Державни торговельно-економічний університет

Кафедра комп'ютерних наук та інформаційних систем

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
на тему:

«Розробка мобільного додатка для управління замовленнями і доставками товарів»

Студента 4 курсу, групи 13,
спеціальності
122 «Комп'ютерні науки» _____ Острінський Нікіта Русланович
підпис студента

Науковий керівник
доктор фізико-математичних наук, професор _____ Пурський Олег Іванович
підпис керівника

Гарант освітньої програми
кандидат технічних наук, доцент _____ Демідов Павло Георгійович
підпис керівника

Київ 2023

Київський національний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра комп'ютерних наук та інформаційних систем
Спеціальність 122 «Комп'ютерні науки»

Затверджую
Зав. кафедри _____Пурський О.І..
«12 » грудня 2022р.

Завдання
на випускну кваліфікаційну роботу (проект) студента

Остринський Нікіта Русланович
(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної роботи (проекту)
«Розробка мобільного додатка для управління замовленнями і доставками товарів»
Затверджена наказом ректора від *«09» грудня 2022 р. № 3332*
2. Строк здачі студентом закінченої роботи *30 травня2023 року*
3. Цільова установка та вихідні дані до роботи
Мета роботи: Розробка мобільного додатка для управління заказами і доставками товарів
Об'єкт дослідження: процеси управління замовленнями і доставками товарів .
Предмет дослідження: інформаційні системи і технології в системі управління замовленнями і доставками товарів
4. Перелік графічного матеріалу_____
_____
_____

5. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Шклярський С.М.		
2	Шклярський С.М.		
3	Шклярський С.М.		

6. Зміст випускної кваліфікаційної роботи (проекту) (перелік питань за кожним розділом)

ЗМІСТ

ВСТУП

ГЛАВА 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Опис предметної області
1.2. Функціональність програми
1.3. Огляд існуючих додатків з доставки продуктів
1.4. Вибір засобів розробки і реалізація 1

ГЛАВА 2. ПРОЕКТУВАННЯ
2.1. Розроблення архітектури програмної системи
2.2. Проектування структури бази даних

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ
3.1. Програмна реалізація проекту
3.2. Програмна реалізація бази даних

РОЗДІЛ 4 ТЕСТУВАННЯ ТА ДОСЛІДНА ЕКСПЛУАТАЦІЯ
4.1. Тестування
4.2. Розгортання програмного продукту

ВИСНОВКИ
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ
ДОДАТОК А DDL-код створення таблиць у базі даних
ДОДАТОК Б Лістинг головних модулів

7. Календарний план виконання роботи

№ Пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	фактично
1	2	3	4
1	*Вибір теми випускної кваліфікаційної роботи*	*04.10.2022*	*04.10.2022*

2	*Розробка та затвердження завдання на випускну кваліфікаційну роботу*	*15.12.2022*	*15.12.2022*
3	*Вступ*	*03.02.2023*	*03.02.2023*
4	*РОЗДІЛ 1. Аналітичне дослідження специфіки діяльності закладів вищої освти в Україні*	*28.02.2023*	*28.02.2023*
5	*РОЗДІЛ 2. Розробка моделі перевірки та оцінювання знань студентів*	*06.04.2023*	*06.04.2023*
6	*РОЗДІЛ 3. Інформаційна система перевірки та оцінювання знань студентів*	*12.05.2023*	*12.05.2023*
7	*Висновки*	*15.05.2023*	*15.05.2023*
8	*Здача випускної кваліфікаційної роботи на кафедру науковому керівнику*	*30.05.2023*	*30.05.2023*
9	*Попередній захист випускної кваліфікаційної роботи*	*31.05.2023 -01.06.2023*	*31.05.2023 -01.06.2023*
11	*Виправлення зауважень, зовнішнє рецензування випускної кваліфікаційної роботи*	*02.06.2023*	*02.06.2023*
12	*Представлення готової зшитої випускної кваліфікаційної роботи на кафедру*	*05.06.2023*	*05.06.2023*
13	*Публічний захист випускної кваліфікаційної роботи*	*За розкладом роботи ЕК*	

8. Дата видачі завдання « » 2022 р.

9. Керівник випускної кваліфікаційної роботи (проекту) Пурський О.І.
(прізвище, ініціали, підпис)

10. Гарант освітньої програми Демідов П.Г.
(прізвище, ініціали, підпис)

11. Завдання прийняв до виконання студент-дипломник
Остринський Н.Р.
(прізвище, ініціали, підпис)

12. Відгук керівника випускної кваліфікаційної роботи (проекту)

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____

Керівник випускної кваліфікаційної роботи (проекту)

____
(підпис, дата)

13. Висновок про випускну кваліфікаційну роботу (проект)

Випускна кваліфікаційна робота (проект) студента ____________________
(прізвище, ініціали)
може бути допущена до захисту в екзаменаційній комісії.

Гарант освітньої програми______________________Демідов П.Г.______
(підпис, прізвище, ініціали)

Зав кафедри __________________________________Пурський О.І.______
(підпис, прізвище, ініціали)
«_____»_________________2023 р.

ЗМІСТ

ВСТУП

Актуальність. У сучасному світі всі хоч раз стикалися з проблемою браку часу або фізичної можливості (через захворювання) для відвідування магазину і здійснення повсякденної покупки. А враховуючи, що магазини періодично влаштовують акції на різні товари і продають їх з хорошою знижкою, перед покупцями постає питання не тільки купити необхідний товар, а й постаратися заощадити на покупці. Але їхати в магазин, розташований далеко від дому або місця роботи (навчання) - не завжди зручно. У зв'язку з цим актуально мати під рукою мобільний додаток, що дозволяє не тільки знайти товар, вигідний за ціною, але і замовити його з доставкою додому або самовивозом. На сьогоднішній день існує певна кількість додатків, що реалізують доставку товарів з магазинів. І попит на подібні додатки для мобільних пристроїв стабільно зростає вже кілька останніх років. Тому актуальність розробки додатків, які надають допомогу в пошуку магазину або товару, цілком доцільна і обов'язково отримає відповідне визнання користувачів.

Метою даної роботи є розробка мобільного застосування на ОС Android [3], за допомогою якого будь-який користувач зможе замовити додому або забрати самовивозом продукти харчування чи інші товари.

У мобільному додатку буде реалізований:

• Простий і зрозумілий інтерфейс для користувача;

• Повний опис кожного товару, включаючи склад товару, його виробника і терміни придатності;

• Перегляд акцій всіх магазинів;

• Відображення інформації про попередні ціни на товар і виводитися у вигляді графіка;

• Відображення найбільш купованих товарів за місяць або тиждень.

Цільова аудиторія проекту - магазини (супермаркети) і споживачі послуг доставки їжі додому.

Об'єктом дослідження даної роботи є процеси розробки мобільного

додатку на замовлення товарів і продуктів, що знаходяться в наявності у супермаркеті.

Для досягнення зазначеної мети були поставлені та вирішені наступні завдання:

1) Провести аналіз і порівняти схожі мобільні додатки;

2) Вивчити технології необхідні для розробки програми;

3) Вивчити платформи необхідні д ля розробки мобільних додатків;

4) Вивчити платформу "Android Studio";

5) Розробити мобільний додаток за допомогою платформи і технологій "Android Studio";

Теоретичне значення роботи полягає в опрацюванні і подальший розвиток методів та інструментів для розробки мобільних додатків.

Практична значимість даної роботи дозволяє використовувати додаток для замовлення продуктів додому.

Структура. Робота складається зі вступу, трьох розділів, висновків, списку використаної літератури та додатку. У першому розділі описано предметну область, способи оплати, функціональність програми.У другому розділі розглянуті засоби для розробки мобільного застосування, а саме, платформи, які були використані. У третьому розділі розписаний процес розробки мобільного застосування і спосіб автоматичної категоризації товарів на основі нейронної мережі, який увійшов в мобільний додаток.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

У цьому розділі розглядається опис предметної області, функціонал сайту, переваги та недоліки сайту на замовлення товарів і продуктів, аналоги розробленого, засоби розробки і реалізації.

1.1. Опис предметної області

Розробка мобільних додатків під Android на сьогоднішній день дуже затребувана через високу популярність даної ОС [9]. Кожне мобільний додаток має певне призначення. Перевагою використання мобільного застосування на замовлення покупок через інтернет є:

• Економія часу - відпадає необхідність поїздки в магазин для здійснення покупки, а доставка покупок здійснюється в строго обумовлений час;

• Економія коштів - магазини регулярно проводять акції на певні товари, мобільний додаток дозволяє своєчасно їх відслідковувати; крім того, в ньому можна знайти товар, що цікавить за найнижчою ціною серед усіх магазинів.

Додаток, що розробляється в рамках роботи, дозволяє здійснити перегляд і покупку товарів з магазинів. Споживачеві набагато простіше зайти в один додаток, в якому зосереджений максимум можливих варіантів товарів для вибору і покупки, ніж заходити в окремий додаток кожного магазину. Встановивши даний мобільний додаток, споживач крім асортименту отримує і іншу інформацію - повний опис кожного товару, включаючи склад товару, його виробника і терміни придатності; акції магазинів; зміна ціни на товар за проміжок часу і т.д.

