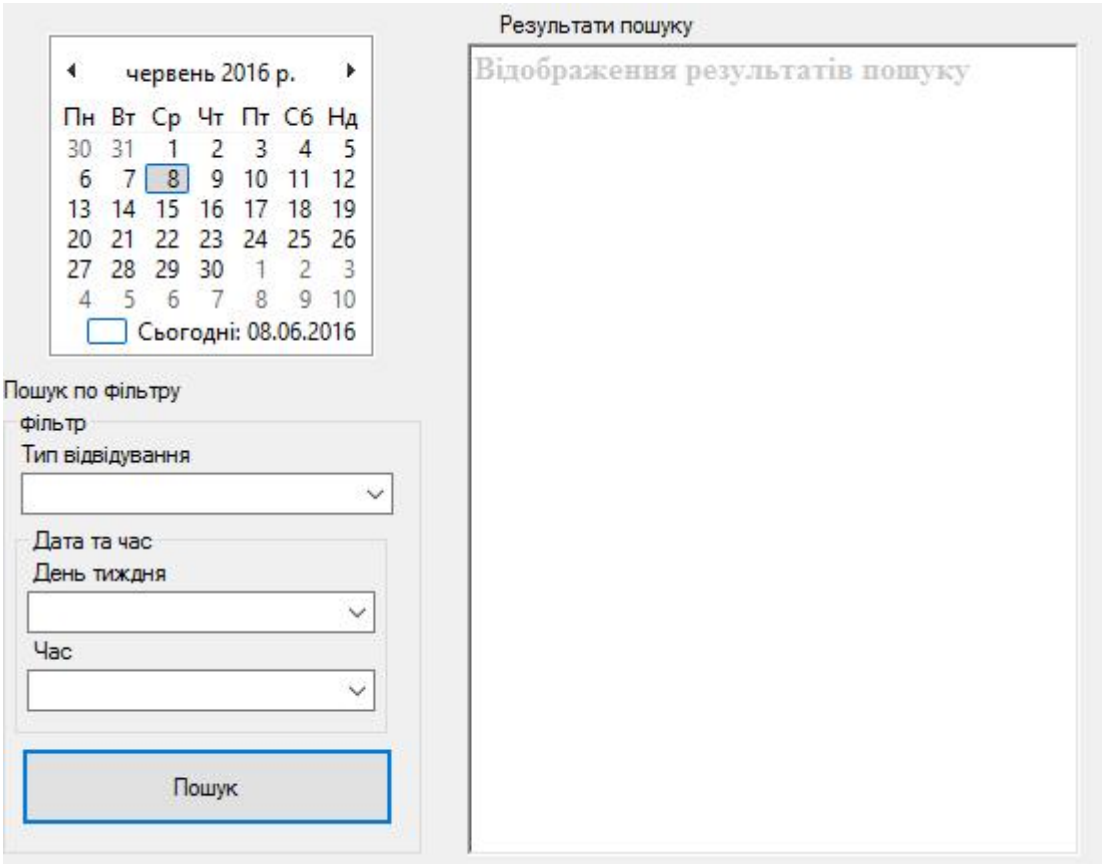


Рис.1.18. Обліковий запис адміністратора. Перегляд за критеріями

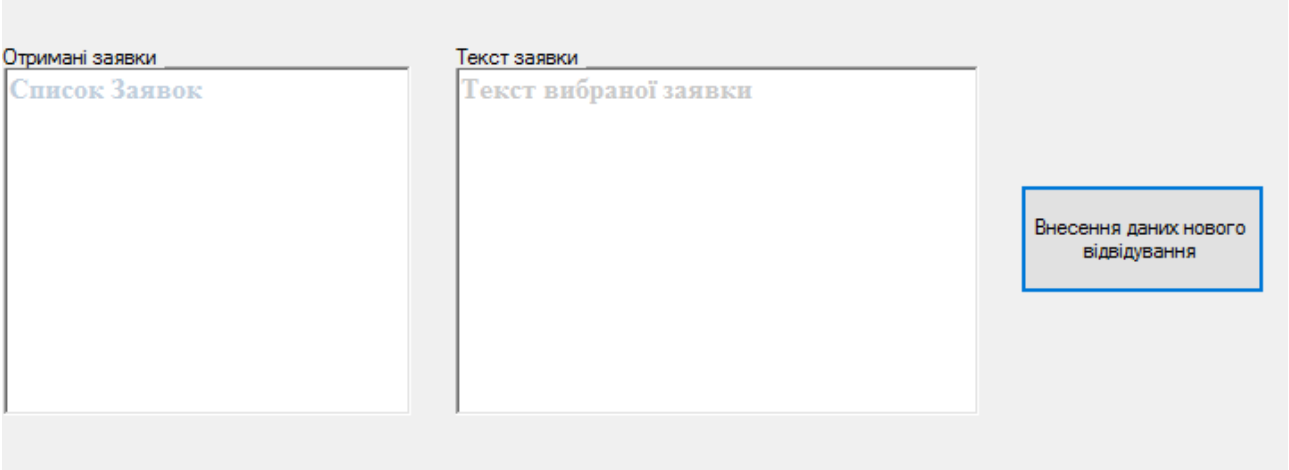


Рис.1.19. Обліковий запис адміністратора. Перегляд заявок

Провівши розкадровку, можна виокремити основні функціональні вимоги системи, які представлений у таблиці 1.14., та нефункціональні – у таблиці 1.15.

Таблиця 1.14.

Специфікація функціональних вимог

Ідентифікатори вимоги	Назва вимоги	Атрибут вимог		
		Пріоритет	Складність	Контакт
1	Авторизація	Обов’язкове	Середня	Адміністратор Користувач
2	Перегляд графіку відвідувань	Обов’язкове	Середня	Адміністратор Користувач
3	Перегляд запитів	Обов’язкове	Середня	Адміністратор
4	Формування запиту	Обов’язкове	Низька	Користувач
5	Внесення даних нового відвідування	Обов’язкове	Висока	Адміністратор
6	Внесення студентських даних	Обов’язкове	Середня	Адміністратор
7	Редагування студентських даних	Рекомендоване	Складна	Адміністратор
8	Редагування графіку відвідувань	Рекомендоване	Середня	Адміністратор

Таблиця 1.15.

Специфікація нефункціональних вимог

Ідентифікатор вимоги	Назва вимоги	Атрибути вимог		
		Пріоритет	Складність	Контакт
1. Застосовність				
1.1	Основні вимоги застосовності нової системи відносно інших систем, які знають користувачі	Рекомендована	Низька	Адміністратор, користувач
1.2	Вимоги по відповідальності і стандартам графічного інтерфейсу користувача	Обов’язкова	Низька	Адміністратор, користувач

Продовження таблиці 1.15.

1.3	Час, необхідний для навчання звичайних і досвідчених користувачів	Рекомендована	Середня	Адміністратор, користувач
2. Надійність				
2.1	Доступність	Обов’язкова	Середня	Адміністратор, користувач
2.2	Середній час безвідмовної роботи	Обов’язкова	Середня	Адміністратор, користувач
2.3	Точність	Обов’язкова	Середня	Адміністратор
3. Робочі характеристики				
3.1	Використання ресурсів	Рекомендована	Середня	Адміністратор
4. Проектні обмеження				
4.1	Вимоги до технології програмування	Рекомендована	Середня	Адміністратор

Значення нефункціональних вимог:

- час, необхідний для навчання звичайних користувачів – 30 хвилин;

- час, необхідний для навчання досвідчених користувачів – 15 хвилин;

- основні вимоги застосовності нової системи відносно інших систем, які знають користувачі – всі функції системи є легкими у виконанні, а структура програми не відрізняється від існуючих аналогів;

- вимоги по відповідності загальним стандартам застосовності та стандартам графічного інтерфейсу користувача – додаток повинен працювати у всіх ведучих браузерах;

- доступність – час, що витрачається на обслуговування системи не повинен перевищувати 3% від загального часу роботи;

- середній час безвідмовної роботи – 3 години;

- використання ресурсів – мінімальні системні вимоги:

o Об’єм HDD: > 20 Мб;

- Об’єм RAM: > 512 Мб;
- Відео пам’ять: > 128 Мб;
- Процесор: > 1 ГГц;
- Клавіатура, маніпулятор миша;
- Доступ до інтернет;
- Веб браузер.

ВИСНОВКИ ДО ПЕРШОГО РОЗДІЛУ:

1. Визначено специфікацію вимог до розроблюваної системи та проаналізовано об’єкт автоматизації.
2. Здійснено порівняння трьох веб-сайтів. В результаті порівняння було прийнято рішення створення веб-орієнтованої системи планування графіку відвідувань спортивного залу ТНЕУ.
3. Наведено діаграми варіантів використання для акторів системи та здійснено їх розкадровку для наочного показу функцій, які необхідно реалізувати в системі.
4. Визначено всі функціональні та не функціональні вимоги до системи, що дозволить перейти до проектування програмної системи.

РОЗДІЛ 2 ПРОЕКТУВАННЯ

2.1. Розроблення архітектури програмної системи

Для розробки додатку було обрано трирівневу архітектуру системи або модель прикладного сервера (AS — Application Server). У цій моделі реалізовано трирівневу систему розподілу функцій (Рисунок2.1).

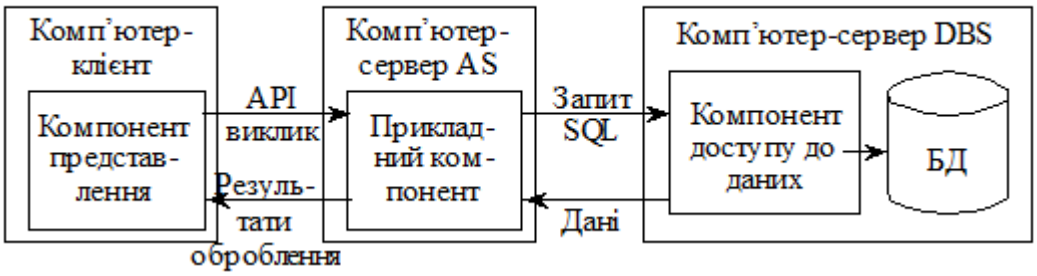


Рис. 2.1. Модель трирівневої архітектури

Першим рівнем є комп'ютер клієнта, де розміщений користувацький інтерфейс та функції локального редагування та логіка для перевірки даних. Також системі необхідно взаємодіяти з мережею. Тобто на клієнті виконуються функції першої групи, що відповідають за інтерфейс із користувачем. Звертаючись до компонента, що розміщений на сервері, комп'ютер виступає клієнтом додатку.

Другим рівнем є прикладний сервер, який є відмінною ознакою трирівневої архітектури клієнт-сервер. Основне його призначення – зберігання та виконання бізнес-правил. Він реалізований як група процедур, які виконують прикладні функції та називається прикладним сервером або Application Server. Виокремлення прикладної логіки в самостійний архітектурний рівень дозволяє реалізувати її поширеними мовами програмування (C, C++, C#, Cobol, php), які мають великі переваги з мовами роботи із БД, оскільки вони – достатньо спеціалізовані. Завдяки виокремленню прикладної логіки підвищується незалежність функціональних компонентів одного рівня від змін або вдосконалень компонентів іншого.

Третій рівень – сервер бази даних. Він відповідає за зберігання та підтримку даних, включаючи також їх узгоджене перетворення, попередження несанкціонованого чи некоректного коригування БД, створення резервних копій. Тобто він забезпечує осмислення інформації, що зберігається в БД[6,8].

Для опису бізнес логіки і наочного представлення на використано UML-діаграму класів (Рисунок 2.2.)

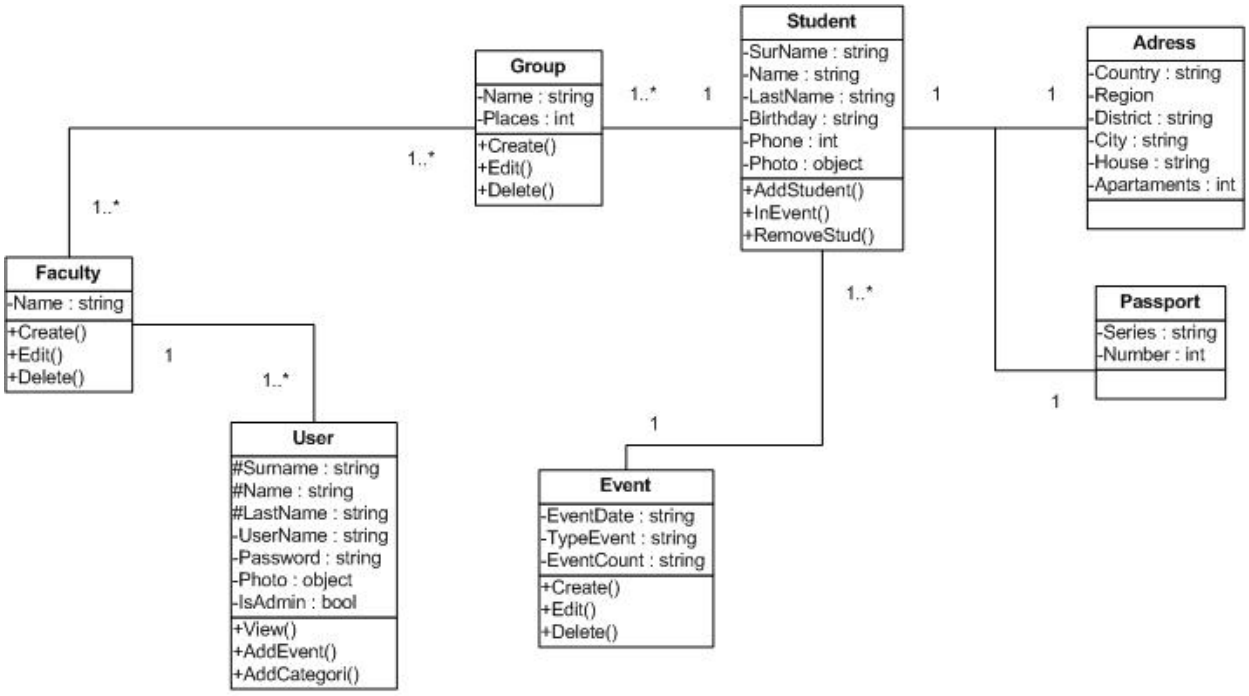


Рис.2.2. UML-діаграма класів

На даній діаграмі представлені прототипи основних класів системи на зв’язків між ними. Описано функції додавання, редагування, видалення, перегляду графіку відвідувань та внесення нових категорій у графік

Функціональну структуру системи умовно можна розділити на такі модулі:

- модуль обробки інформації про користувачів:
  1) реєстрація;
  2) авторизація;
- модуль обробки інформації про графіки відвідувань:

1) додавання відвідування;
2) Додавання студентських даних

3) перегляд графіку відвідування.

