Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Розробка системи бронювання квитків з використанням технологій .Net та MySQL

Виконав(ла): студент(ка) 6 курсу, групи СПм-61
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)

	(підпис)	Геник М.І. (прізвище та ініціали)
Керівник	(підпис)	Стоянов Ю.М. (прізвище та ініціали)
Нормоконтроль	(підпис)	Стоянов Ю.М. (прізвище та ініціали)
Завідувач кафедри	(підпис)	Петрик М.Р. (прізвище та ініціали)
Рецензент	(підпис)	(прізвище та ініціали)

Тернопіль
2023

РЕФЕРАТ

Атестаційна робота магістра. Тернопільський національний технічний університет імені Івана Пулюя, кафедра програмної інженерії, спеціальність 121 «Інженерія програмного забезпечення». ТНТУ, 2023. Сторінок 39, таблиць 3, рисунків 21, презентація

Тема: Розробка системи бронювання квитків з використанням технологій .Net та MySQL.

В атестаційній роботі магістра проведено розробку програмного забезпечення системи бронювання квитків. В процесі розробки було проведено аналіз галузі, виявлено доцільність та сформульовано вимоги. Виявлено варіанти використання системи та побудовано бізнес-модель. Сформульовано загальну архітектуру системи. Побудовано базу даних на основі системи керування базами даних MySQL. Створено графічний інтерфейс до бази даних на платформі Windows Forms фреймворку .NET мовою C#. Успішно проведено перевірку на надійність та працездатність програмного забезпечення.

.NET, БРОНЮВАННЯ КВИТКІВ, WINFORMS, C#, MYSQL, СКБД.

ABSTRACT

Master's certification work. Ternopil Ivan Puluj National Technical University, Department of Software Engineering, specialty 121 "Software Engineering". TNTU, 2023. Pages 39, tables 3, figures 21, presentation

Topic: Development of the system of ticket booking using the MySQL and .NET technologies.

In the Master's certification work, the development of software for the ticket reservation system was carried out. During the development process, industry analysis was conducted, expediency was identified, and requirements were formulated. Options for using the system were identified and a business model was built. The general architecture of the system is formulated. A database was built based on the MySQL database management system. Created a graphical interface to the database on the Windows Forms platform of the .NET Framework in C#. The software has been successfully tested for reliability and functionality.

.NET, TICKET BOOKING, WINFORMS, C#, MYSQL, DBMS.

ЗМІСТ

ВСТУП

Останніми десятиліттями досить консервативна галузь транспорту стала інтенсивно автоматизовуватися, що дозволило пришвидшити перевезення та оформлення пасажирів та вантажів. Одним з основних напрямів автоматизації є бронювання квитків.

Внаслідок створення нових технологій та платформ як програмування, так і взаємодії та керування даними станом на сьогоднішній день стало можливо створити сучасну систему для бронювання квитків, яка при цьому повинна також бути надійною, швидкою та ефективною. Водночас сучасні технології та методики програмної інженерії також дозволяють зробити код самої системи достатньо простим та зрозумілим сторонньому розробнику.

Метою атестаційної роботи магістра є створення якісного необхідного програмного забезпечення, що повинне спростити бронювання квитку. Таким чином, однією з умов до цієї програми буде інтуїтивно зрозумілий інтерфейс.

Програма повинна давати можливість користувачу додавати та редагувати інформацію про клієнтів, їхню інформацію для ідентифікації, квитки, які вони купують, в тому числі місце відправки та призначення, їхню ціну задля змоги повернення, місця в потягах, а також збереження та видалення інформації в базах.

В ході розробки було виконано проєктування та створення програмної системи бронювання квитків. Як засоби візуальної розробки були використані інтегровані середовища розробки Microsoft Visual Studio 2019 та MySQL Workbench версії 8.0.34. Серверну частину виконано на базі MySQL Server версії 8.0.28, програмну ж частину створено на базі інтерфейсу програмування додатків Windows Forms платформи .NET Framework версії 4.8.

Створене програмне забезпечення дозволить зменшити час та зусилля, необхідні для оформлення квитка та розвантажити роботу касира, а відповідно матиме значний економічний ефект.

1. ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ТРАНСПОРТНОЇ ГАЛУЗІ

1.1 Огляд конкурентних систем

Бронювання квитків за допомогою комп'ютера вже деякий час застосовується деякими підприємствами, тому є сенс оглянути наявні конкурентні системи. Конкурентами даної системи є онлайн-системи бронювання квитків, нижче наведено деякі із них.