Доставка обраних товарів може бути здійснена трьома способами:

1) самовивіз;

2) Доставка кур'єром;

3) Самовивіз з місця перебування кур'єра.

Способи оплати:

1. Безготівковий спосіб оплати через термінал доставки в зоні самовивозу або у кур'єра;

2. Готівковими коштами кур'єру.

1.2 Аналіз етапів проектування веб-додатаку

Розробка веб-додатку – це процес створення веб-сторінок або сайтів. Веб-сторінки створюються з використанням HTML, CSS і JavaScript і т.д. Сторінки можуть містити простий текст і графіку, нагадуючи собою статичний документ. Сторінки також можуть бути інтерактивними або відображати інформацію, що змінюється. Створювати інтерактивні сторінки складніше, але вони дозволяють створювати веб-сайти з більш наповненим вмістом. Сьогодні більшість сторінок інтерактивні і надають сучасні інтерактивні послуги, такі як корзини Internet-магазинів, динамічна візуалізація та навіть складні соціальні мережі [3].

Створенню конкретної веб- додатку передує детальний комплексний аналіз, визначаючий критерії, яким вона має відповідати. Процес створення включає шість основних етапів:

- визначення цілей та задач;
- розробка структури;
- розробка дизайн-макетів;
- HTML-верстка;
- програмування та контроль якості;

● запуск та оптимізація.

Кожен з перелічених етапів самодостатній, що дозволяє обирати схему роботи і виконавця для кожного з них окремо. Розглянемо кожний етап більш детально. На етапі проектування формуються бізнес-цілі проекту, що створюється, визначаються вимоги, яким він повинен відповідати, розробляється загальна концепція. Під час роботи на цьому етапі уточнюються вимоги замовника, формується технічне завдання, виконується аналіз цільової аудиторії. Для більш глибокого аналізу можна запросити у замовника відповідні матеріали: брошури, щорічні звіти, зразки продукції, інші супутні дані - все, що допоможе скласти уявлення про те, хто і з якою метою буде відвідувати сайт, які завдання будуть виконуватися на сайті. Важливо з'ясувати технічні можливості майбутньої основної користувацької аудиторії - пропускну спроможність каналів зв'язку, які використовуються Internet-браузери і т.д.

Для знаходження цільової аудиторії доцільно увійти в роль кваліфікованого користувача на аналогічно створюваних Internet-ресурсах. Це допоможе виявити нові креативні концепції для того, щоб сайт був більш конкурентоспроможним і «не загубився» серед безлічі інших. Коли цілі визначені, приступають до складання розширеного плану проекту, що відображає скільки часу, грошей та інших коштів знадобиться для виконання робіт на кожному з наступних етапів. Такий план часто містить інформацію про бюджет проекту, графік робіт (з відповідним розподілом ролей між веб-розробниками), технічну документацію, а також розділ «деталей і уточнень», де обумовлені конкретні аспекти можливих спірних питань. У цей розділ також включають пропозиції по готових розробках і шаблонах.

Наступний етап – розробка структури. Включає в себе зміст сторінки в якій буде розміщено веб- додаток, а також – інформаційну стратегію, яка визначає, як організована подача інформації, щоб майбутні користувачі могли легко та зручно використовувати надані ресурси. Головною задачею цього етапу розробки є створення карти, яка відображає зв'язок сторінок та їх

найбільш значущі функціональні можливості. Її подають у вигляді блок-схеми, на якій кожна сторінка відображається окремим прямокутником, зв’язки між ними означають переходи між сторінками.

Дизайн-макет – це графічне, наглядне зображення елементів сайту. Дизайн-макет повністю втілює візуальну концепцію сайту. Його розробка виконується в одній з графічних програм (у переважній більшості випадків - в Adobe Photoshop). У процесі розробки дизайнер керується письмовою угодою (брифом) на створення дизайн-макету, який заповнюється замовником і містить побажання до дизайну: тип, кольорова гама, наявність тих чи інших графічних елементів, тощо. На цій стадії створюються всі елементи веб-дизайну відповідно до стилю подачі інформації та загальної концепції.

Головним при дизайні сайту є вміння розробити графічні об'єкти, які б швидко завантажувалися і добре виглядали, незалежно від використовуваного Internet-браузера. Часто вдаються до використання готових дизайн-шаблонів, які широко представлені в мережі Internet або є вбудованими в різні графічні редактори, такі як, Microsoft FrontPage або Adobe Photoshop. За допомогою подібних шаблонів сайт створюється за максимально стислий час. Однак слід зазначити, що у такого рішення є ряд істотних недоліків, головний з яких - повторюваність і не унікальність дизайну. Шаблон є оболонкою з мінімальною кількістю інтерактивних елементів і корисних модулів. Тому при виборі шаблону варто звертати увагу не тільки на дизайн, але і на функціональність.

Важливим елементом дизайну є графіка, яку умовно розділяють на три категорії:

- ілюстративна графіка – пояснювальні зображення, схеми та графіки, фото;
- функціональна графіка – кнопки навігації, лічильники та інші елементи управління;
- декоративна графіка – естетичні елементи дизайну, такі як фоновий

малюнок, заголовки, банери, рамки.

Така класифікація передбачає чітке розмежовування форматів і передбачає використання певних категорій по максимуму (наприклад, без використання функціональної графіки навігація буде не зручною в той час як без декоративного оформлення сайт буде більш швидко завантажуватися і буде більш простим для сприйняття, не буде перевантажений графікою). HTML-верстка макета є наступним кроком після розробки сайту.

Верстка - це перетворення створених дизайнером графічних макетів сторінок в HTML-код, який би відображався в Internet-браузері в точній відповідності до вихідного макету. Складність верстки залежить від складності дизайну. Основними завданнями при верстці є:

● коректність відображення сторінок сайту при різних розширеннях екрану;

● кросбраузерність – однакове відображення сторінок сайту в найбільш популярних браузерах - Internet Explorer, Mozilla Firefox, Opera, Chrome.

Програмування - це практична реалізація проекту, інтеграція напрацювань за окремими напрямками. Іншими словами, це процес побудови функціональних інструментів для наповнення і обробки даних. Програмування визначає наскільки стабільним і захищеним буде функціонування сайту. Вибір платформи, технологій і грамотного підходу до програмування відіграє істотну роль. На даному етапі важливо визначитися з підходом до створення веб- системи: чи буде вона статичною або динамічною.

Створення веб- додатку, як і будь-якого іншого програмного продукту, зіштовхується з проблемою постійної зміни даних та файлів. Контроль за змінами, що вносяться в проект, допомагають забезпечити системи управління версіями (Version Control System - VCS), які зберігають попередні версії вихідних файлів проекту, відстежують вироблені в файлах зміни, забезпечують спільну командну роботу над проектом. До найбільш популярних на поточний момент VCS відносяться: SVN, GIT, Microsoft VSS.

Використання системи контролю версій піднімає загальний рівень якості розробки.

По завершенні етапу активного програмування починається етап тестування коректності функціонування створеного веб- додатку: перевірки на наявність граматичних помилок, пропущених картинок, непрацюючих посилань і т.д., а також перевірки функціонування сайту в різних веб-браузерах.

1.3. Огляд існуючих додатків з доставки продуктів

Zakaz.ua

Сервіс доставки продуктів із супермаркетів Auchan, Novus, Metro, «Фуршет» і «МегаМаркет». На сайті кожного магазину попереджають про відсутність певних товарів і можливу затримку через велике навантаження. Нині служба доставки дуже завантажена та планує запустити волонтерську програму, щоб мати змогу обслуговувати клієнтів оперативніше, розповіли The Village Україна в Zakaz.ua.

Зона доставки: Київ та околиці

Вартість доставки: від 69 грн; від 45 кілограмів – доплата за вагу;

Приблизний час очікування: 4-5 днів

Fozzy

За останні дні попит на доставку продуктів дуже зріс, кажуть у Fozzy Group. Вільний час для доставки покупці можуть відстежувати прямо на сайті. Також у магазині є послуга Fozzy Drive, коли можна онлайн замовити продукти, оплатити їх карткою, вибрати час, заїхати до Fozzy C&C і забрати зібраний кошик. У компанії рекомендують телефонувати у службу доставку, щоб дізнаватися час очікування у реальному часі.

Зона доставки: увесь Київ та околиці

Вартість доставки: 69 грн (від 1 300 грн –безкоштовно)

Приблизний час очікування: день

FreshMart

Fresh Mart пропонує лише товари рослинного походження. Серед асортименту сезонні та екзотичні фрукти, овочі, зелень, гриби, горіхи, сухофрукти, бакалія (крупи, спеції, олія) та натуральні солодощі. Під час оформлення замовлення до 14:00 доставку здійснюють того ж дня. Є два часових проміжки доставки: денна з 12:00 до 18:00 та вечірня з 18:00 до 22:00. На час карантину кур'єрський сервіс працює і в деяких районах передмістя Києва (детальніше про доставку в передмісті варто уточнювати в оператора).