Для того, щоб краще зрозуміти процес функціонування системи, побудовано діаграми активності для кожного модуля.

Діаграму активності авторизації користувача зображено на рисунку.2.3.

Рис.2.3. Діаграма активності процесу авторизації користувача

Діаграму активності реєстрації користувача зображено на рисунку 2.4.

Рис.2.4. Діаграма активності для процесу реєстрації користувача

Діаграму активності для модуля обробки інформації про графіки відвідувань на рисунку 2.5.

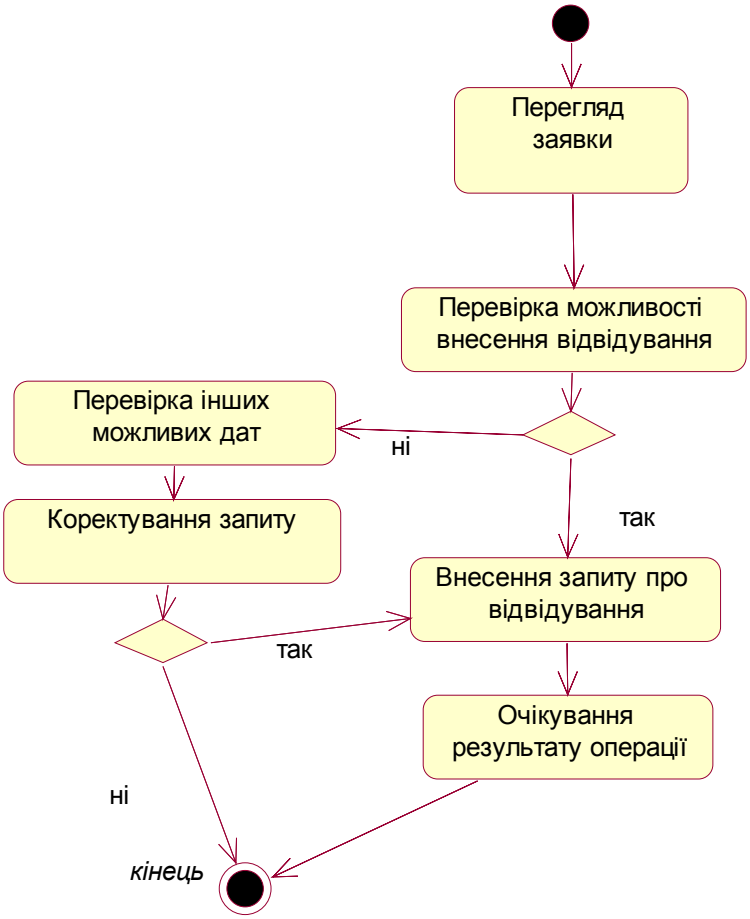


Рис.2.5. Діаграма активності додавання відвідування

2.2. Проектування структури бази даних

Етап проектування бази даних (БД) вважається одним із самих складних етапів створення БД, який не має явно вираженого початку й закінчення.

Порівняно з аналізом вимог до БД або розробкою додатків, проектування БД, на думку багатьох провідних фахівців, є невдало структурованим завданням. Якщо всі етапи створення БД перекриваються один з одним у своїй послідовності, то етап проектування перекривається з усіма іншими етапами.

Проектування починається з моменту прийняття стратегічних рішень і триває на етапах реалізації й тестування[14].

Процес проектування БД охоплює кілька основних сфер:

- проектування об’єктів БД (таблиці, подання, індекси, тригери, збережені процедури, функції, пакети) для подання даних ПЗ в БД;
- проектування інтерфейсу взаємодії з БД (форми, звіти й т.д.), тобто проектування додатків, які будуть супроводжувати дані в БД і реалізовувати питально-відповідні відношення на цих даних;
- проектування БД під конкретне обчислювальне середовище або інформаційну технологію (архітектура «клієнт-сервер», паралельні архітектури, розподілене обчислювальне середовище);
- проектування БД під призначення системи (інтелектуальний аналіз даних, OLAP, OLTP і т.д.)[12].

Проектування бази даних слід розпочинати зі створення DFD-моделі предметної області. DFD-модель представлена на рисунку 2.6.

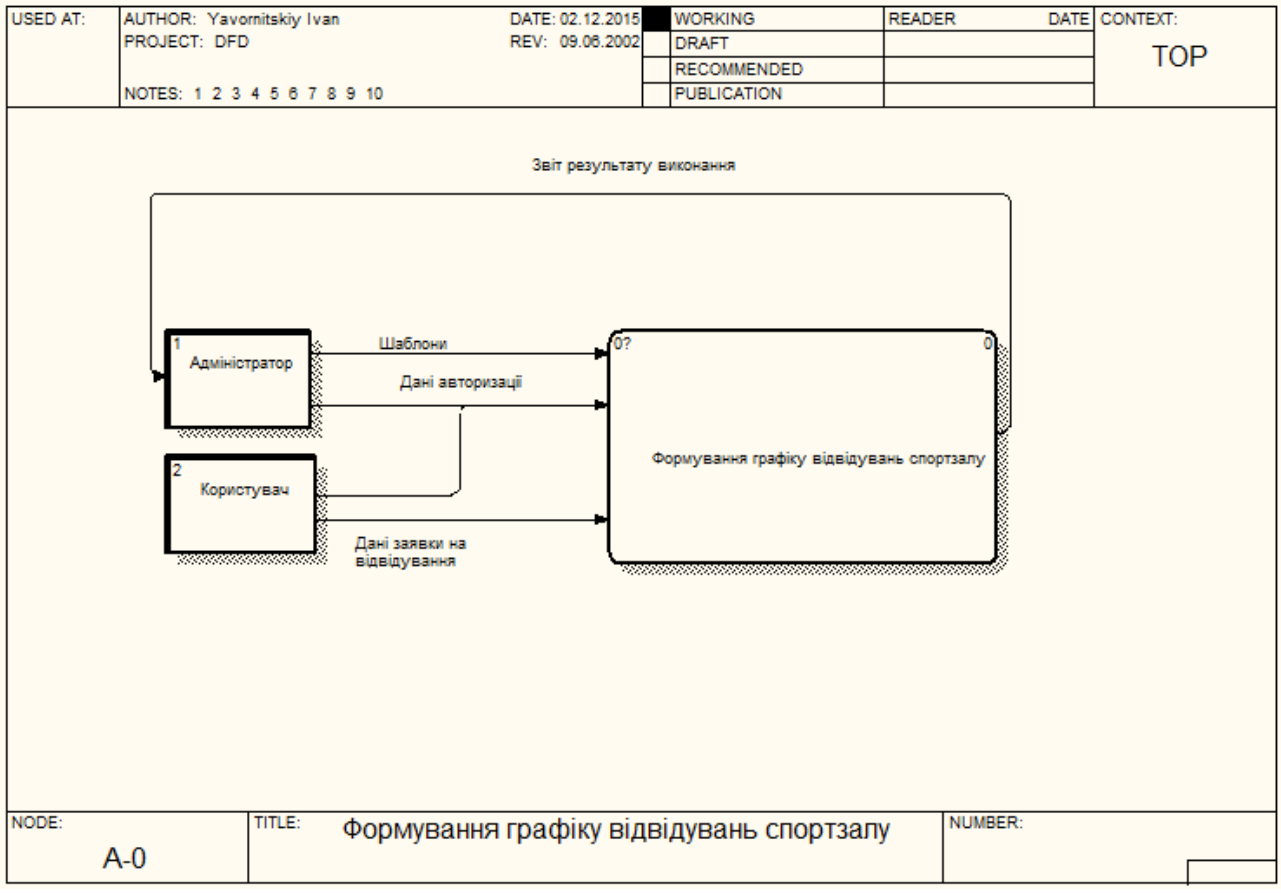


Рис.2.6. Діаграма потоків даних

Діаграма декомпозиції зображена на рисунку 2.7.

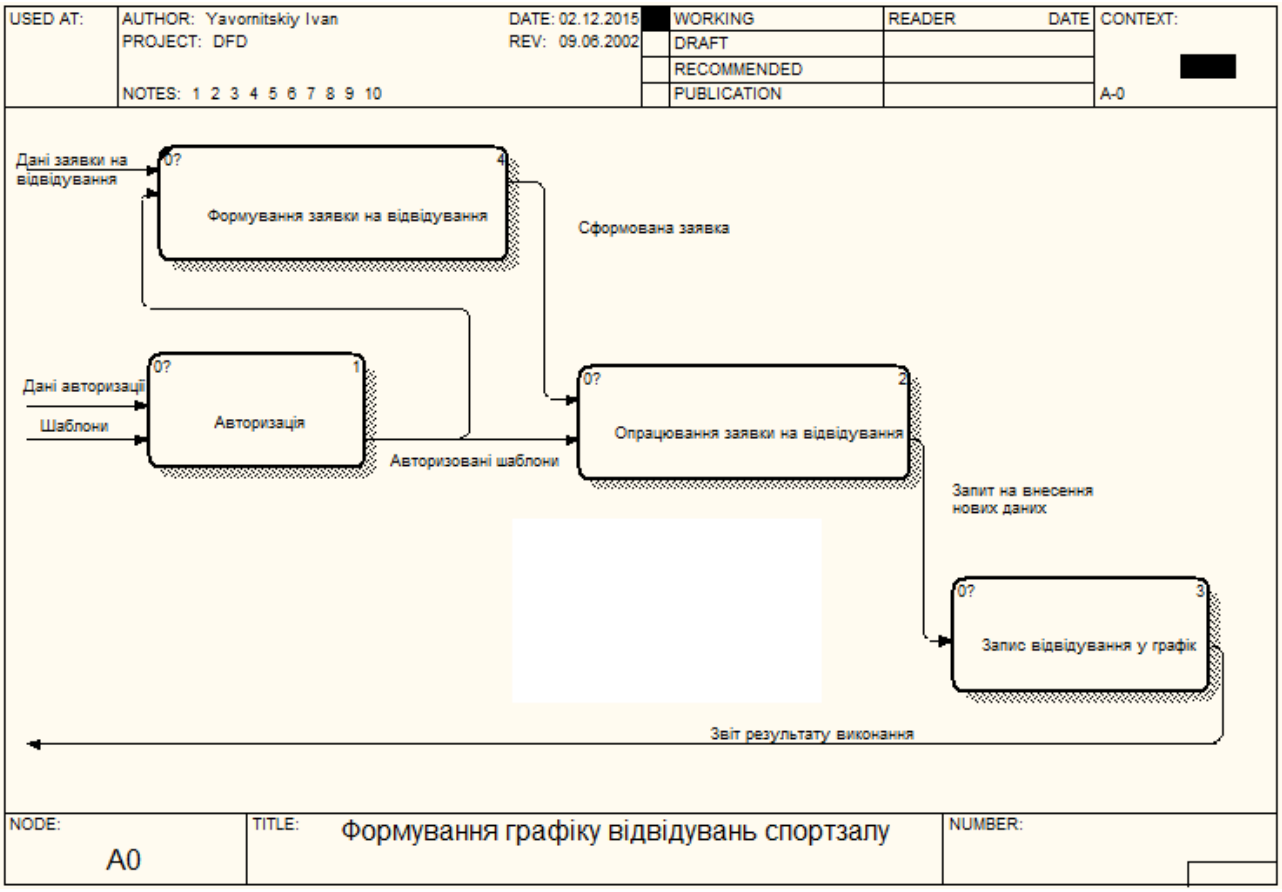


Рис. 2.7. Діаграма декомпозиції

Як показано на рисунку 2.6. робота програмної системи здійснюється наступним чином:

Входи (input), потоки, які поступають у систему і переробляються нею у вихідні величини, на діаграмі зображені ліворуч:

- ✓ Дані авторизації;
- ✓ шаблони;
- ✓ дані заявки на відвідування.

Виходи (output), продукти діяльності системи, тобто результати перетворення вхідних величин тощо, на діаграмі зображені праворуч – звіт результату виконання;

Процес формування графіку відбувається за допомогою адміністратора, який взаємодіє з програмною системою. Адміністратор, переглянувши заявку

на відвідування, вибирає дату початку виконання події відвідування та дату її завершення виконання, щоб визначити межі виконання. Після цього він вводить дані про подію – опис, категорію події. Наступним кроком є внесення самих користувачів системи, а оскільки користувачами є студенти і викладачі, то необхідно розробити ієрархію сутностей починаючи від факультету і закінчуючи студентом та його даними.