Зона доставки: Київ та інші міста України «Новою поштою»

Вартість доставки: 69 грн (від 750 грн – безкоштовно)

Приблизний час очікування: день

«Агроном»

Сервіс «Агроном» пропонує безкоштовну доставку овочів, фруктів, зелені, сухофруктів та екзотичних продуктів. «Щодня приїжджають поставки з фермерських угідь України та з-за кордону, що гарантує свіжість та якість кожного овоча, пучка зелені, цитрусу тощо», – розповідають в сервісі. Доставку виконують на наступний день після замовлення.

Зона доставки: Київ, Вишгород, Софіївська та Петропавлівська Борщагівка

Вартість доставки: безкоштовно

Приблизний час очікування: день

Takfur

В інтернет-магазині можна замовити продукти, а також товари для дому і гігієнічні засоби. У період карантину попереджають про можливі «заміни в замовленнях» на 20-40%. Замовлення приймають лише від 1 000 гривень.

Зона доставки: увесь Київ та околиці

Вартість доставки: 60 грн за доставку містом +10 гривень за кожен кілометр від Києва

Приблизний час очікування: 3 дні

Glovo

Окрім їжі з ресторанів, кур'єри Glovo доставляють продукти з магазинів «Ашан», «Континент», «АТБ», Winetime, Varus та «Молоко від фермера». Для безпеки покупців та кур'єрів у Glovo ввели режим безконтактної доставки: якщо це можливо, кур'єри залишають замовлення біля дверей. Щоб скористатися таким режимом, варто обрати опцію оплати карткою.

Зона доставки: центр Києва, Дніпровський район, частина Дарницького району, Вишневе

Вартість доставки: 60 грн

Приблизний час очікування: день

Kabanchik

Щоб замовити доставку на Kabanchik.ua, треба створити замовлення на сайті або в мобільному додатку. У заявці замовник має описати всі деталі доставки, залишити контактні дані та вказати адресу, куди треба привезти товар. Під час створення заявки треба вказати, де забрати чи придбати продукти (ресторан, кафе, магазин тощо), а також прикріпити список товарів.

У режимі карантину оперативність доставки залежить від того, чи має кур'єр власний транспорт. Якщо доставка термінова та об'ємна, краще зазначити, що кур'єр має бути на власному авто. Що ближче доставка, то оперативніше вона відбудеться. Час на неї можна обговорити безпосередньо з кур'єром. Якщо вас не влаштує цей термін, можна відхилити пропозицію та зачекати, доки сервіс підбере іншого кур'єра.

1.4. Вибір засобів розробки і реалізація

Вивчивши всі існуючі платформи - Microsoft Xamarin, Eclipse, Android studio, Intellij IDEA з розробки додатків під операційну систему Android для мобільних додатків, я вирішив зупинитися на платформі Android Studio, яка заснована на програмному забезпеченні IntelliJ IDEA від компанії JetBrains.

Відмінні сторони:

• Володіє хорошими можливостями для того, щоб редагувати макети, в тому числі підтримує функцію Drag and Drop, що спрощує процес роботи.

• Присутня функція Instant Run, яка дозволяє розробнику побачити різницю, після зміни коду і як зміна вплинула на результат.

• Це вдосконалений і доопрацьований додаток в плані зручності в порівнянні з IntelliJ IDEA, хоча зроблено на його програмному забезпеченні.

• Є можливість перетворити весь програмний код в APK формат, для установки на мобільний пристрій.

• на етапі тестування, ми можемо перевірити готовий продукт на різних пристроях, які розташовуються в дата-центрі.

Висновок до 1 розділу

Розробка веб- додатку - це комплексний багатокроковий процес, що вимагає знання безлічі різних технологій і мов програмування, вміння працювати з базами даних, використовувати безліч інструментальних засобів і програмних пакетів. Підсумовуючи все вищесказане, слід при створенні веб- системи пройти всі перелічені етапи та визначити задачі, параметри та цільову аудиторію.

ГЛАВА 2. ПРОЕКТУВАННЯ

У цьому розділі ми розглянемо обрані технології, які будемо використовувати для розробки мобільного застосування, з докладним описом кожної технології.

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 (Рисунок 2.1).

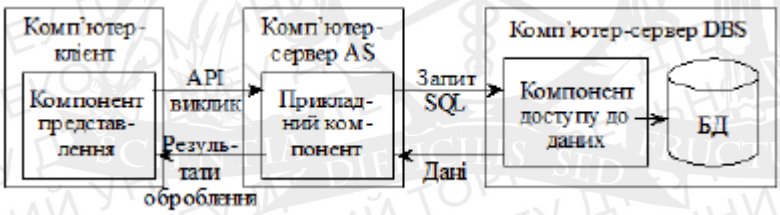

Рис. 2.1. Модель трирівневої архітектури

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.)

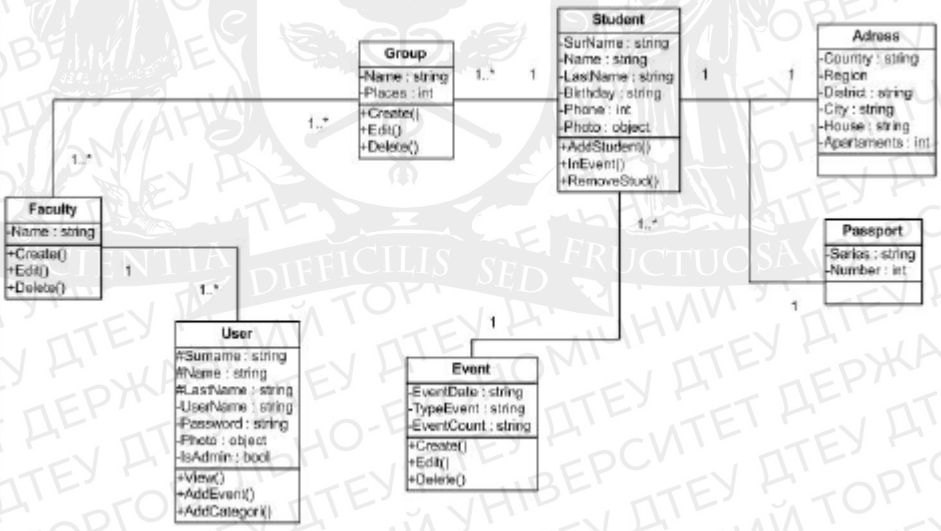

Рис.2.2. UML-діаграма класів

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

Функціональну структуру системи умовно можна розділити на такі модулі:

- модуль обробки інформації про користувачів:

1) реєстрація;

2) авторизація;

- модуль обробки інформації про собівартість продукції.

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.

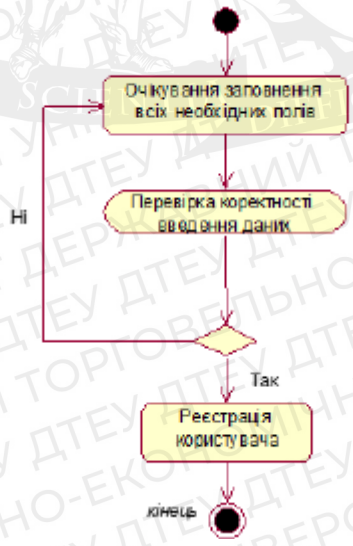

Рис.2.4. Діаграма активності для процесу реєстрації користувача

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].

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 собівартості продукції відбувається за допомогою

адміністратора, який взаємодіє з програмною системою. Адміністратор, переглянувши заявку, вибирає дату початку виконання події 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.

Наступним кроком є виявлення основних сутностей та зв’язків між ними.

Після цього створюють таблицю ідентифікаторів.

Завершальним етапом проектування бази даних є створення ER-діаграми.

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

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1. Програмна реалізація проекту

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.

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 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 нової продукції;
- редагування;
- додавання даних;
- перегляд інформації;
- редагування 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.

3.2. 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 передбачає створення однієї бази даних. Дана база передбачена для збереження собівартості продукції, даних користувачів та даних.

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 3.1.

Рис.3.1. 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 3.2. Це дозволяє створювати

нові бази даних і надалі працювати з ними. У даному середовищі також можливий імпорт/експорт та відновлення баз даних.

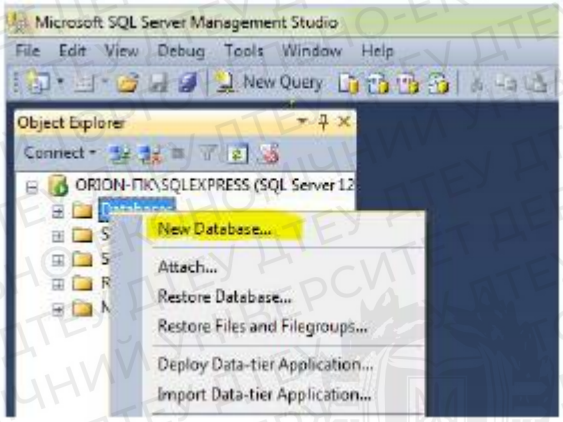

Рис.3.2. Створення нової бази даних

Результатом даної послідовності буде виклик вікна конфігурації нової бази даних що показано на рисунку 3.3. 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 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 3.4

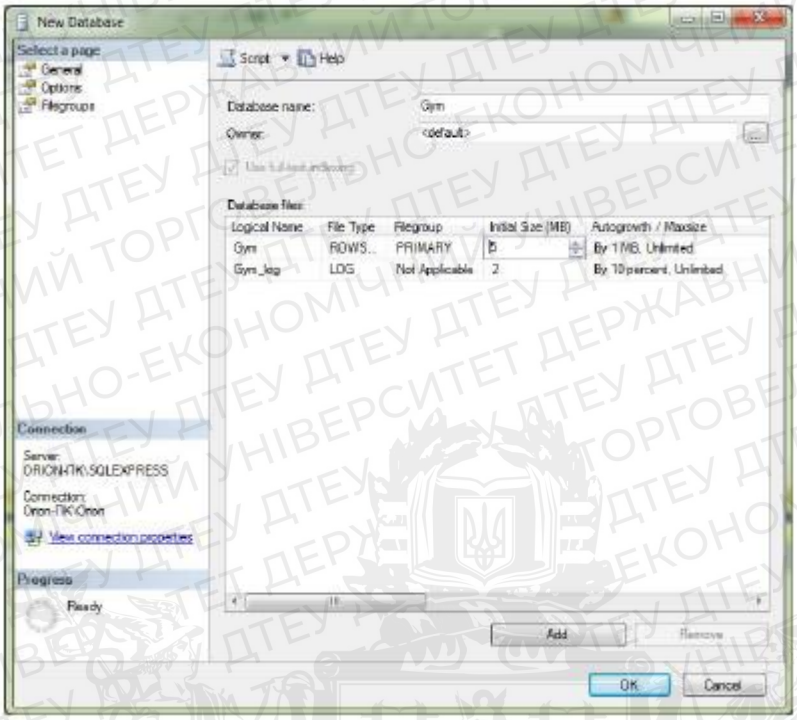

Рис.3.3. Вікно налаштування нового об’єкту бази даних

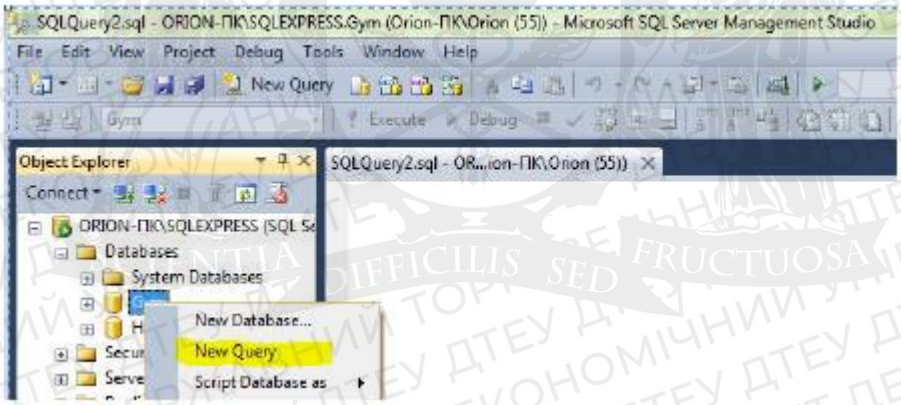

Рис.3.4 Відкриття редактора SQL-скриптів

На рисунку 3.5 зображено вікно редактора скриптів із результатом відлагодженого скрипта.

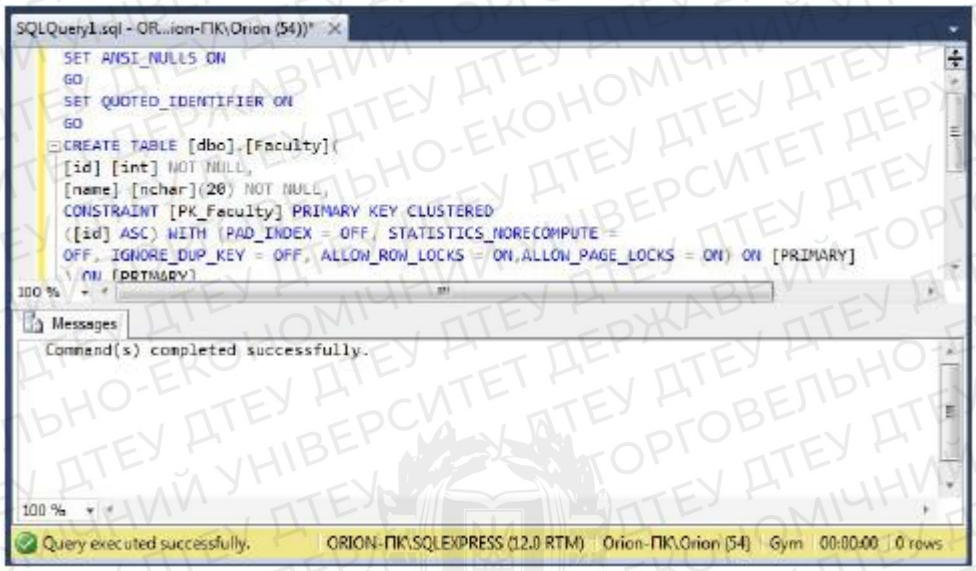

Рис.3.5. Вікно редактора скриптів

Код відлагодженого SQL-скрипта 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 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.

РОЗДІЛ 4 ТЕСТУВАННЯ ТА ДОСЛІДНА ЕКСПЛУАТАЦІЯ

4.1. Тестування

Для адміністрування інтернет-магазину існує відповідний розділ, він доступний тільки адміністраторам сайту. В даному розділі знаходиться стандартний набір сторінок дозволяють адміністратору:

• Додавати, видаляти, змінювати інформацію про товар;

• Редагувати дані користувачів;

• Призначати ролі зареєстрованим користувачам;

• Змінювати інформацію про інтернет-магазині;

• Переглядати список проданих товарів;

• Додавати, видаляти, змінювати категорії, підкатегорії і вміст інших таблиць-довідників.

Для виключення вірогідності втрати даних слід налаштувати регулярне резервне копіювання бази даних, причому найкраще зі збереженням резервних копій, наприклад, за останні кілька днів. Для цього можна використовувати або вбудований в SQL Server планувальник завдань - «SQL Server Agent» або стандартний «Планувальник Windows» в поєднанні з утилітою SQLCMD.EXE, яка дозволяє виконувати запити до SQL Server з командного рядка [15].

Налаштування обслуговування.Повний щоденне обслуговування (в 00:00 за місцевим часом, кожен день):

- перевірка цілісності баз даних примірника MS SQL Server (всі бази даних);

- повне оновлення статистики бази даних (тільки БД магазину);

- очищення журналу примірника MS SQL Server;

- повне резервне копіювання баз даних примірника MS SQL Server (всі бази даних);

- в разі успішного завершення резервного копіювання баз даних примірника - видалення застарілих резервних копій баз даних (період зберігання - 1 тиждень) ;

- в разі успішного завершення резервного копіювання баз даних примірника - видалення застарілих файлів журналів резервного копіювання (період зберігання - 2 тижні);

- в разі успішного завершення резервного копіювання баз даних примірника - реорганізація індексів таблиць бази даних (тільки БД магазину);

- в разі успішного завершення резервного копіювання баз даних примірника і успішного завершення реорганізації індексів БД - стиснення балки транзакцій БД магазина;

- створення файлу-журналу резервного копіювання; - в разі неуспішного завершення резервного копіювання відправка листа з повідомленням про помилку адміністратору;

- копіювання файлів резервних копій на дублюючі сховище поза сервера (період зберігання - 1 тиждень);

Створення інкрементальних копій (з 6:00 по 23:30 за місцевим часом, кожні 15 хв.):

- формування інкрементальних (наростаючим підсумком) резервних копій журналу транзакцій БД магазина;

- в разі успішного завершення резервного копіювання журналу транзакцій БД магазина - видалення застарілих резервних копій журналу транзакцій (період зберігання - 24 години);

- створення файлу-журналу резервного копіювання;

- в разі неуспішного завершення резервного копіювання відправка листа з повідомленням про помилку адміністратору;

Тестове відновлення БД магазина (один раз в проміжку часу не менше 2-3 тижнів):

- відновлення БД магазина з резервних копій на тестовому екземплярі MS SQL Server. Для перевірки працездатності резервних копій.

Головна сторінка

При першому вході на сайт, всі відвідувачі потрапляють на головну сторінку сайту. Тут повинні відображатися найпопулярніші по відвідуваності продукти, що відносяться до різних категорій. Так само головна сторінка містить функціональне меню з можливістю вибору товару за категоріями і підкатегоріями. Як і більшість сучасних інтернет ресурсів, що розробляється інтернет-магазин крім звичайного пошуку повинен містити хмару тегів, що дозволяє швидко знайти потрібний товар. Головна сторінка є основним, найбільш відвідуваним ресурсом інтернет-магазину.