Наступним кроком є виявлення основних сутностей та зв’язків між ними. Для цього створено діаграму елементів та зв’язків (Рисунок 2.8).

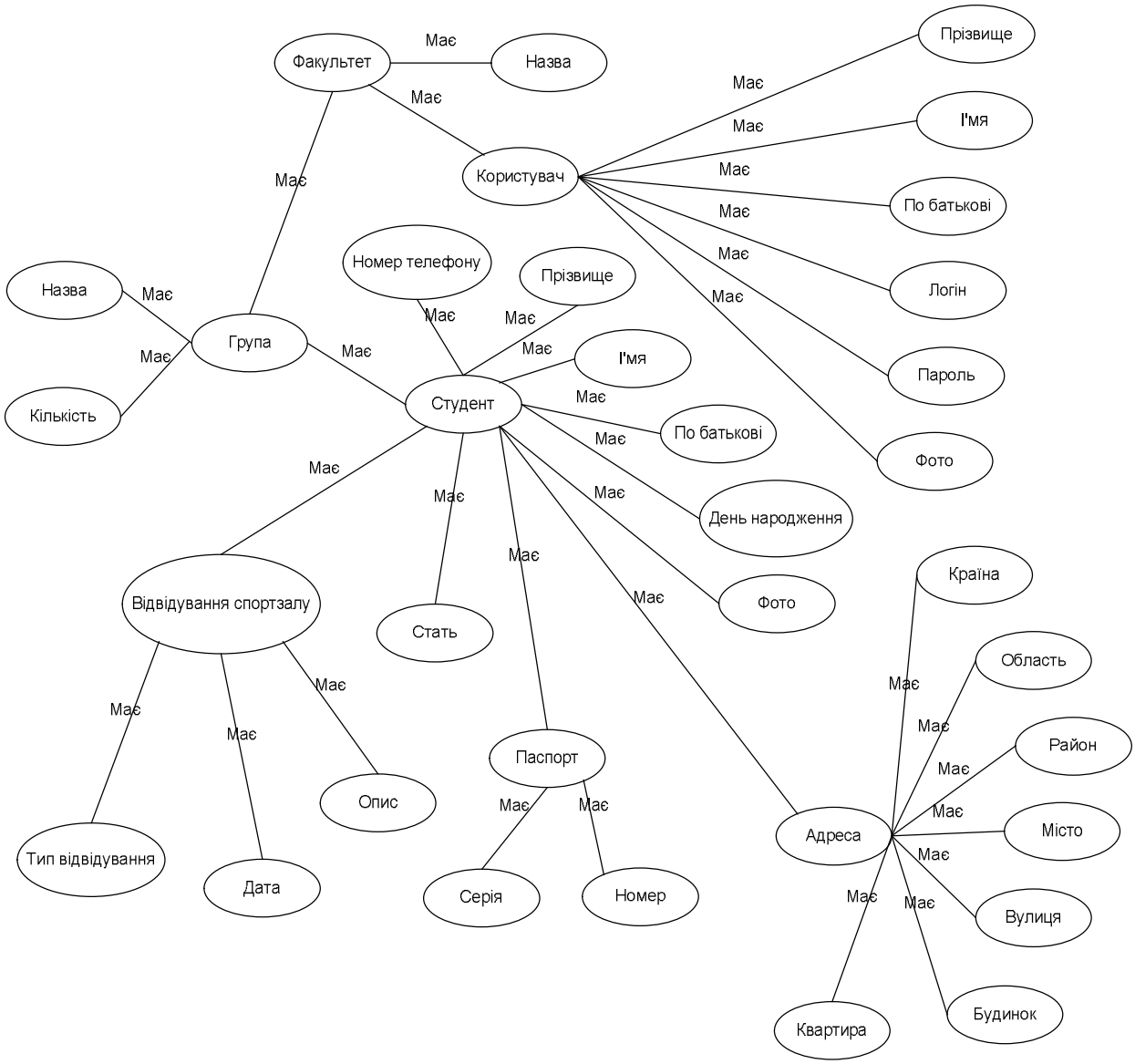


Рис.2.7. Діаграма елементів і зв’язків

Після цього було створено таблицю ідентифікаторів, яку подано у таблиці 2.1.

Таблиця 2.1.

Таблиця ідентифікаторів

Об’єкт	Властивість	Тип	Розмірність	Ідентифікатор
Факультет	Назва	String	30	Faculty Name
Користувач	Прізвіще Ім’я По батькові Логін Пароль Фото	String String String String String Image	50 35 30 20 40 200 kb	User Surname Name Lastname Username Password Photo
Група	Назва Кількість Кількість державників	String Integer Integer	50 5 5	Group Name Places StateOrder
Студент	Ім’я Прізвище По батькові День народження Номер телефону Фото Стать	String String String Date Integer Image String	35 50 50 10 200 kb 10	Student FirstName Name LastName Birthday Phone Photo Sex
Адреса	Країна Область Район Місто Вулиця Квартира	String String String String String Integer	20 20 20 20 20 5	Adress Country Region District City Street House
Паспорт	Серія Номер	String Integer	100 10	Passport Series Number
Подія	Дата Тип відвідування Кількість міст	Date String String	 4 150	Event EventDate TypeEvent EventCount

Завершальним етапом проектування бази даних є створення ER-діаграми (Рисунок 2.5.)

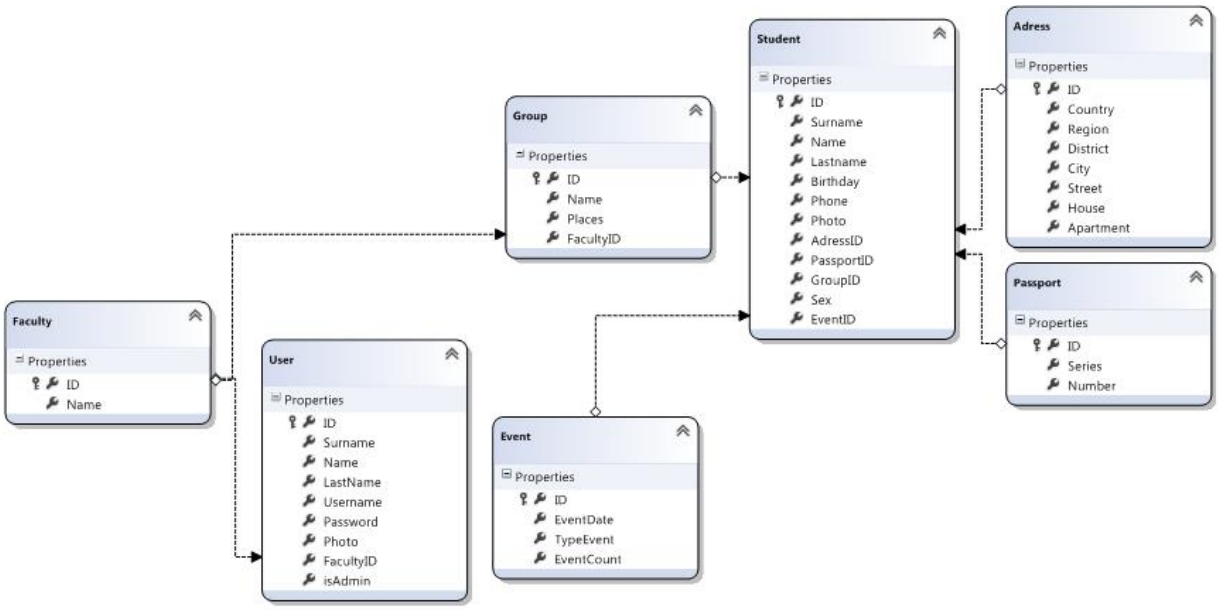


Рис.2.5. ER-діаграма

Проектування структури бази даних дозволяє визначити та повністю описати всі відношення та поля і є завершальною ланкою перед етапом програмної реалізації бази даних.

ВИСНОВКИ ДО ДРУГОГО РОЗДІЛУ:

1. Розроблено архітектуру програмної системи, що дозволить краще зрозуміти функції основних її частин.
2. Створено та описано структурну схему програмної системи, основними компонентами якої є: рівень клієнта, рівень бізнес-логіки та рівень даних.
3. Описано функціональну структуру програмної системи та її основних елементів – модулів обробки даних.
4. Для кращого розуміння процесу функціонування програми побудовано UML-діаграми активності до кожного модуля програмної системи.
5. Визначено основні елементи бази даних та встановлено зв’язки між ними.
6. Спроектовано структуру бази даних.

7. Завершення проектування дозволяє розпочати програмну реалізацію системи.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1. Програмна реалізація проекту

3.1.1. Обґрунтування вибору мови програмування системи.

Платформа ASP.NET представляє собою технологію від компанії Microsoft, призначену для створення різного роду веб-додатків: від невеликих веб-сайтів до великих веб-порталів і веб-сервісів. Вона є складовою частиною платформи Microsoft .NET і розвитком старішої технології Microsoft ASP. На даний момент останньою версією цієї технології є ASP.NET 5.

З одного боку, ASP.NET 5 є продовженням розвитку платформи ASP.NET. Але з іншого боку, це не просто черговий реліз. Вихід ASP.NET 5 фактично означає революцію всієї платформи, її якісна зміна.

ASP.NET 5 тепер повністю є opensource фреймворком і має повноцінну кросс-платформенність. Всі вихідні файли фреймворку доступні на GitHub. Тобто ми можемо запускати веб-додатки не тільки на ОС Windows, але і на Linux і MacOS.

Додаток ASP.NET 5 може працювати з двома виконуваними середовищами: .NET Core і з повною версією фреймворка .NET.

.NET Core представляє собою модульне крос-платформне виконуюче середовище, яка спрощує розгортання програми.

Завдяки модульності всі необхідні компоненти веб-додатку можуть завантажуватися як окремі модулі через пакетний менеджер Nuget. Крім того, на відміну від попередніх версій платформи немає необхідності використовувати бібліотеку System.Web.dll.

ASP.NET 5 – більш оптимізована для використання в хмарі і має вбудовану підтримку впровадження залежностей.

При розгортанні, для веб-додатку, можна використовувати традиційний IIS. Але також можна запускати веб-додаток, використовуючи крос-платформний веб-сервер Kestrel.

ASP.NET 5 включає в себе фреймворк MVC 6, який об'єднує функціональність MVC, Web API і Web Pages. У попередніх версіях платформи дані технології реалізувалися окремо і тому містили багато дублюючої функціональності. Зараз же вони об'єднані в одну програмну модель MVC 6. А Web Forms повністю пішли в минуле.

Крім об'єднання вищезазначених технологій в одну модель в MVC був доданий ряд додаткових функцій.

Однією з таких функцій є тег-хелпери (tag helper), які дозволяють більш органічно поєднувати синтаксис html з кодом C #.

C # відноситься до сім'ї мов з C-подібним синтаксисом, з них його синтаксис найбільш близький до C ++ і Java. Мова має статичну типізацію, підтримує поліморфізм, перевантаження операторів (в тому числі операторів явного і неявного приведення типу), делегати, атрибути, події, властивості, узагальнені типи і методи, ітератори, анонімні функції з підтримкою замикань, LINQ, виключення, коментарі в форматі XML[11].

Перейнявши багато від своїх попередників – мов C ++, Pascal, Smalltalk і, особливо, Java – C #, спираючись на практику їх використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем. Наприклад, C #, на відміну від C ++, не підтримує множинне успадкування класів (між тим допускається множинне спадкування інтерфейсів)[11].

Для створення даної системи використано Microsoft Visual Studio 2013.

Структура системи передбачає дві ролі: користувача та адміністратора. Користувач – це людина, яка перейшовши на головну сторінку сайту може переглядати інформацію і при проходженні авторизації подати заявку на абонемент в спортзал. Адміністратор також повинен пройти авторизацію, але його права, після авторизації, є значно ширші. Він може додавати, видаляти та

змінювати усю інформацію на сайті, а також має доступ до персональних даних клієнтів.

Програмування сайтів на ASP.NET складається з двох етапів: програмування інтерфейсу системи та самого алгоритму продукту. Інтерфейс складається з файлів, які містять розмітку сторінки та інші елементи, а алгоритм системи складається з коду, в якому програмується взаємодія з сайтом.

Реалізація авторизації адміністратора на сайті з визначеним логіном та паролем представлена наступним кодом

```
void Login_Click(object sender, EventArgs e)
  {
if ((UserLogin.Text == «ADMIN») &&
          (UserPass.Text == «12345»))
     {
        FormsAuthentication.RedirectFromLoginPage
          (UserLogin.Text, Persist.Checked);
     }
else
     {
        Msg.Text = «Не вірно введені дані. Будь ласка спробуйте ще раз.»;
     }
```

Після авторизації на веб-сайті адміністратор зможе виконувати такі операції:

- додавання нового відвідування;
- редагування графіку відвідувань;
- додавання студентських даних;
- перегляд заявок;
- редагування студентських даних.

У великих компаніях передбачено декілька адміністраторів і великої кількості користувачів, тому дана система передбачає таку ситуацію і перед внесенням змін даних на сайті відбувається пошук і витягнення даних із бази даних про даного адміністратора чи користувача. Дана операція призначена для моніторингу внесених змін на веб-сайті та передбачення персональної відповідальності за них представлено

```
using (var connection = new SqlConnection(strcon))
   {       connection.Open();
          var cmdIn = new SqlCommand(«SELECT userId FROM users WHERE userName=@userName», connection);
          cmdIn.Parameters.AddWithValue(«@userName», Session[«User»]);
            var rdr = cmdIn.ExecuteReader();
                rdr.Read();
                var userId = (int)rdr[«userId»];
                connection.Close();
```

Спочатку відбувається підключення до бази даних, потім відбувається пошук інформації про даного адміністратора та зчитування його ідентифікаційного номера.

3.1.2. Організація інтерфейсу з користувачем

Завантаження головної сторінки сайту (Рисунок 3.1.) для адміністратора і звичайного користувача є однаковою, вони отримують доступ до загальної інформації та до можливості перегляду розкладу занять (Рисунок 3.2.), але після проходження авторизації переходять відповідно на свої облікові записи. Якщо користувач не зареєстрований він натискає кнопку «Реєстрація», та переходить у вікно реєстрації (Рисунок 3.3.)

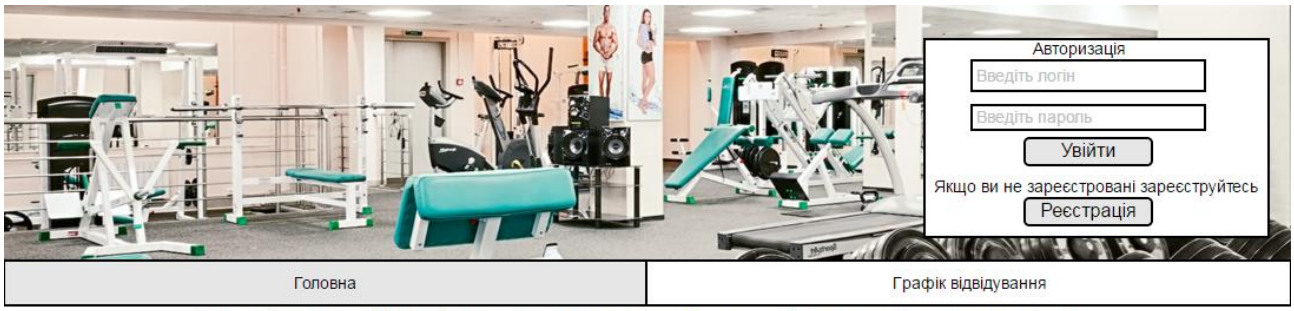
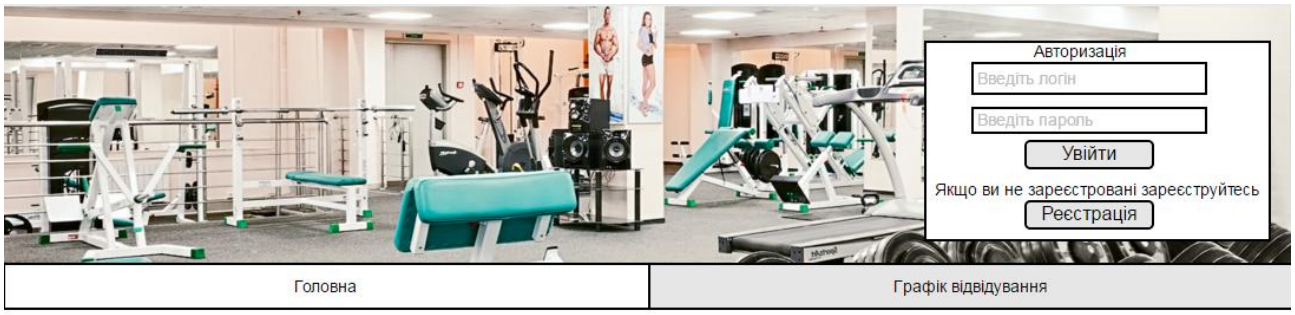


Рис.3.1. Головна сторінка сайту



Розклад занять в спортзалі станом на 8 червня 2016року

▼ Час	▼ Тип відвідування	▼ Вільних місць
08.00. - 10.00.	Футболу	10
10.00. - 12.00.	Великого тенісу	15
12.00. - 14.00	Волейболу	5
14.00. - 16.00.	Баскетболу	1
16.00. - 18.00.		

червень 2016 р.

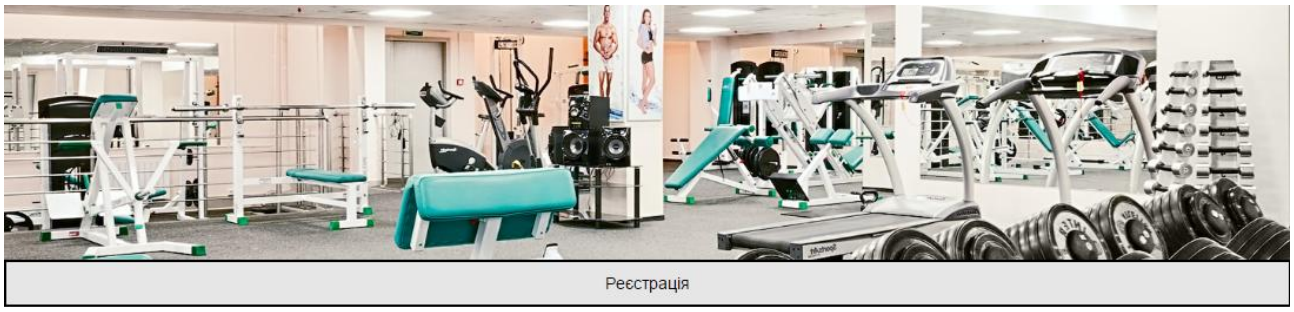
Пн	Вт	Ср	Чт	Пт	Сб	Нд
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	1	2	3
4	5	6	7	8	9	10

Сьогодні: 08.06.2016

Розклад занять в тренажерному залі станом на 8 червня 2016року

▼ Час	▼ Вільних місць
08.00. - 10.00.	6
10.00. - 12.00.	12
12.00. - 14.00	11
14.00. - 16.00.	7
16.00. - 18.00.	2

Рис.3.2. Сторінка перегляду розкладу занять



Введіть логін

Введіть пароль

Введіть пароль ще раз

Зареєструвати

Рис.3.3. Сторінка реєстрації

При авторизації адміністратор переходить в свій обліковий запис, сторінка якого має дві вкладки «Внесення студентських даних» та «Заявки». На

вкладці «Внесення студентських даних» (Рисунок 3.4.) адміністратор вводить, редагує чи видаляє дані студента.

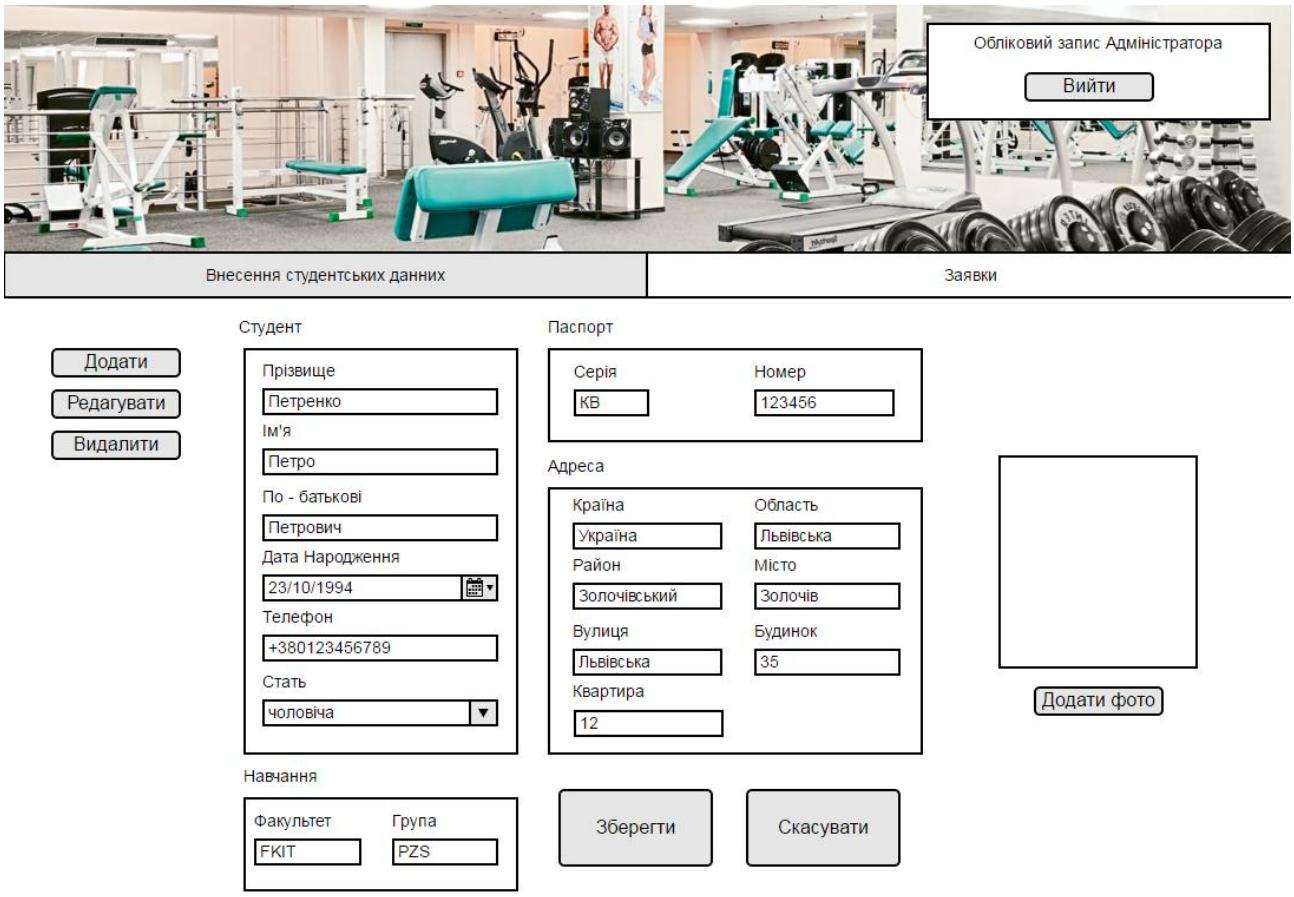


Рис.3.4. Сторінка облікового запису адміністратора, вкладка «Внесення студентських даних»

Сторінка подачі заявки на сторінці користувача зображена на рисунку 3.5.

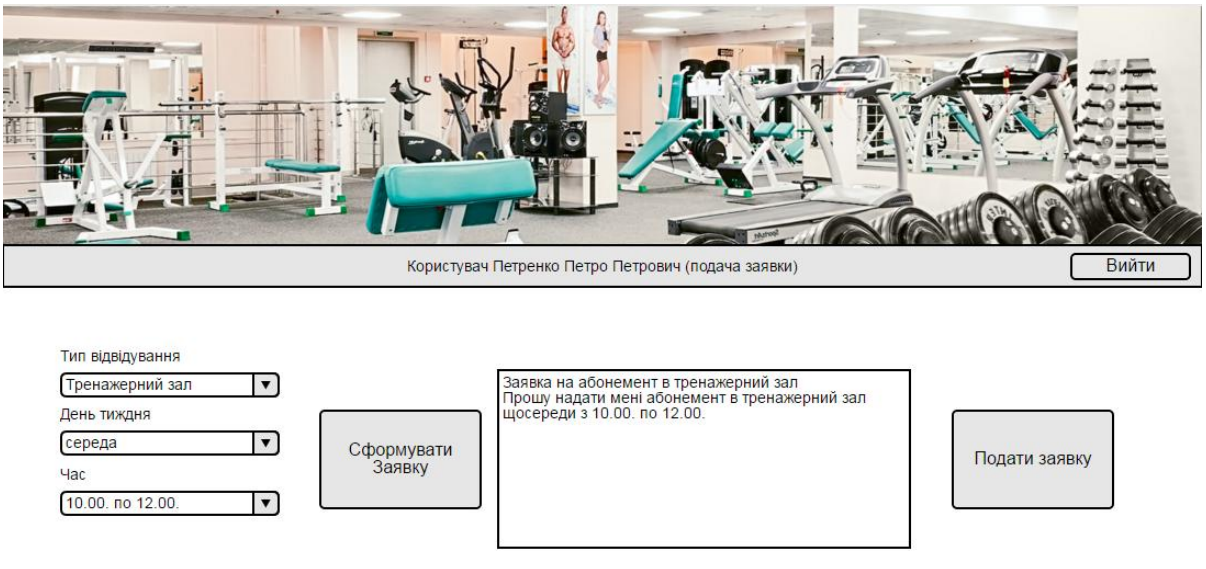


Рис.3.5. Сторінка подачі заявки (обліковий запис користувача)

На вкладці «Заявки» (Рисунок 3.6.) адміністратор переглядає подані заявки, звіряє їх з розкладом, який він може переглядати за допомогою пошуку по категоріях. Підтверджує їх якщо вони відповідають розкладу, або редагує їх таким чином, щоб вони відповідали графіку і також підтверджує їх.