На головній сторінці розташовуються наступні основні блоки:

• Меню сайту (включає в себе підміню з посиланнями на головну, новини, каталог, акції, питання-відповідь і контакти);

• Каталог продукції (відображає короткий опис товару, ціну і зображення);

• Кошик (відображає список обраних користувачем товарів, кнопки видалення товарів, загальну вартість і посилання на сторінку оформлення замовлення);

Меню сайту

Меню інтернет магазину повинно відрізнятися зручністю навігації і привабливим дизайном. Головне меню складається з трьох рівнів:

• Верхній рівень складається з чотирьох елементів, що управляють: новини, особистий кабінет користувача, перегляд кошика, вхід в систему;

• Середній рівень містить категорії і підкатегорії товарів оформлених у вигляді «дерев». Так, наприклад, вибравши категорію «Електроінструмент» користувачу будуть представлені всі підкатегорії обраного типу товару.

• Нижній рівень. Тут Ви завжди маєте можливість скористатися пошуком товару. А також перейти на сторінку розширеного пошуку.

На верхньому рівні містяться переходи на найбільш важливі для користувача сторінки, якими він зможе скористатися, не чекаючи завантаження вмісту всієї сторінки. Вибір категорій продукції представлений

у вигляді меню, що випадає з використанням бібліотеки java-скриптів jQuery [2], що дозволяє досить швидко і барвисто спроектувати і створити повнофункціональний меню сайту. Так само на середньому рівні меню міститься зображення з назвою інтернет-магазину. Нижній рівень так само може використовуватися користувачем в якості швидкого пошуку, а саме без вказівки різних категорій, підкатегорій і тегів товару. Функціональне навантаження меню була розділена на рівні в зв'язку зі зручністю використання і структурою сайту.

Каталог продукції

Список товарів представляється користувачеві у вигляді таблиці, в кожному осередку відображається картка товару. Кожна картка автоматично генерується і складається з трьох полів:

- Текстовий інформаційний блок;
- Зображення товару;
- Вирішення.

Перше поле відображає назва, модель і ціну за одиницю продукції. Так користувач може ознайомитися з короткими даними про товар, не чекаючи завантаження зображення продукції. У другому полі розташовується зменшене зображення товару. Крім того натиснувши один раз на зображення товару користувач може перенести товар в область кошика і тим самим додати товар в корзину. Дана технологія неодмінно привертає попит користувачів у даного ресурсу в порівнянні з аналогами. Під зображенням товару розташовуються дві клавіші управління товаром, а саме користувач може перейти на сторінку детального перегляду товару, або додати товар, який сподобався в кошик при натисканні на кнопку з відповідною назвою.

Якщо кількість товарів не поміщається на першій сторінці каталогу товарів, користувач може переміщатися по сторінках каталогу за допомогою текстових покажчиків «вперед», «назад» і покажчиків у вигляді цифр з номерами сторінок.

Кошик

Для здійснення покупки користувачеві спочатку необхідно додати, що цікавить його товар в корзину. У даній частині проекту користувач може перейти на сторінку оформлення замовлення, переглянути список обраних товарів і загальну вартість кошика.

Список товарів містить наступні поля:

• Кількість одиниць товару;

• Зменшене зображення товару;

• Назва;

• Кнопка видалення товару;

• Вартість одиниці товару;

Дану частину проекту необхідно реалізувати з використанням технології AJAX [3]. Це означає, що після додавання товару в корзину або його видалення, сторінка не буде повністю оновлюватися, а оновитися лише вміст кошика.

Детальний перегляд товару

Детальний перегляд товару є візитною карткою кожного інтернет-магазину. Користувач, відвідуючи цей розділ, бажає ознайомитися з докладними даними про товар (опис, технічні дані, ціна і т.д.). І саме тут необхідно залучити користувача і вплинути на його вибір. Як нетривіального детального перегляду зображень товару, реалізувати привабливу зміну видів зображень.

Система аутентифікації і реєстрації користувачів Дану частину проекту необхідно зробити за допомогою форм, так як в цьому випадку розробник повністю визначає всі функції перевірки і має повний контроль над зовнішнім виглядом системи. Аутентифікація форм - привабливий вибір для розробників з кількох причин:

• Повний контроль над кодом аутентифікації.

• Повний контроль над зовнішнім виглядом форми реєстрації.

• Працює з будь-яким браузером.

• Дозволяє вибирати спосіб зберігання інформації про користувачів.

Так само крім стандартних полів введення і перевірки введеної інформації, необхідно реалізувати перевірку: чи доступний для реєстрації бажаний користувачем логін, або він вже зайнятий.

Оформлення замовлення

Для покупки товару користувачеві необхідно перейти на сторінку «оформлення замовлення», на якій він зможе в черговий раз переглянути вміст свого кошика, редагувати список обраних товарів і перейти до оплати замовлення. В якості системи оплати була обрана відома система «ROBOKASSA». ROBOKASSA - це сервіс організації платежів за послуги та покупки, який дозволяє здійснювати операції за допомогою банківських карт, систем електронних платежів і мобільних додатків. Високий ступінь захищеності, багатофункціональність і відносна простота зробили платформу дуже популярною. Її функціонування забезпечує система скриптів (модулів, написаних на програмно-сценарному мовою) - вони інтегруються з акаунтами платіжних систем, з яких списуються кошти. Клієнт за фактом отримує зручний і простий інструмент для здійснення платежів в самих різних сферах послуг.

4.2. Розгортання програмного продукту

Даний проект повинен задовольняти основним вимогам, які є актуальними в даний момент для власника інтернет-магазину.

Зручність навігації. Меню і пункти каталогу повинні відображатися таким чином, щоб у користувача не виникало труднощів, як перейти з однієї точки сайту в іншу, тобто навігація по сайту повинна бути інтуїтивно зрозумілою і зручною для користувача.

Каталог товарів. Каталог товарів також є частиною навігації сайту. Він також повинен забезпечувати зрозумілість відображення. Якщо каталог великий, то, потрібно ввести декілька можливих шляхів навігації, наприклад,

по фірмі або по моделі товару. Дуже зручно використовувати пошук по каталогу.

Кошик покупок. Важливо використовувати віртуальну кошик при навігації по каталогу, це дасть можливість відвідувачеві міг відкласти в неї товар під час перегляду каталогу товарів. Так само важливо мати можливість доступу до кошику на етапі перегляду її перед оформленням замовлення для перерахунку або видалення товарів.

Новини сайту. Самий оновлюваний розділ сайту - це новини. Новини, повинні бути, по можливості, автоматизовані для зручності адміністрування, що як результат керувати сайтом стає простіше.

Оформлення замовлення. Дуже важливо, щоб кожне замовлення обов'язково супроводжувався видачею ідентифікатора замовлення. Реєстрація покупців. Плюси в реєстрації дає відвідувачеві багато можливостей (надання історії покупок, адрес доставки і т.д.), але є і суттєві інші мінуси, відвідувачеві потрібно заповнювати численні форми, вводити свої дані і взагалі проходити реєстрацію, а це люблять далеко не всі.Таким чином, було вирішено відкрити можливість пошуку, перегляду та додавання в корзину товару всім категоріям користувачів.

Можна виділити два типи користувачів: звичайні відвідувачі і користувачі системи. Звичайні відвідувачі, запитують мала кількість сторінок і залишаються на них протягом короткого проміжку часу. Такі відвідувачі найменше зацікавлені в інформаційному наповненні сайту. Але їх можна зацікавити зображеннями, дизайном, різними елементами управління і персоналізованих підходом.

Користувачі запитують більше число сторінок і залишаються на сайті протягом більшої кількості часу, ніж відвідувачі. Мають меншим інтересом до зображень і дизайну, їх більше цікавить вміст сайту. Можуть завести собі обліковий запис на сайті і використовувати переваги персональної панелі. Для успішної комерціалізації проекту важливий будь-який користувач. Важливо вміти зацікавити кожного. Найбільш ефективний спосіб цього

досягти - створити привабливий інтерфейс користувача. Коли користувача привертає інтерфейс, він витрачає більше часу на перегляд сайту, тим самим ростуть шанси його зацікавити.

В даний час все більше і більше людей користуються інтернетом - це призвело до того, що стало з'являтися багато різних браузерів (основний інструмент користувача інтернетом). Поява великої кількості браузерів привернуло за собою проблеми сумісності. Хоча, кожен з розробників браузерів і вважає себе найкращим, але, тим не менш, єдиного стандарту для всіх немає.Дуже часто можна зустріти сайт, вид сторінок якого буде кардинально відрізнятися в різних браузерах. Ці проблеми призвели до виникнення стандарту W3C [1].

У цьому стандарті міститися тільки ті інструменти, які підтримуються всіма браузерам (більшістю). Грунтуючись на статистиці (на момент написання цієї пояснювальної записки), більшість користувачів інтернету використовують такі браузери:

За даними OpenStat:

Chrome - 43.96%

Firefox - 10.55%

Opera - 9.11%

Safari - 8.05%

Microsoft Internet Explorer - 5.71%

За даними LiveInternet :

Google Chrome - 46.38%

Safari 1 - 12.68%

Android Browser - 10.23%

Firefox - 5.79%

Розробка проекту орієнтувалася на браузери Google Chrome, Firefox, Opera. Таким чином в цих браузерах дизайн відображається однаково, і вся функціональність працює коректно.