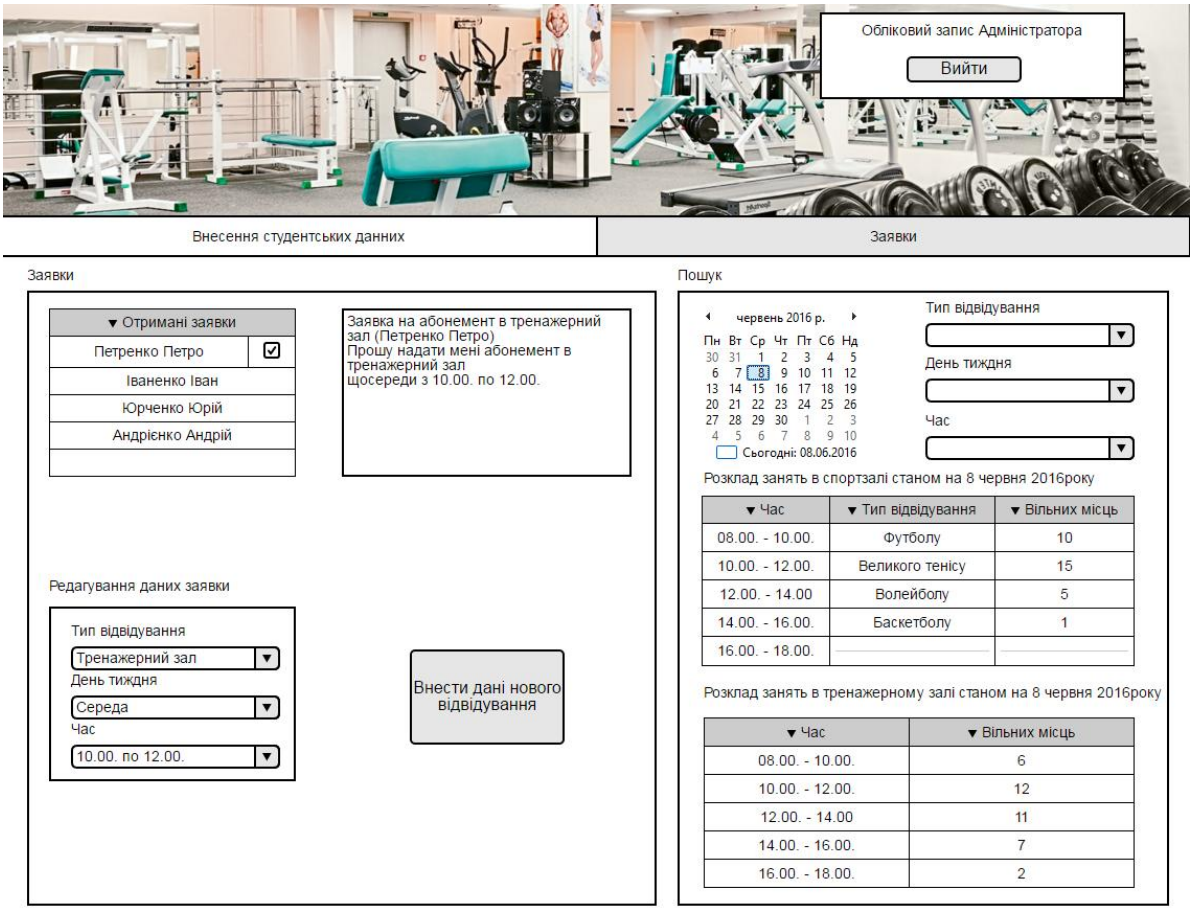


Рис.3.6. Сторінка облікового запису адміністратора, вкладка «Заявки»

3.2. Програмна реалізація бази даних

Для створення бази даних було обрано Microsoft SQL Server Management Studio 2016. Даний програмний продукт використовується для роботи в СУБД Microsoft SQL Server 2016 і здійснює управління, налаштування та адміністрування компонентів бази даних. Поєднується з середовищем Microsoft Visual Studio та забезпечує підтримку і сумісність розробки БД з ПЗ, оскільки розроблені з високим ступенем інтеграції.

SQL Server Management Studio – менеджер для управління об’єктами в SQL сервері. Він має зручний користувацький інтерфейс та можливість написання скриптів для адміністрування БД.

Веб-орієнтована система планування графіку спортивного залу передбачає створення однієї бази даних. Дана база передбачена для збереження графіку відвідувань спортзалу, даних користувачів та даних про студентів, які є головними відвідувачами спортзалу. Отже, в базі даних організації графіку відвідувань повинні бути реалізовані таблиці Faculty, User, Specialization, Profession, Student, Adress, Pasport, Event, Gym.

Проте, для того щоб створити першу таблицю, необхідно встановити та налаштувати Microsoft SQL Server та SQL Server Management Studio. Після чого запустивши SQL Server Management Studio, під’єднатися до сервера, що показано на рисунку 3.7.

Рис.3.7. Підключення до локального SQL Server

У головному вікні, використовуючи Object Exporer, необхідно натиснути праву клавішу миші на пункті Databases та обрати варіант «New Database». Послідовність зображена на рисунку 3.8. Це дозволяє створювати нові бази даних і надалі працювати з ними. У даному середовищі також можливий імпорт/експорт та відновлення баз даних.

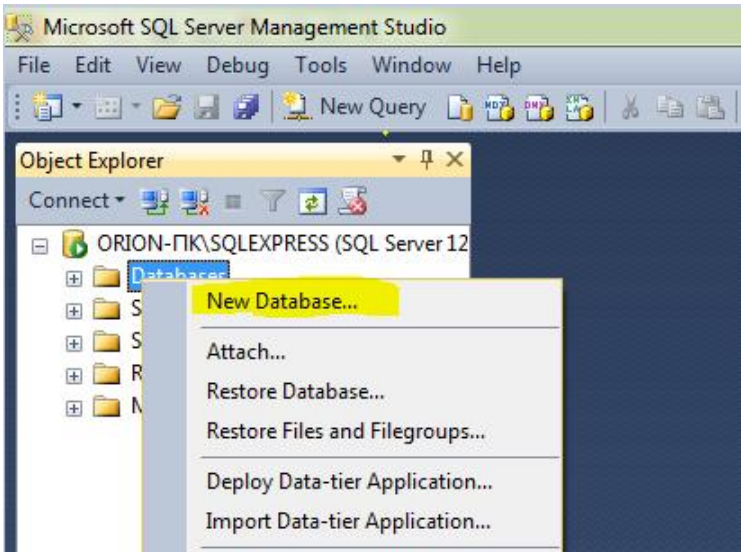


Рис.3.8. Створення нової бази даних

Результатом даної послідовності буде виклик вікна конфігурації нової бази даних що показано на рисунку 3.9. У вікні необхідно ввести назву та підтвердити виконання.

В результаті буде отримано та виконано такий SQL-скрипт:

```
USE [master]
GO
CREATE DATABASE [Gym]
CONTAINMENT = NONE
ON PRIMARY
GO
USE [Gym]
GO
```

Виконання цього скрипта призведе до створення нової бази даних Jym та дасть змогу використовувати наступні скрипти уже в її маштабах.

Наступним кроком буде створення таблиці Faculty. Для цього необхідно запустити редактор скриптів, виконавши таку послідовність: натиснути правою кнопкою миші на базі даних і обрати пункт «New Query». Дана послідовність зображена на рисунку 3.10.

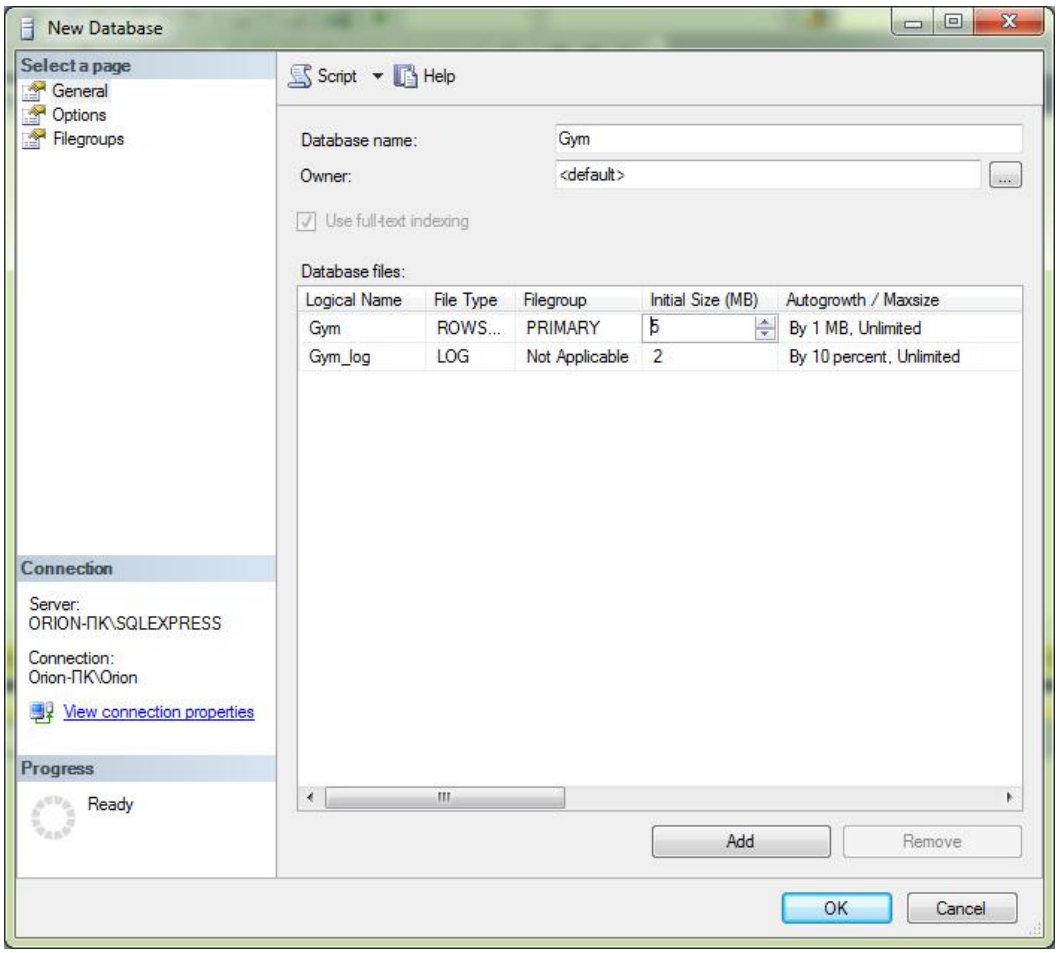


Рис.3.9. Вікно налаштування нового об’єкту бази даних

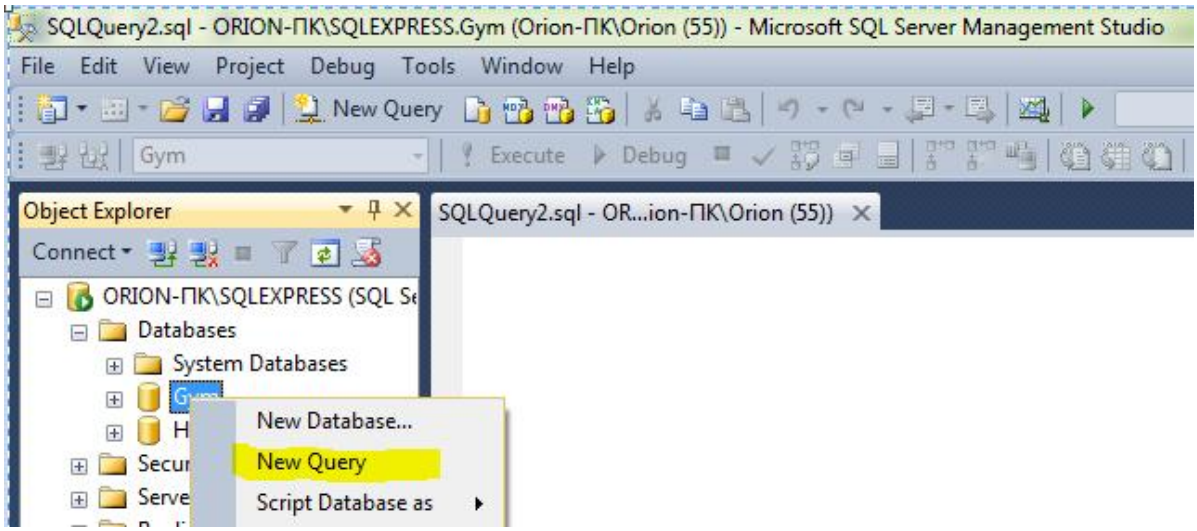


Рис.3.10 Відкриття редактора SQL-скриптів

На рисунку 3.11 зображено вікно редактора скриптів із результатом відлагодженого скрипта.

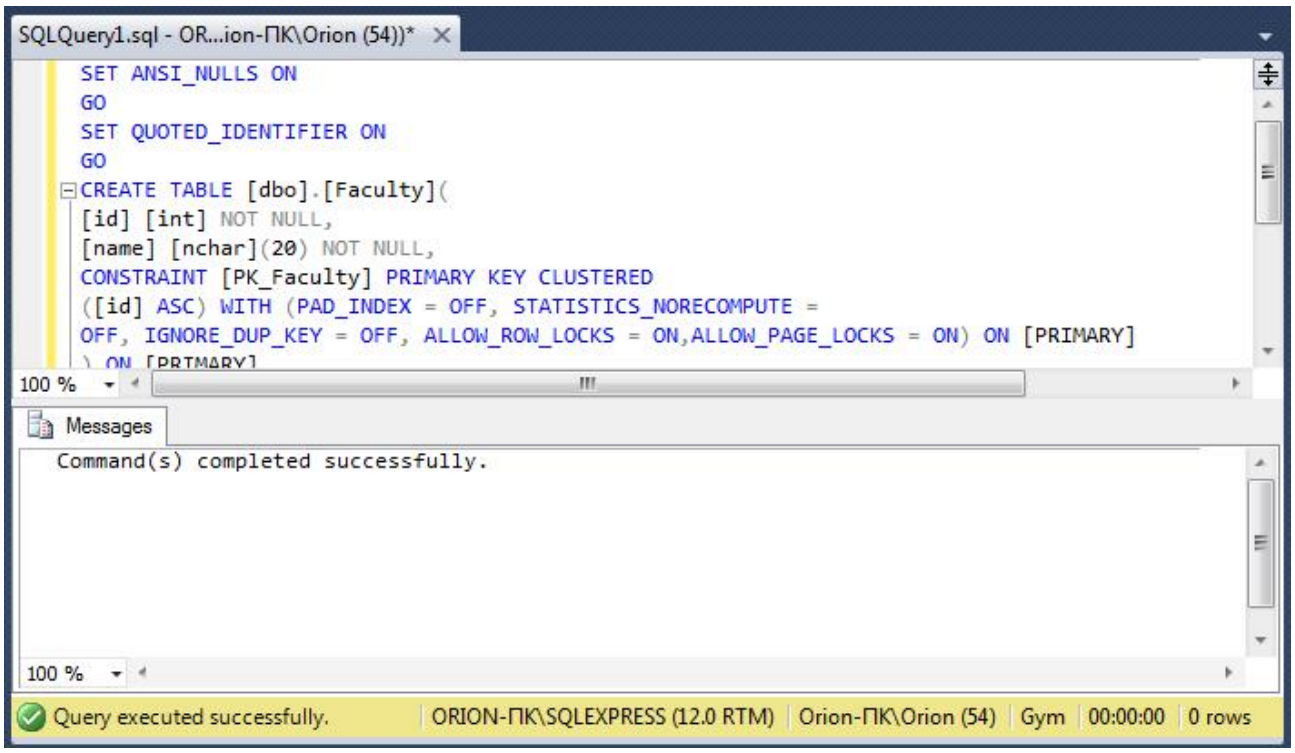


Рис.3.11. Вікно редактора скриптів

Код відлагодженого SQL-скрипта із рисунка 1.14 подано нижче:

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Faculty](
[id] [int] NOT NULL,
[name] [nchar](20) NOT NULL,
CONSTRAINT [PK_Faculty] PRIMARY KEY CLUSTERED
([id] ASC) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE =
OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
```

Виконання цього скрипта призвело до створення таблиці Faculty з первинним ключем id та з полями і типами цієї сутності згідно з таблицею ідентифікаторів та ERD.

Наступним буде створення таблиці User. Буде використано такий SQL-запит:

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[User](
[id] [int] NOT NULL,
[Surname] [nchar](50) NOT NULL,
```

```
[Name] [nchar](35) NOT NULL,
[Lastname] [nchar](30) NOT NULL,
[Username] [nchar](20) NOT NULL,
[Password] [nchar](40) NOT NULL,
[Photo] [varbinary](max) NOT NULL,
[Faculty_id] [int] NOT NULL,
CONSTRAINT [PK_User] PRIMARY KEY CLUSTERED
([id] ASC)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
ON [PRIMARY]
) ON [PRIMARY]
GO
```

Виконання цього скрипта призведе до створення таблиці User з первинним ключем id та вторинним ключем Faculty_id та з полями і типами цієї сутності згідно з таблицею ідентифікаторів та ERD.

Бізнес логіку забезпечує створення зв’язків між таблицями за допомогою первинних та вторинних ключів та використання полів індексів – полів по яких сутність буде визначатися при пошуку.

Створення інших таблиць подано у додатку А.

ВИСНОВКИ ДО ТРЕТЬОГО РОЗДІЛУ:

1. Обґрунтовано вибір мови програмування.
2. Проведено детальний опис розробки системи з фрагментами коду.
3. Проведено короткий аналіз бази даних SQL Server
4. Створено таблиці в базі даних та наведено приклади їх виконання.
5. Створено початкову версію системи, яку необхідно піддати тестуванню.

РОЗДІЛ 4 ТЕСТУВАННЯ ТА ДОСЛІДНА ЕКСПЛУАТАЦІЯ

4.1. Тестування

Для перевірки продукту на відповідність поставленим вимогам та відсутність помилок було проведено тестування. Взявши до уваги специфіку продукту було складено та проведено функціональне тестування та GUI тестування.

Функціональне тестування проводить перевірку програмного продукту відповідно до функціональних вимог. Тести проводяться над усіма функціями програмної системи та охоплюють всі розроблені функції. Для проведення функціонального тестування було розроблено тестові випадку у вигляді таблиці, що містить таку інформацію:

- № (номер тестового випадку);
- Description (опис об’єкту тестування);
- Steps to reproduce (кроки відтворення тестового випадку);
- Expected result (очікуваний результат);
- Actual result (фактичний результат);
- Passed/Failed (чи пройдений тест, чи ні.);
- Environment (середовище, в якому здійснювалося тестування).

Було проведено 31 тест, з них усі виявились успішними. Звідси випливає, що усі функції програми відповідають функціональним вимогам.

Таблиця 4.1.

Функціональні тестові випадки

Варіанти використання	**Тестові випадки**	**Тестові дані**
Авторизація	3	9
Перегляд графіку відвідувань	5	15
Внесення даних нового відвідування	7	20

Продовження таблиці 4.1.

Формування заявки	7	11
Внесення студентських даних	3	7
Перегляд по категоріям	2	5
Перегляд заявок	4	20
Загалом	**31**	**87**

Таблиця 4.2

Специфікація тестів для функціонального тестування

№ п/п	**Тестові випадки**	**Тестові дані**
1	Перевірка авторизації	9
2	Перевірка графіку відвідувань	15

3	Перевірка внесення студентських даних	7
4	Перевірка внесення даних нового відвідування	20
5	Перевірка формування заявки	11
6	Перевірка редагування студентських даних	8
7	Перевірка редагування графіку відвідувань	5
8	Перевірка перегляду по категоріям	5
9	Перевірка перегляду заявок	20
10	Перевірка з’єднання з БД	7
	Загалом	107

GUI тестування

GUI тестування призначене для тестування графічного інтерфейсу користувача. Він перевіряє відповідність роботи інтерфейсу користувача поставленим вимогам. Дії потенційного користувача відтворюються для перевірки очікуваного результату з реальним. Для цього було складено чек-ліст та проведено ручне тестування інтерфейсу.

Було проведено 10 тестів, з них усі виявились успішними. Отже, інтерфейс користувача працює добре та відповідає заявленим вимогам.

Таблиця 4.3

GUI тестові випадки

Варіанти використання	**Тестові випадки**	**Тестові дані**
Авторизація	2	8
Перегляд графіку відвідувань	4	16
Перегляд студентських даних	2	4
Перегляд заявок	4	3
Загалом	10	32

Таблиця 4.4

Специфікація тестів для GUI тестування

№ п/п	**Тестові випадки**	**Тестові дані**
1	Перевірити-> Кнопку “Авторизація”	2
2	Перевірити-> Кнопку “Календар”	4
3	Перевірити-> Кнопку “Користувачі”	3
4	Перевірити-> Кнопку “Факультети”	1
5	Перевірити-> Кнопку “Студенти”	10
6	Перевірити-> Кнопку “Графік відвідування”	4
7	Перевірити-> Кнопку “Додати”	4
8	Перевірити-> Кнопку “Зберегти”	4
9	Перевірити-> Кнопку “Додати фото”	2
10	Перевірити-> Кнопку “Завершити редагування”	4

4.2. Розгортання програмного продукту

Для роботи на сайті користувачеві потрібно:

1. Підключитися до мережі Інтернет.
2. Запустити браузер.
3. Ввести URL-адресу сайту.
4. Дочекатися відкриття сайту.

Коректна робота системи буде забезпечена лише при стабільному підключенні до Інтернет мережі.

Для роботи програмної системи необхідне таке апаратне та програмне забезпечення:

Системні вимоги до сервера:

1) Вимоги до апаратного забезпечення:

• Оперативна пам’ять - 8Гб;

• Об’єм дискового простору - 320Гб.

2) Вимоги до програмного забезпечення:

• Операційна система - OS Windows 7 і вище;

• .NET Framework 4.0;

• MS SQL Server 2016.

Системні вимоги до клієнта:

Вимоги до апаратного забезпечення:

• Оперативна пам’ять – 256 мб.

• Жорсткий диск 100 Гб.

Вимоги до програмного забезпечення:

• Операційна система - OS Windows 7 і вище;

• .NET Framework 4.0.

4.3. Інструкція користувача

4.3.1. Компоненти ПЗ

4.3.2.

Програмна система розроблена на мові програмування C# у середовищі Microsoft Visual Studio 2013 на платформі ASP.NET і працює у веб переглядачах.

Для функціонування системи планування відвідування спортивного залу необхідне встановлення на СУБД сервера бази даних , файл якої постачаються разом з програмною системою.

4.3.2. Встановлення ПЗ

Для роботи програмної системи необхідно встановити Microsoft SQL Server та Microsoft .NET Framework 4.

На стороні сервера, використовуючи СУБД Microsoft SQL Server 2016, розгорнути базу даних: Jym. Для цього використати середовище Microsoft SQL Server Management Studio 2016, в якому після підключення до СУБД необхідно обрати імпорт бази даних в панелі інструментів. Джерелом даних слід вказати файли баз даних, які надаються разом з програмною системою.

4.3.3. Налаштування

Відповідно до створених баз даних, користувачів та налаштувань з’єднання, необхідно записати стрічку підключення до бази даних в файл конфігурації SQLConnection.config в стрічки параметри «Gym».

4.3.4. Базові функції ПЗ

На сторінці користувача:

- перегляд графіку на головній сторінці;
- надсилання заявок на сторінці користувача;

На сторінці адміністратора:

- внесення студентських даних;
- перегляд заявок;
- внесення нових відвідувань.

4.3.4. Аналіз помилок

При помилках «Server connection timed out» чи «Cannot connect to server» необхідно перевірити з’єднання клієнта та сервера з мережею та активність СУБД сервера.

ВИСНОВКИ ДО ЧЕТВРТОГО РОЗДІЛУ:

1. Проведено функціональне та GUI тестування, під час яких суттєвих помилок не було виявлено.
2. Визначено особливості даної системи для нормального функціонування.
3. Створено інструкції для користувача.

ВИСНОВКИ

Результатом дипломної роботи є створений сайт планування відвідувань спортивного залу. Даний ресурс буле доступний усім користувачам персональних комп’ютерів чи ноутбуків з активним підключенням до мережі Інтернет.

Під час виконання було описано об’єкт управління, розглянуто три системи-аналоги, в яких виділено недоліки та функціональні можливості. На основі аналізу існуючих систем створено специфікацію вимог до системи.

Проектування складалося з вибору архітектури системи, вибору та проектування баз даних.

Програмна реалізація була проведена у два етапи: програмна реалізація системи мовою програмування C# за допомогою технології ASP.NET та фізична реалізація бази даних SQL Server 2016.

Для тестування системи було використано ручне функціональне та GUI тестування, яке було успішно пройдено.

Проведено детальний опис вимог до програмного забезпечення та характеристик сервера на якому система буде нормально функціонувати.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. С.Ю. Золотов. Проектирование информационных систем: Учебное методическое пособие. – Томск: ТМЦДО, 2006 – 34.

2. Вендров А.М. Проектирование программного обеспечения экономических информационных систем: Учебник. – 2-е изд., перераб. и доп.– М.: Финансы и статистика, 2006. – 544 с: ил.

3. Дивак М. П., Шпінталь М.Я., Козак О.Л., Струбицька І.П., Спільчук В.М., Піговський Ю.Р. Методичні рекомендації до виконання дипломної роботи освітньо кваліфікаційного рівня «бакалавр» студентам усіх форм навчання для напряму підготовки 6.050103 – «Програмна інженерія» // Тернопіль : ФОП Шпак П.П. - 2014. - 54 с.

4. Основы системного анализа и проектирования АСУ./ под. ред.А.А.Павлова. -К: Вища школа, 1991.

5. Офіційний сайт ТНЕУ [Електронний ресурс]. – Режим доступу http://www.tneu.edu.ua/

6. Побудова діаграми варіантів використання [Електронний ресурс],– Режим доступу: http://moodle.ipo.kpi.ua/

7. Перегудов Ф.И, Тарасенко Ф.П. Введение в системный анализ. М: Высшая школа, 1992.

8. Системный анализ и структуры управления./ под ред. В.Г.Щорина -М.: Знание, 1975.

9. Шарапов О.Д., Терехов Л.М., Сіднєв С.П. Системний аналіз. К.: Вища школа, 1983.

10. Hawryszkiewych I.T. Introduction to system analysis and design. New York, 1992. 379 p.

11. Бібліотека MSDN - цінне джерело інформації для розробників, які використовують засоби, продукти, технології та служби корпорації Майкрософт [Електронний ресурс]. — Режим доступу:

http://msdn.microsoft.com/uk-ua/library/ms123401, для доступу до інформаційних ресурсів авторизація непотрібна. — Назва з екрану. Библиотека MSDN.