Сторінка «Реєстрації» являє собою одну загальну панель, на якій

розташовані питання і поля для введення відповідей. Для перевірки коректності введених даних використовуються пояснюючі написи, які вказують на тип помилки і що повинен користувач виправити при відповіді на питання.

Передбачена кнопка перевірки облікового запису (логін) на зайнятість. Якщо користувач ввів логін, який вже присутній в системі, то на формі з'явиться повідомлення про те, що даний Login можна використовувати для реєстрації. Використовується валідатори для підтвердження коректності введення даних. Валідатори - це стандартні елементи управління в ASP.NET, які представляються користувачеві у вигляді написів, що свідчать про неправильне або некоректному введенні даних на сторінці. Існує кілька типів валідаторів, деякі з них містять у собі регулярний вираз або правило, за яким визначається правильність або достовірність введених даних.

Для кожного користувача додатки створюється своя власна сесія зі своїми власними значеннями. Для ідентифікації користувачів ASP.NET використовує 120-бітний ключ, іменований SessionID і складається тільки з ASCII-символів, які допустимі для використання в URL. Під час отримання запиту від користувача, який не має SessionID в URL запиту, ініціюється створення нової сесії. При цьому відбувається генерація нового унікального SessionID. Після чого виникає подія Session_Start веб-додатки.

Сесія існує до тих пір, поки користувач працює з веб-додатком і ще трохи після. Для цього в настройках аплікації виставляється таймаут сесії, який за замовчуванням становить 20 хвилин. Тобто якщо користувач протягом 20 хвилин не вчинив жодного запиту до додатка, то сесія цього користувача знищується. При цьому виникає подія Session_End додатки. У поточному проекті час сесії була укорочена до 5 хвилин з міркувань безпеки. В реалізованої підсистемі сесія використовується для зберігання об'єкта класу User, в якому зберігаються дані про поточного користувача системи.

При спробі входу в систему, створюється запит на вибірку з бази даних користувача із зазначеним логіном і паролем, далі відбувається перевірка на

існування запису про таке користувача, а саме перевіряється, чи існує id у обраного користувача. Так само сесія використовується для користувачів вже пройшли аутентифікацію. Використовуючи дані про користувача, формується запит до бази даних про наявність товару, раніше покладеного в кошик користувачем. При виявленні таких записів користувач може побачити свою корзину вже заповнену вибраним раніше товаром, що дозволить заощадити час на пошук товару.

Аутентифікація форм - привабливий вибір для розробників з кількох причин:

1 Повний контроль над кодом аутентифікації.

2 Повний контроль над зовнішнім виглядом форми реєстрації.

3 Працює з будь-яким браузером.

4 Дозволяє вибирати спосіб зберігання інформації про користувачів.

Оскільки аутентифікація форм реалізована повністю всередині ASP.NET, розробник отримує повний контроль над виконанням аутентифікації. Немає необхідності покладатися ні на яку-небудь зовнішню систему, як це має місце при аутентифікації Windows або Passport. Так само з'являється можливість самостійно налаштувати поведінку аутентифікації форми під свої потреби. Іншими словами, розробнику надається можливість оформляти вхідну сторінку реєстрації як завгодно. Гнучкість зовнішнього вигляду недоступна при інших методах аутентифікації. Аутентифікація Windows вимагає, щоб браузер запитував призначене для користувача посвідчення, а Passport-аутентифікація вимагає, щоб ви залишили свій Web-сайт і відвідали сайт Passport для введення своєї реєстраційної інформації (посвідчення) [4].

При реалізації проекту важливу роль відіграє грамотне складання структури бази даних проекту [15]. При створенні таблиць бази даних були враховані основні положення тематики проекту, а саме інтернет-магазину. При роботі з проектом всі користувачі будуть розділені на 3 основні групи, такі як зареєстровані, незареєстровані користувачі і адміністратори сайту.

Кожній групі користувачів відповідають певні права: можливість оформлення замовлення, можливість редагування каталогів товарів, можливість додавання тегів до товару і т.д. Для зберігання інформації про пропонований товар використовується таблиця «dbo.Product». В даній таблиці зберігається повний опис товару, ціна, категорія, а так само кількість штук.

Щоб спростити взаємодію користувача з кошиком, було вирішено вивести окрему таблицю dbo.Basket, в якій буде зберігатися інформація про товари, доданих до кошика. Необхідно розуміти, що користувач може з яких-небудь технічних причин покинути сайт, але при наступному відвідуванні інтернет-магазину, йому доведеться знову шукати, вибирати і додавати вже вибрані раніше товари. Щоб уникнути такої ситуації нам слід при черговому вході користувач, лише згідно з таблицею його кошик і в разі виявлення додавалися товарів, просто відобразити кошик користувача.

Наступна таблиця - таблиця замовлень «dbo.Order» необхідна для операцій безпосереднього продажу і доставки товару користувачеві. Після опису ключових моментів проектування бази даних, необхідна реалізація бази даних проекту.

Для реалізації спроектованої бази даних була обрана система управління базами даних MS SQL Server 2008. Це обумовлено тим, що, дана СУБД має більшу функціональність, безліч засобів для підтримки і роботи з нею, розвинену інфраструктуру для інтеграції баз даних в призначені для користувача програми.

У створюваній базі даних будуть використовуватися такі типи даних:

- INT - цілочисельний тип. Розмір - 4 байта
- NVARCHAR - Строковий тип
- BIT - Бітовий тип. Використовується як логічний тип
- DATE - Тип, який визначає дату
- DATETIME - Тип, який визначає дату і час
- FLOAT - Речовий тип

- MONEY - Грошовий тип
- REAL - Речовий тип

Опишемо всі таблиці, які будуть створені в базі даних.

Таблиця Аксесуари містить назву аксесуару. Її структура приведена в таблиці 3.1.

Таблиця 3.1 - Характеристика атрибутів таблиці Аксесуари

Ім'я атрибута	**Тип**	**Опис**
ID_Аксесуари	INT	Ідентифікатор аксесуару
Назва	NVARCHAR(50)	Назва аксесуару
ID_тип_аксесуарів	INT	Ідентифікатор типу аксесуару

Таблиця *Повернення* содержит інформацію про повернення товару, який був бракований і причини повернення. Також дату повернення. Її структура приведена в таблиці 3.2.

Таблиця 3.2 – Характеристика атрибутів таблиці Повернення

Ім'я атрибута	**Тип**	**Опис**
Number	INT	Ідентифікатор повернення
Причина	NVARCHAR(50)	Причина
IDПродажа	INT	Ідентифікатор продажі
Дата_ повернення	DATE	Дата повернення

Таблиця *Гарантія* містить інформацію про гарантії, а саме: примітка і дату закінчення і дату початку. Її структура приведена в таблиці 3.3.

Таблиця 3.3 – Характеристика атрибутів таблиці Гарантія

Ім’я атрибута	Тип	Опис
Number	INT	Ідентифікатор гарантії
Дата_закінчення	DATE	Дата закінчення
Примітки	NVARCHAR (50)	Примітки
IDПродажа	INT	Ідентифікатор продажі
Дата_начало	DATE	Дата продажі

Таблиця *Посади* служить для показу до якого типу належить товар. Її структура приведена в таблиці 3.4.

Таблиця 3.4 – Характеристика атрибутів таблиці Посада

Ім’я атрибута	Тип	Опис
ID	INT	Ідентифікатор посади
Посада	NVARCHAR (50)	Посада співробітника

Таблиця Каса служить для обліку продажів. Її структура приведена в таблиці 3.5.

Таблиця 3.5 - Характеристика атрибутів таблиці Каса

Ім'я атрибута	Тип	Опис
ID	INT	Ідентифікатор каси
Сума	MONEY	Сума каси
Дата_і_час	DATETIME	Дата і час покупки

Таблиця Клієнти містить інформацію про покупців. Її структура приведена в таблиці 3.6.

Таблиця 3.6 - Характеристика атрибутів таблиці Клієнти

Ім'я атрибута	**Тип**	**Опис**
ID	INT	Ідентифікатор клиента
ПІП	NVARCHAR (50)	Прізвище, Імя, По батькові клієнта
Телефон	NVARCHAR (20)	Телефон клієнта

Таблиця Оплата для обліку покупки товару, зазначення дати продажу, суми проданого товару. Її структура приведена в таблиці 3.7.

Таблиця 3.7 - Характеристика атрибутів таблиці Оплата

Ім'я атрибута	**Тип**	**Опис**
ID	INT	Ідентифікатор оплати
Дата продажі	DATE	Дата продажі
IDКаса	INT	Ідентифікатор каси
Сума	MONEY	Сума оплати
IDКлієнта	INT	Ідентифікатор клієнта
ID_співробітника	INT	Ідентифікатор співробітника

Таблиця Список_товарів служить для обліку ціни, назви товару. Її структура приведена в таблиці 3.8.

Таблиця 3.8 - Характеристика атрибутів таблиці Список_товарів

Ім'я атрибута	**Тип**	**Опис**

ID_товара	INT	Ідентифікатор товару
Ціна	MONEY	Ціна товару
Назва_товару	NVARCHAR(50)	Назва товару
ID_тип_товару	INT	Ідентифікатор типу товару

Таблиця Поставки служить для зберігання даних про постачання, датою поставки, ціні поставки і надбавки. Її структура приведена в таблиці 3.9.

Таблиця 3.9 - Характеристика атрибутів таблиці Поставки

Ім'я атрибута	**Тип**	**Опис**
ID	INT	Ідентифікатор товару
Дата_поставки	DATE	Ціна товару
ID_постачальника	INT	Назва товару
ID_співробітника	INT	Ідентифікатор типу товару
Ціна	MONEY	Сума
Торгова_надбавка	REAL	Торгова надбавка
IDТовару	INT	Ідентифікатор товару
ID_Тип_доставки	INT	Ідентифікатор типу доставки

Таблиця Постачальники служить для зберігання даних про постачальників, а саме: про країну, адресу, телефон та найменування товару. Її структура приведена в таблиці 3.10.

Таблиця 3.10 - Характеристика атрибутів таблиці Постачальники

Ім’я атрибута	Тип	Опис
ID	INT	Ідентифікатор товару
Країна	NVARCHAR(50)	Ціна товара
Адреса	NVARCHAR(200)	Назва товара
Телефон	NVARCHAR(25)	Ідентифікатор типа товара
Назва	NVARCHAR(150)	Сума

Таблиця Надходження Товарів служить для зберігання даних про постачальників, а саме: про країну, адресу, телефон та найменування товару. Її структура приведена в таблиці 3.11.

Таблиця 3.11 - Характеристика атрибутів таблиці Надходження Товарів

Ім’я атрибута	Тип	Опис
ID	INT	Ідентифікатор надходження товара
IDНадходження	INT	Ідентифікатор надходження
Партія	INT	Партія надходження
Кількість	INT	Кількість надходження
Гарантія_з	DATE	Гарантія з
Гарантія_по	DATE	Гарантія по
IDтовар	INT	Ідентифікатор товара
Дата надходження	DATE	Дата надходження
IDСотрудника	INT	Ідентифікатор співробітника

Таблиця Продажі містить інформацію про продані товари. Її структура приведена в таблиці 3.12.

Таблиця 3.12 - Характеристика атрибутів таблиці Продажі

Ім'я атрибута	Тип	Опис
IDПродажі	INT	Ідентифікатор продажі
Кількість	INT	Кількість
IDТовара	INT	Ідентифікатор товара
IDОплата	INT	Ідентифікатор оплати

Таблиця Співробітники містить інформацію про співробітників, а саме: його освіта, стать, П.І.П. і дата народження. Її структура приведена в таблиці 3.13.

Таблиця 3.13 - Характеристика атрибутів таблиці Співробітники

Ім'я атрибута	Тип	Опис
ID_	INT	Ідентифікатор співробітника
ID_посади	INT	Ідентифікатор посади
ПІП	NVARCHAR(50)	Прізвище, Імя, По батькові співробітника
Дата_народження	DATE	Дата народження
Стать	BIT	Стать співробітника
Дата_прийому_на_роботу	DATE	Дата прийому співробітника на работу
Освіта	NVARCHAR(50)	Освіта співробітника

Таблиця Тип_аксесуару містить інформацію про тип аксесуару. Її структура приведена в таблиці 3.14.

Таблиця 3.14 - Характеристика атрибутів таблиці Тип_аксесуару

Ім'я атрибута	Тип	Опис

ID_тип_Аксесуари	INT	Ідентифікатор типу аксесуарів
Тип_аксесуари	NVARCHAR(50)	Типи аксесуарів

Таблиця Тип_доставки містить інформацію про тип доставки. Її структура приведена в таблиці 3.15.

Таблиця 3.15 - Характеристика атрибутів таблиці Тип_доставки

Ім'я атрибута	**Тип**	**Опис**
ID_Тип_доставки	INT	Ідентифікатор типа доставки
Тип_доставки	NVARCHAR(50)	Типи доставки

Таблиця Тип_товара містить інформацію про тип товару. Її структура приведена в таблиці 3.16.

Таблиця 3.16 - Характеристика атрибутів таблиці Тип_товара

Ім'я атрибута	**Тип**	**Опис**
ID_тип_товара	INT	Ідентифікатор тип товара
Тип_товара	NVARCHAR(50)	Тип товара

У спроектованої бази даних реалізований контроль цілісності і коректності модифікацій, шляхом виконання тригерів і накладення обмежень на поля. Так для вилучення з каси суми, необхідної для повернення клієнту за бракований товар, використовується тригер MinusReturn в таблиці Повернення, який при вставці нового запису, автоматично вставляє в таблицю Каса, запис з негативною вартістю бракованого товару. Далі наводиться реалізація даного тригера:

```
CREATE TRIGGER [dbo]. [MinusReturn]
 ON [dbo]. [Повернення]
 AFTER INSERT
AS
BEGIN
SET NOCOUNT ON;
DECLARE @val money
SET @val = (SELECT Список_товаров.Ціна
FROM Повернення INNER JOIN
 Продажі ON Возврат.IDПродажа = Продажі.IDПродажі INNER JOIN
 Перечень_товаров ON Продажі.IDТовара = Список_товарів.ID_товара
WHERE Повернення.Number IN (SELECT Number FROM inserted))
DECLARE @ide int
SET @ide = (SELECT MAX (ID) FROM dbo.Касса)
INSERT dbo.Касса VALUES (@ ide + 1, - @ val, GETDATE ())
END
```

Для визначення перевірки умови на вік співробітника було створено обмеження, обчислює різницю між датою народження і датою прийняття на роботу. Текст даного обмеження наведено далі:

```
ALTER TABLE [dbo]. [Співробітники] WITH CHECK ADD CONSTRAINT [CK_Співробітники] CHECK ((datediff (year, [Дата_народження], [Дата_прийому_на_работу])> (16)))
```

Microsoft SQL Server в якості мови запитів використовує версію SQL, що отримала назву Transact-SQL [9] (скорочено T-SQL), що є реалізацією SQL-92 (стандарт ISO для SQL) з множинними розширеннями. T-SQL дозволяє використовувати додатковий синтаксис для збережених процедур і забезпечує підтримку транзакцій (взаємодія бази даних з керуючим додатком). Microsoft SQL Server і Sybase ASE для взаємодії з мережею використовують протокол рівня додатка під назвою Tabular Data Stream (TDS, протокол передачі табличних даних).

Протокол TDS також був реалізований в проекті FreeTDS з метою забезпечити різним додаткам можливість взаємодії з базами даних Microsoft SQL Server і Sybase. Microsoft SQL Server також підтримує Open Database Connectivity (ODBC) - інтерфейс взаємодії додатків з СУБД. Версія SQL Server 2014 забезпечує можливість підключення користувачів через веб-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 додаткам під керуванням Java (таким як BEA і IBM WebSphere) з'єднуватися з Microsoft SQL Server 2014. SQL Server підтримує віддзеркалення і кластеризації баз даних.

Кластер сервера SQL - це сукупність однаково конфігурованих серверів; така схема допомагає розподілити робоче навантаження між декількома серверами. Всі сервера мають одне віртуальне ім'я, і дані розподіляються по IP-адресами машин кластеру протягом робочого циклу.Також в разі відмови або збою на одному з серверів кластера доступний автоматичний перенос навантаження на інший сервер. У SQL Server 2014 вбудована підтримка .NET Framework. Завдяки цьому, збережені процедури БД можуть бути написані на будь-якій мові платформи .NET, використовуючи повний набір бібліотек, доступних для .NET Framework, включаючи Common Type System (система поводження з типами даних в Microsoft .NET Framework).

Однак, на відміну від інших процесів, .NET Framework, будучи базисною системою для SQL Server 2014 року, виділяє додаткову пам'ять і вибудовує засоби управління SQL Server замість того, щоб використовувати вбудовані засоби Windows. Це підвищує продуктивність в порівнянні з загальними алгоритмами Windows, так як алгоритми розподілу ресурсів спеціально налаштовані для використання в структурах SQL Server.

Коли існуючих можливостей ставати мало, а вдосконалювати існуюче вже нікуди, тоді і відбувається технологічний прорив. Таким проривом і є

AJAX (Asynchronous JavaScript and XML) - підхід до побудови призначених для користувача інтерфейсів веб-додатків, при якому web-сторінка, що не перезавантажуючись, сама довантажує потрібні користувачу дані. AJAX - один з компонентів концепції DHTML [10]. Що ж дає нам ця технологія.

В даний час розробка WEB додатків прагне до розмежування клієнтської частини і серверної. При розробці складних проектів виникає необхідність в структурованості і легкості читання коду. Не слід засмічувати код програміста кодом верстальника, а код верстальника - правками дизайнера, і так далі.Виникає необхідність в розмежуванні роботи. Так, наприклад, дизайнер буде робити свою роботу, верстальник свою, програміст свою, і при цьому ніхто один одному заважати не буде. У підсумку кожному учаснику проекту досить буде знати тільки ті дані, з якими йому доведеться працювати. В такому випадку продуктивність групи і якість проекту підвищується в рази.