12. ADO.net on-line tuturial
13. MS JDBC SQL Server driver documentation
14. SQL Native Client documentation on MSDN
15. Transact-SQL Reference on MSDN

ДОДАТКИ

ДОДАТОК А

DDL-код створення таблиць у базі даних

```
USE [master]
GO
CREATE DATABASE [Gym]
CONTAINMENT = NONE
ON PRIMARY
GO
USE [Gym]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Faculty](
[id] [int] NOT NULL,
[name] [nchar](20) NOT NULL,
CONSTRAINT [PK_Faculty] PRIMARY KEY CLUSTERED
([id] ASC) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE =
OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,ALLOW_PAGE_LOCKS =
ON) ON [PRIMARY]
) ON [PRIMARY]
GO
CREATE TABLE [dbo].[User](
[id] [int] NOT NULL,
[Surname] [nchar](50) NOT NULL,
[Name] [nchar](35) NOT NULL,
[Lastname] [nchar](30) NOT NULL,
[Username] [nchar](20) NOT NULL,
[Password] [nchar](40) NOT NULL,
[Photo] [varbinary](max) NOT NULL,
[Faculty_id] [int] NOT NULL,
CONSTRAINT [PK_User] PRIMARY KEY CLUSTERED
([id] ASC)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
CREATE TABLE [dbo].[Group](
[id] [int] NOT NULL,
[Name] [nchar](50) NOT NULL,
[Places] [int](5) NOT NULL,
[Faculty_id] [int] NOT NULL,
CONSTRAINT [PK_ Group] PRIMARY KEY CLUSTERED
([id] ASC)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
ON [PRIMARY]
) ON [PRIMARY]
GO
CREATE TABLE [dbo].[ Student](
[id] [int] NOT NULL,
[FirstName] [nchar](35) NOT NULL,
[Name] [nchar](50) NOT NULL,
[Lastname] [nchar](50) NOT NULL,
[Birthday] [Date] NOT NULL,
[Phone] [int](10) NOT NULL,
[Photo] [varbinary](max) NOT NULL,
[Sex] [nchar](10) NOT NULL,
[Group _id] [int] NOT NULL,
CONSTRAINT [PK_Student] PRIMARY KEY CLUSTERED
([id] ASC)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
ON [PRIMARY]
) ON [PRIMARY]
GO
CREATE TABLE [dbo].[Adress](
[id] [int] NOT NULL,
[Country] [nchar](20) NOT NULL,
[Region] [nchar](20) NOT NULL,
[District] [nchar](20) NOT NULL,
[City] [nchar](20) NOT NULL,
[Street] [nchar](20) NOT NULL,
[House] [int](5) NOT NULL,
[Student _id] [int] NOT NULL,
```

```
CONSTRAINT [PK_Adress] PRIMARY KEY CLUSTERED
([id] ASC)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
ON [PRIMARY]
) ON [PRIMARY]
GO
CREATE TABLE [dbo].[Passport](
[id] [int] NOT NULL,
[Series] [nchar](100) NOT NULL,
[Number] [int](10) NOT NULL,
[Student _id] [int] NOT NULL,
CONSTRAINT [PK_ Passport] PRIMARY KEY CLUSTERED
([id] ASC)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
ON [PRIMARY]
) ON [PRIMARY]
GO
CREATE TABLE [dbo].[ Event](
[id] [int] NOT NULL,
[EventDate] [DATE] NOT NULL,
[ExentType] [nchar](40) NOT NULL,
[Description] [nchar](150) NOT NULL,
[Student _id] [int] NOT NULL,
CONSTRAINT [PK_ Event] PRIMARY KEY CLUSTERED
([id] ASC)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
ON [PRIMARY]
) ON [PRIMARY]
GO
```

ДОДАТОК Б

Лістинг головних модулів системи

Лістинг головної сторінки користувача:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="UserMain.aspx.cs"
Inherits="JymYavornitskiy.UserMain1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>

<asp:Calendar ID="Calendar1" runat="server" BackColor="White" BorderColor="White"
BorderWidth="1px" Font-Names="Verdana" Font-Size="9pt" ForeColor="Black" Height="235px"
NextPrevFormat="FullMonth" Width="451px" OnSelectionChanged="Calendar1_SelectionChanged">
<DayHeaderStyle Font-Bold="True" Font-Size="8pt" />
<NextPrevStyle Font-Bold="True" Font-Size="8pt" ForeColor="#333333" VerticalAlign="Bottom" />
<OtherMonthDayStyle ForeColor="#999999" />
<SelectedDayStyle BackColor="#333399" ForeColor="White" />
<TitleStyle BackColor="White" BorderColor="Black" BorderWidth="4px" Font-Bold="True" Font-
Size="12pt" ForeColor="#333399" />
<TodayDayStyle BackColor="#CCCCCC" />
</asp:Calendar>
<br />
<asp:Label ID="Label1" runat="server" Text="Дата початку (Обрати на календарі)"></asp:Label>
<br />
<asp:TextBox ID="TextBox1" runat="server" Enabled="False"></asp:TextBox>
 <asp:TextBox ID="TextBox4" runat="server"></asp:TextBox>
<br />
<asp:Label ID="Label2" runat="server" Text="Дата кінця (формат &quot;00:00:00&quot;)"></asp:Label>
<br />
<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
```

```
<br />
<asp:Label ID="Label4" runat="server" Text="Тип події"></asp:Label>
<br />
<asp:DropDownList ID="DropDownList1" runat="server">
<asp:ListItem>Тренажерний зал</asp:ListItem>
<asp:ListItem>Секція</asp:ListItem>
<asp:ListItem>Пара для групи</asp:ListItem>
</asp:DropDownList>
<br />
<asp:Label ID="Label5" runat="server" Text="Опис події"></asp:Label>
<br />
<asp:TextBox ID="TextBox6" runat="server"></asp:TextBox>
<br />
<asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Зареєструвати" />
<br />
<asp:TextBox ID="TextBox3" runat="server" AutoPostBack="True" Visible="False"></asp:TextBox>
<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$
ConnectionStrings:GymEndConnectionString %>" SelectCommand="SELECT [Id], [Date], [TimeEnd],
[TypeOfbook], [Description] FROM [ToBook] WHERE ([UserID] = @UserID)"
DeleteCommand="DELETE FROM [ToBook] WHERE [Id] = @Id" InsertCommand="INSERT INTO
[ToBook] ([Date], [TimeEnd], [TypeOfbook], [Description]) VALUES (@Date, @TimeEnd,
@TypeOfbook, @Description)" UpdateCommand="UPDATE [ToBook] SET [Date] = @Date, [TimeEnd] =
@TimeEnd, [TypeOfbook] = @TypeOfbook, [Description] = @Description WHERE [Id] = @Id">
<DeleteParameters>
<asp:Parameter Name="Id" Type="Int32" />
</DeleteParameters>
<InsertParameters>
<asp:Parameter Name="Date" Type="DateTime" />
<asp:Parameter DbType="Time" Name="TimeEnd" />
<asp:Parameter Name="TypeOfbook" Type="String" />
<asp:Parameter Name="Description" Type="String" />
</InsertParameters>
<SelectParameters>
<asp:ControlParameter ControlID="TextBox3" Name="UserID" PropertyName="Text" Type="Int32" />
</SelectParameters>
<UpdateParameters>
<asp:Parameter Name="Date" Type="DateTime" />
<asp:Parameter DbType="Time" Name="TimeEnd" />
```

```
<asp:Parameter Name="TypeOfbook" Type="String" />
<asp:Parameter Name="Description" Type="String" />
<asp:Parameter Name="Id" Type="Int32" />
</UpdateParameters>
</asp:SqlDataSource>
<br />
<asp:Label ID="Label3" runat="server" Text="Ваші броні"></asp:Label>
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
DataSourceID="SqlDataSource1" Width="233px" DataKeyNames="Id" BackColor="White"
BorderColor="#CCCCCC" BorderStyle="None" BorderWidth="1px" CellPadding="3">
<Columns>
<asp:BoundField DataField="Id" HeaderText="Id" SortExpression="Id" InsertVisible="False"
ReadOnly="True" />
<asp:BoundField DataField="Date" HeaderText="Date"
SortExpression="Date" />
<asp:BoundField DataField="TimeEnd" HeaderText="TimeEnd" SortExpression="TimeEnd" />
<asp:BoundField DataField="TypeOfbook" HeaderText="TypeOfbook" SortExpression="TypeOfbook" />
<asp:BoundField DataField="Description" HeaderText="Description" SortExpression="Description" />
<asp:TemplateField HeaderText="Delete" ShowHeader="False">
<ItemTemplate>
<asp:LinkButton ID="Button1" runat="server" CausesValidation="False" CommandName="Delete"
Text="Видалити"></asp:LinkButton>
</ItemTemplate>
</asp:TemplateField>
</Columns>
<FooterStyle BackColor="White" ForeColor="#000066" />
<HeaderStyle BackColor="#006699" Font-Bold="True" ForeColor="White" />
<PagerStyle BackColor="White" ForeColor="#000066" HorizontalAlign="Left" />
<RowStyle ForeColor="#000066" />
<SelectedRowStyle BackColor="#669999" Font-Bold="True" ForeColor="White" />
<SortedAscendingCellStyle BackColor="#F1F1F1" />
<SortedAscendingHeaderStyle BackColor="#007DBB" />
<SortedDescendingCellStyle BackColor="#CAC9C9" />
<SortedDescendingHeaderStyle BackColor="#00547E" />
</asp:GridView>

</div>
</form>
```

```
</body>
</html>
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Security.Cryptography;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace JymYavornitskiy
{
public partial class UserMain1 : System.Web.UI.Page
{
protected void Page_Load(object sender, EventArgs e)
{
try
{
TextBox3.Text = Request.QueryString["field1"];
}
catch (Exception)
{



}
System.Globalization.CultureInfo ci = new
System.Globalization.CultureInfo("uk-UA");
System.Threading.Thread.CurrentThread.CurrentCulture = ci;
System.Threading.Thread.CurrentThread.CurrentUICulture = ci;
}

protected void Calendar1_SelectionChanged(object sender, EventArgs e)
{
String[] words = Calendar1.SelectedDate.ToString().Split(new char[] { ' ' },
StringSplitOptions.RemoveEmptyEntries);
TextBox1.Text = words[0];
}
```

```
protected void Button1_Click(object sender, EventArgs e)
{

if (TextBox1.Text.Length >= 1 && TextBox4.Text.Length >= 1 && TextBox2.Text.Length >= 1)
{
SqlConnection con =
new SqlConnection(
" Data Source=USER-PC;Initial Catalog=GymEnd;Integrated Security=True");
con.Open();

String[] timesStrings = Calendar1.SelectedDate.ToString().Split(new char[] { ' ' },
StringSplitOptions.RemoveEmptyEntries);
String[] yearmounthday = timesStrings[0].Split(new char[] { '.' }, StringSplitOptions.RemoveEmptyEntries);
TextBox1.Text = timesStrings[0];

string[] timeStrings = TextBox4.Text.Split(new char[] { ':' }, StringSplitOptions.RemoveEmptyEntries);
SqlCommand cmd =
new SqlCommand(
"insert into ToBook(Date, TimeEnd,UserID,TypeOfbook,Description) values('" + TextBox1.Text+"
"+TextBox4.Text +
"','" + TextBox2.Text + "','" + Int32.Parse(TextBox3.Text) + "','"+DropDownList1.Text+"','" +
TextBox6.Text+ "')", con);
cmd.ExecuteNonQuery();
con.Close();
string display = "Данні додано, спортзал заброньовано";
ClientScript.RegisterStartupScript(this.GetType(), "myalert", "alert('" + display + "');", true);
}
else
{
Label1.Visible = true;
string display = "Помилка : Уведіть данні";
ClientScript.RegisterStartupScript(this.GetType(), "myalert", "alert('" + display + "');", true);
}
Response.Redirect("http://localhost:60959/UserMain.aspx"+"?field1="+TextBox3.Text);



}
```

```
}
}
```

Лістинг сторінки реєстрації:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Registration.aspx.cs"
Inherits="JymYavornitskiy.Registration" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>
<div style="width: 50%; height: 50%; position: relative; top: 5px; left: 238px;"><table>
<tr><td>Ім'я</td><td><asp:TextBox ID="TextBox1" runat="server"></asp:TextBox></td></tr>
<tr><td>Логін</td><td><asp:TextBox ID="TextBox2" runat="server"></asp:TextBox></td></tr>
<tr><td>Пароль</td><td><asp:TextBox ID="TextBox3" runat="server"
TextMode="Password"></asp:TextBox></td></tr>
<tr><td>Підтвердіть пароль</td><td><asp:TextBox ID="TextBox7" runat="server"
TextMode="Password"></asp:TextBox></td></tr>
<tr><td>
<asp:Button ID="Button2" runat="server" OnClick="Button2_Click" Text="Увійти" Visible="False" />
</td><td><asp:Button ID="Button1" runat="server" Text="Зареєструватись"
OnClick="Button1_Click"></asp:Button></td></tr>
<tr><td colspan="2"> <asp:Label ID="Label1" runat="server" Text="Реєстрацію успішно завершено!"
Visible="False" ForeColor="Black"></asp:Label></td></tr>

</table></div>

</div>
</form>
</body>
</html>
using System;
```

```
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace JymYavornitskiy
{
public partial class Registration : System.Web.UI.Page
{
static string MD5(string tohash)
{

byte[] hash = Encoding.ASCII.GetBytes(tohash);

MD5 md5 = new MD5CryptoServiceProvider();

byte[] hashenc = md5.ComputeHash(hash);

string result = "";

foreach (var b in hashenc)
{

result += b.ToString("x2");

}

return result;

}
protected void Page_Load(object sender, EventArgs e)
{

}
```

```
protected void Button1_Click(object sender, EventArgs e)
{
if (TextBox3.Text == TextBox7.Text)
{
SqlConnection con =
new SqlConnection(
" Data Source=USER-PC;Initial Catalog=GymEnd;Integrated Security=True");
con.Open();
SqlCommand cmd =
new SqlCommand(
"INSERT INTO User (Name, Login, Password) VALUES ('" + TextBox1.Text +
"','" + TextBox2.Text + "','" + MD5(TextBox3.Text) + "')", con);
cmd.ExecuteNonQuery();
con.Close();
Label1.Visible = true;
Button1.Visible = false;
}
else
{
Label1.Visible = true;
Label1.Text = "Не вірно повторений пароль";
}
}

protected void Button2_Click(object sender, EventArgs e)
{
Response.Redirect("http://localhost:60959/Main.aspx");
}
}
}
```

Лістинг головної сторінки

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Main.aspx.cs"
Inherits="JymYavornitskiy.WebForm1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>

<div style="background-position: center; background-image: url('Image/zal1.jpg'); position: relative;
background-attachment: scroll; width: 100%; height: 187px; top: 0px; left: 0px;">
<div style="position: relative; top: 18%; left: 70%; height: 55%; width: 23%; background-color:
#FFFFFF;">
 
<asp:Label ID="Label1" runat="server" ForeColor="Black" Text="Авторизація"></asp:Label>
 :
<asp:Label ID="Label4" runat="server" Text="ЙЙЙ" Visible="False"></asp:Label>
<br />
<asp:Label ID="Label2" runat="server" Text="Логін"></asp:Label>
  
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
<br />
<asp:Label ID="Label3" runat="server" Text="Пароль"></asp:Label>
<asp:TextBox ID="TextBox2" runat="server" OnTextChanged="TextBox2_TextChanged"></asp:TextBox>
<asp:TextBox ID="TextBox3" runat="server" Visible="False"></asp:TextBox>
<br />
                  
<asp:Button ID="Button1" runat="server" Text="Увійти" OnClick="Button1_Click" />
<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$
ConnectionStrings:GymEndConnectionString %>" SelectCommand="SELECT [Login], [Password], [Id]
FROM [User] WHERE (([Login] = @Login) AND ([Password] = @Password))">
<SelectParameters>
<asp:ControlParameter ControlID="TextBox1" Name="Login" PropertyName="Text" Type="String" />
<asp:ControlParameter ControlID="TextBox3" Name="Password" PropertyName="Text" Type="String" />
</SelectParameters>
</asp:SqlDataSource>
</div>
</div>
```

```
</div>
<div>
<asp:Calendar ID="Calendar1" runat="server" Height="154px"
OnSelectionChanged="Calendar1_SelectionChanged" Width="594px"></asp:Calendar>
<asp:TextBox ID="TextBox4" runat="server" AutoPostBack="True" Enabled="False"
OnTextChanged="TextBox4_TextChanged"> </asp:TextBox>
<asp:TextBox ID="TextBox5" runat="server"></asp:TextBox>
<asp:TextBox ID="TextBox6" runat="server"></asp:TextBox>
<br />
<asp:SqlDataSource ID="SqlDataSource2" runat="server" ConnectionString="<%$
ConnectionStrings:GymEndConnectionString %>" SelectCommand="SELECT Date AS [Дата початку],
TimeEnd AS [Дата закінчення], TypeOfbook AS [Тип відвідуваня], Description AS [Опис відвідуваня]
FROM ToBook
WHERE (DATEPART(yy, date) = @Year
AND DATEPART(mm, date) = @mounth
AND DATEPART(dd, date) = @day)">
<SelectParameters>
<asp:ControlParameter ControlID="TextBox4" Name="Year" PropertyName="Text" />
<asp:ControlParameter ControlID="TextBox5" Name="mounth" PropertyName="Text" />
<asp:ControlParameter ControlID="TextBox6" Name="day" PropertyName="Text" />
</SelectParameters>
</asp:SqlDataSource>
<div id="gridview">
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
DataSourceID="SqlDataSource2">
<Columns>
<asp:BoundField DataField="Дата початку" HeaderText="Дата початку" SortExpression="Дата початку"
/>
<asp:BoundField DataField="Дата закінчення" HeaderText="Дата закінчення" SortExpression="Дата
закінчення" />
<asp:BoundField DataField="Тип відвідуваня" HeaderText="Тип відвідуваня" SortExpression="Тип
відвідуваня" />
<asp:BoundField DataField="Опис відвідуваня" HeaderText="Опис відвідуваня" SortExpression="Опис
відвідуваня" />
</Columns>
</asp:GridView>
</div>
```

```
 </div>
<div id="hidenoops" aria-hidden="False" runat="server">
<asp:Image ID="Image1" runat="server" Height="253px" ImageUrl="~/Image/Q4zyk-wpRMs.jpg"
Width="292px" />
<br />
<asp:Label ID="Label5" runat="server" Font-Size="X-Large" Text="Записів на цей день
немає"></asp:Label>
</div>
</form>
</body>
</html>
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace JymYavornitskiy
{
public partial class WebForm1 : System.Web.UI.Page
{
static string MD5(string tohash)
{

byte[] hash = Encoding.ASCII.GetBytes(tohash);

MD5 md5 = new MD5CryptoServiceProvider();

byte[] hashenc = md5.ComputeHash(hash);

string result = "";

foreach (var b in hashenc)
```

```
{

result += b.ToString("x2");

}

return result;

}
protected void Page_Load(object sender, EventArgs e)
{
hidenoops.Visible = false;
System.Globalization.CultureInfo ci = new
System.Globalization.CultureInfo("uk-UA");
System.Threading.Thread.CurrentThread.CurrentCulture = ci;
System.Threading.Thread.CurrentThread.CurrentUICulture = ci;
}

protected void Button1_Click(object sender, EventArgs e)
{
try
{
DataView dView = (DataView)SqlDataSource1.Select(DataSourceSelectArguments.Empty);
if (TextBox1.Text == "admin" && TextBox2.Text == "123")
{
Response.Redirect("http://localhost:60959"+ "/AdmminMain.aspx");
}
else
if (dView.Count >= 1)
{
//+dView.ToTable().Columns[0].;
/* DataRowView rowViews;
rowViews.DataView = dView*/
/*foreach (DataRowView rowView in dView)
{
DataRow row = rowView.Row;
Label3.Text = row[2].ToString();
}*/
```

```
Label3.Text = dView[0]["Id"] as string;
Response.Redirect("http://localhost:60959" + "/UserMain.aspx?field1=" + dView[2]["Id"] as string);
/*GridView newGridview1 = new GridView();
newGridview1.SelectMethod = SqlDataSource1.Select(DataSourceSelectArguments.Empty);
newGridview1.Rows[4].Cells["Name"].Value.ToString();*/
}
else
{
Label3.Text = "ERROR";
}
}
catch (Exception)
{

}
Label3.Visible = true;
}

protected void TextBox2_TextChanged(object sender, EventArgs e)
{
TextBox3.Text = MD5(TextBox2.Text);
}

protected void Calendar1_SelectionChanged(object sender, EventArgs e)
{
String[] words = Calendar1.SelectedDate.ToString().Split(new char[] { ' ' },
StringSplitOptions.RemoveEmptyEntries);
//TextBox4.Text = words[0];
String[] dateStrings = words[0].ToString().Split(new char[] { '.' }, StringSplitOptions.RemoveEmptyEntries);
TextBox4.Text = dateStrings[2];
TextBox5.Text = dateStrings[1];
TextBox6.Text = dateStrings[0];
DataView dv = (DataView)SqlDataSource2.Select(DataSourceSelectArguments.Empty);
if (dv.Table.Rows.Count >= 1)
{
hidenoops.Visible = false;
}
else
```

```
{
hidenoops.Visible = true;
}



}

protected void TextBox4_TextChanged(object sender, EventArgs e)
{
/* SqlConnection coneConnection = new SqlConnection(" Data Source=USER-PC;Initial
Catalog=GymEnd;Integrated Security=True");
coneConnection.Open();
SqlCommand cmd = new SqlCommand("SELECT Date AS [Дата початку], TimeEnd AS [Дата
закінчення], TypeOfbook AS [Тип відвідуваня], Description AS [Опис відвідуваня] FROM ToBook
WHERE(TimeEnd Like '%"+TextBox4.Text+"%')",coneConnection);
cmd.ExecuteNonQuery();
coneConnection.Close();
GridView1.DataSource = cmd;*/

}
}
}
```

Лістинг сторінки адміністратора:

```
<%@ Page Language="C#" AutoEventWireup="true" Culture="Auto" UICulture="uk-UA"
CodeBehind="AdminMain.aspx.cs" Inherits="JymYavornitskiy.UserMain" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>
```

```
</div>
<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$
ConnectionStrings:GymEndConnectionString %>" SelectCommand="SELECT Date AS [Дата початку],
TimeEnd AS [Дата закінчення], TypeOfbook AS [Тип відвідуваня], Description AS [Опис відвідуваня]
FROM ToBook
WHERE (DATEPART(yy, date) = @Year
AND DATEPART(mm, date) = @mounth
AND DATEPART(dd, date) = @day)" DeleteCommand="DELETE FROM ToBook">
<SelectParameters>
<asp:ControlParameter ControlID="TextBox4" Name="Year" PropertyName="Text" />
<asp:ControlParameter ControlID="TextBox5" Name="mounth" PropertyName="Text" />
<asp:ControlParameter ControlID="TextBox6" Name="day" PropertyName="Text" />
</SelectParameters>
</asp:SqlDataSource>
<asp:Calendar ID="Calendar1" runat="server" BackColor="White" BorderColor="White"
BorderWidth="1px" Font-Names="Verdana" Font-Size="9pt" ForeColor="Black" Height="235px"
NextPrevFormat="FullMonth" Width="451px" OnSelectionChanged="Calendar1_SelectionChanged1">
<DayHeaderStyle Font-Bold="True" Font-Size="8pt" />
<NextPrevStyle Font-Bold="True" Font-Size="8pt" ForeColor="#333333" VerticalAlign="Bottom" />
<OtherMonthDayStyle ForeColor="#999999" />
<SelectedDayStyle BackColor="#333399" ForeColor="White" />
<TitleStyle BackColor="White" BorderColor="Black" BorderWidth="4px" Font-Bold="True" Font-
Size="12pt" ForeColor="#333399" />
<TodayDayStyle BackColor="#CCCCCC" />
</asp:Calendar>
<br />
<asp:TextBox ID="TextBox1" runat="server" AutoPostBack="True" Visible="False"></asp:TextBox>
<asp:TextBox ID="TextBox4" runat="server"></asp:TextBox>
<asp:TextBox ID="TextBox5" runat="server"></asp:TextBox>
<asp:TextBox ID="TextBox6" runat="server"></asp:TextBox>
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
DataSourceID="SqlDataSource1">
<Columns>
<asp:BoundField DataField="Дата початку" HeaderText="Дата початку" SortExpression="Дата початку"
/>
<asp:BoundField DataField="Дата закінчення" HeaderText="Дата закінчення" SortExpression="Дата
закінчення" />
```

```
<asp:BoundField DataField="Тип відвідуваня" HeaderText="Тип відвідуваня" SortExpression="Тип
відвідуваня" />
<asp:BoundField DataField="Опис відвідуваня" HeaderText="Опис відвідуваня" SortExpression="Опис
відвідуваня" />
<asp:CommandField />
<asp:TemplateField ShowHeader="False">
<EditItemTemplate>
<asp:LinkButton ID="LinkButton1" runat="server" CausesValidation="True" CommandName="Update"
Text="Обновить"></asp:LinkButton>
 <asp:LinkButton ID="LinkButton2" runat="server" CausesValidation="False"
CommandName="Cancel" Text="Отмена"></asp:LinkButton>
</EditItemTemplate>
<ItemTemplate>
<asp:LinkButton ID="Button1" runat="server" CausesValidation="False" CommandName="Edit"
Text="Редагувати"></asp:LinkButton>
</ItemTemplate>
</asp:TemplateField>
<asp:TemplateField ShowHeader="False">
<ItemTemplate>
<asp:LinkButton ID="Button2" runat="server" CausesValidation="False" CommandName="Delete"
Text="Видалити"></asp:LinkButton>
</ItemTemplate>
</asp:TemplateField>
</Columns>
</asp:GridView>
</form>
</body>
</html>
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace JymYavornitskiy
{
public partial class UserMain : System.Web.UI.Page
```

```
{
protected void Page_Load(object sender, EventArgs e)
{
System.Globalization.CultureInfo ci = new
System.Globalization.CultureInfo("uk-UA");
System.Threading.Thread.CurrentThread.CurrentCulture = ci;
System.Threading.Thread.CurrentThread.CurrentUICulture = ci;
}
protected void Calendar1_SelectionChanged1(object sender, EventArgs e)
{
String[] words = Calendar1.SelectedDate.ToString().Split(new char[] { ' ' },
StringSplitOptions.RemoveEmptyEntries);
//TextBox4.Text = words[0];
String[] dateStrings = words[0].ToString().Split(new char[] { '.' }, StringSplitOptions.RemoveEmptyEntries);
TextBox4.Text = dateStrings[2];
TextBox5.Text = dateStrings[1];
TextBox6.Text = dateStrings[0];
}
}
}
```