Отже, AJAX - це ідея, яка базується на двох основних принципах:

• Використання DHTML для динамічної зміни змісту сторінки.

• Використання XMLHttpRequest для звернення до сервера "на льоту".

Використання цих двох підходів дозволяє створювати набагато зручніші WEB-інтерфейси користувача на тих сторінках сайтів, де необхідна активна взаємодія з користувачем. Використання Ajax стало найбільш популярне після того, як компанія Google почала активно використовувати його при створенні своїх сайтів, таких як Gmail, Google maps і Google suggest. Створення цих сайтів підтвердило ефективність використання даного підходу.

Клієнт, набираючи в рядку пошуку адресу цікавить його ресурсу, потрапляючи на сервер, робить до нього запит. Сервер виробляє обчислення за результатами запитів, звертається до бази даних і так далі, після чого отримані дані йдуть клієнтові і, в разі необхідності підставляються в шаблони і обробляються браузером. Результатом є сторінка, яку ми бачимо, і яку 80% населення країни, що знаходиться в WEB називають Інтернетом. Це

класична модель, яка встигла себе зарекомендувати і заслужити собі почесне місце під сонцем. Це найпростіша модель взаємодії і, як наслідок, найпоширеніша.

При зверненні до сервера, генерується сторінка, яка буде відображатися користувачеві, і пропонувати йому зробити потрібну йому послідовність дій.При свідомому (хоча і не обов'язково) виборі клієнта, його запит буде звертатися до AJAX модулю, який і буде виробляти всі потрібні йому обчислення і роботу з сервером як таким.

Основна відмінність в тому що цей метод дає нам можливість динамічно звертатися до сервера і виконувати цікаві для нас дії. Наприклад, нам потрібно виконати звернення до бази даних і отримати цікаві для нас дані, які потім будемо використовувати.

Бібліотека jQuery jQuery - це популярна javascript бібліотека, здатна істотно спростити життя веб-розробнику. Бібліотека jQuery містить функціонал, корисний для максимально широкого кола завдань. Проте, розробниками бібліотеки не ставилося завдання суміщення в jQuery функцій, які підійшли б усюди, оскільки це призвело б до великого коду, велика частина якого не затребувана. Тому була реалізована архітектура компактного універсального ядра бібліотеки і плагінів. Це дозволяє зібрати для ресурсу саме той JavaScript-функціонал, який на ньому був би затребуваний [11].

Бібліотека jQuery в першу чергу забезпечує несуперечливу роботу програмного коду у всіх типах браузерів, вирішуючи такі складні проблеми JavaScript, як очікування завантаження сторінки перед тим, як виконувати будь-які операції. На той випадок, якщо в бібліотеці виявитися недолік функціональності, розробники передбачили простий, але дуже дієвий спосіб її розширення. Багато починаючі програмісти jQuery виявляють цю гнучкість на практиці, розширюючи можливості jQuery в перший же день.Щоб привнести динамічну функціональність на будь-яку інтернет сторінку, доводитися слідувати одному і тому ж шаблону: спочатку відбирається

елемент або група елементів, а потім над ними виконуються деякі дії, наприклад, приховувати або показувати, що цікавлять нас елементи, додавати до них клас CSS, створювати анімаційні ефекти або змінювати атрибути. З звичайним JavaScript для вирішення кожного із завдань буде потрібно десятки рядків програмного коду. Творець jQuery розробив свою бібліотеку саме для того, щоб зробити найбільш загальні завдання тривіальними. Наприклад, щоб створити таблицю з різним кольором фону для парних і непарних рядків, дизайнеру потрібно написати до 10 рядків коду на мові JavaScript, а ось з використанням jQuery цей ефект досягається з використанням не більше ніж одного рядка.

ВИСНОВКИ

Результатом роботи є створений веб додаток для управління товарами і заказами товарів. 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 реалізація бази даних SQL Server 2016.

Для тестування додатку 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. ADO.net on-line tuturial
2. Hawryszkiewych I.T. Introduction to system analysis and design. New York, 1992. 379 p.
3. Martin Fowler — GUI Architectures. Часть 2 [Електронний ресурс]. — 2009 — Режим доступу: https://habr.com/post/53536/.
4. MS JDBC SQL Server driver documentation
5. SQL Native Client documentation on MSDN
6. Transact-SQL Reference on MSDN
7. W3C Standards HTML & CSS [Электронный ресурс]. — Режим доступу: https://www.w3.org/standards/webdesign/htmlcss
8. Аткинсон Л. PHP 5. Библиотека профессионала / Л. Аткинсон, З. Сураски – М: Вильямс, 2006 – 944с.
9. Бадд Э. CSS. Профессиональное применение Web-стандартов / Э. Бадд, К. Молл, С. Коллизон – М: Вильямс, 2009 – 272с.
10. 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.
11. Болье А. Изучаем SQL – М: Символ-Плюс, 2007 – 309с.
12. В. Дунаев HTML, скрипты и стили – СПб.: БХВ-Петербург, 2015 – 816с.
13. Вайк А. JavaScript. Энциклопедия пользователя / А. Вайк, Р. Вагнер – М: ДиаСофт, 2001 – 480с.
14. Вейл Э. HTML5. Разработка приложений для мобильных устройств – СПб.: Питер, 2015 – 480с.
15. Веллинг Л. Разработка веб-приложений с помощью PHP и MySQL / Л. Веллинг, Л. Томсон – М: Вильямс, 2016 – 768с.

16. Вендров А.М. Проектирование программного обеспечения экономических информационных систем: Учебник. – 2-е изд., перераб. и доп.– М.: Финансы и статистика, 2006. – 544 с: ил.
17. Грант К. CSS для профи – СПб.: Питер, 2016. – 496с.
18. 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.
19. Золотов С.Ю. Проектирование информационных систем: Учебное методическое пособие. – Томск: ТМЦДО, 2006 – 34.
20. Кириченко А. Динамические сайты на HTML, CSS, JavaScript и Bootstrap. Практика, практика и только практика / А. Кириченко, Е. Дубовик – СПб.: Наука и техника, 2016 – 272с.
21. Колисниченко Д. PHP и MySQL. Разработка Web-приложений – СПб.: БХВ-Петербург, 2016. – 640с.
22. Котеров Д. В. PHP 7 / Д. В. Котеров, И. В. Симдянов. – СПб.: БХВ-Петербург, 2016. – 1066с.
23. Куссуль Н. Использование PHP. Самоучитель / Н. Куссуль, А. Шелестов – М: Вильямс, 2006 – 272с.
24. Маккоу А. Веб-приложения на JavaScript– СПб.: Питер, 2012 – 288с.
25. Макфарланд Д. Новая большая книга CSS – СПб.: Питер, 2016. – 720с.
26. Мейер Э. CSS. Карманный справочник – М: Вильямс, 2016 – 288с.
27. Морето С. Bootstrap в примерах – М: ДМК Пресс, 2016 – 314с.
28. Никсон Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5 - СПб.: Питер, 2016. – 816с.
29. Основы системного анализа и проектирования АСУ./ под. ред.А.А.Павлова. -К: Вища школа, 1991.
30. Офіційний сайт ТНЕУ [Електронний ресурс]. – Режим доступу

http://www.tneu.edu.ua/

31. Перегудов Ф.И, Тарасенко Ф.П. Введение в системный анализ. М: Высшая школа, 1992.
32. Побудова діаграми варіантів використання [Електронний ресурс],– Режим доступу: http://moodle.ipo.kpi.ua/
33. Сергеев А. HTML 4.0 / А. Сергеев, А. Матросов, М. Чаунин – СПб.: БХВ-Петербург, 2008. – 672с.
34. Системный анализ и структуры управления./ под ред. В.Г.Щорина - М.: Знание, 1975.
35. Скляр Д. PHP. Рецепты программирования / Д. Скляр, А. Трахтенберг – СПб.: Питер, 2015 – 784с.
36. Соколов С. CSS3 в примерах. Профессиональная работа – М: Вильямс, 2008 – 352с.
37. Уэнц К. PHP. Карманный справочник. Необходимый код и команды – М: Вильямс, 2007 – 384с.
38. Флэнаган Д. JavaScript. Подробное руководство – М.:Символ-Плюс, 2012 – 1080с.
39. Хольцшлаг М. Использование HTML и XHTML. Специальное издание – М: Вильямс, 2004 – 736с.
40. Шарапов О.Д., Терехов Л.М., Сіднєв С.П. Системний аналіз. К.: Вища школа, 1983.

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="ЇЙЇЙ" 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 />
               &nbs
p;   
<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 відвідування], Description AS [Опис відвідування]
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 відвідування" HeaderText="Тип відвідування" SortExpression="Тип
відвідування" />
<asp:BoundField DataField="Опис відвідування" HeaderText="Опис відвідування" SortExpression="Опис
відвідування" />
<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];
}
}
}
```