Booking.uz.gov.ua

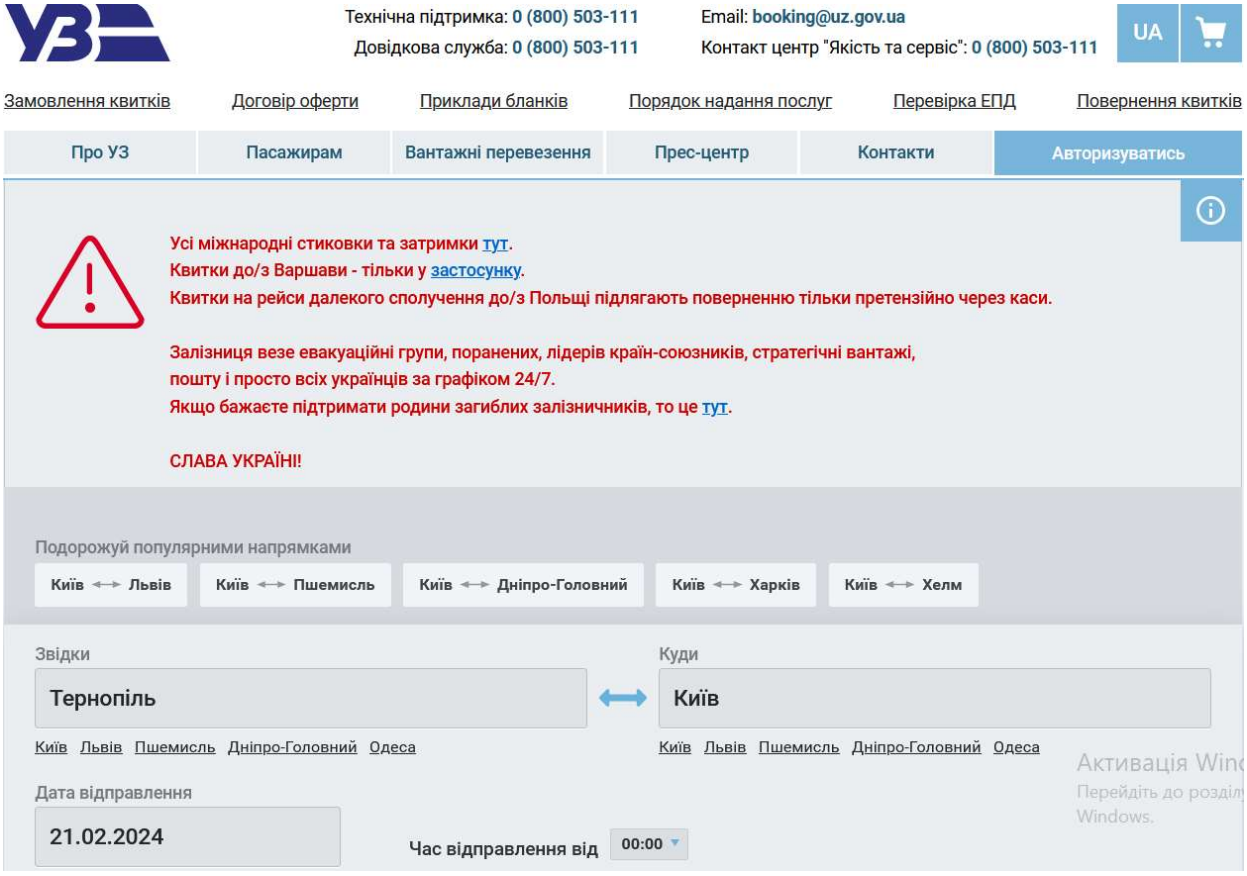


Рисунок 1.1. Інтерфейс пошуку потяга Укрзалізниці (квитків немає)

Одним з найпоширеніших сервісів бронювання квитків є сервіс бронювання квитків від Укрзалізниці. Спершу варто ввести потрібний маршрут та дату,

внаслідок чого сайт видасть список потягів. Після обрання потрібного потяга сайт показує кількість вільних місць та їхнє розташування в вагонах. Після обрання потрібних місць розпочинається оформлення купівлі. Можна обрати опції, як от чаї та обіди в потязі. Оплата здійснюється через LiqPay. Позитивом цієї системи є те, що сайт генерує одразу електронну версію квитка, яка дійсна навіть якщо показати її з телефону, а також можливістю купити окремо студентські квитки (тимчасово не дійсна).

Poizdato.net

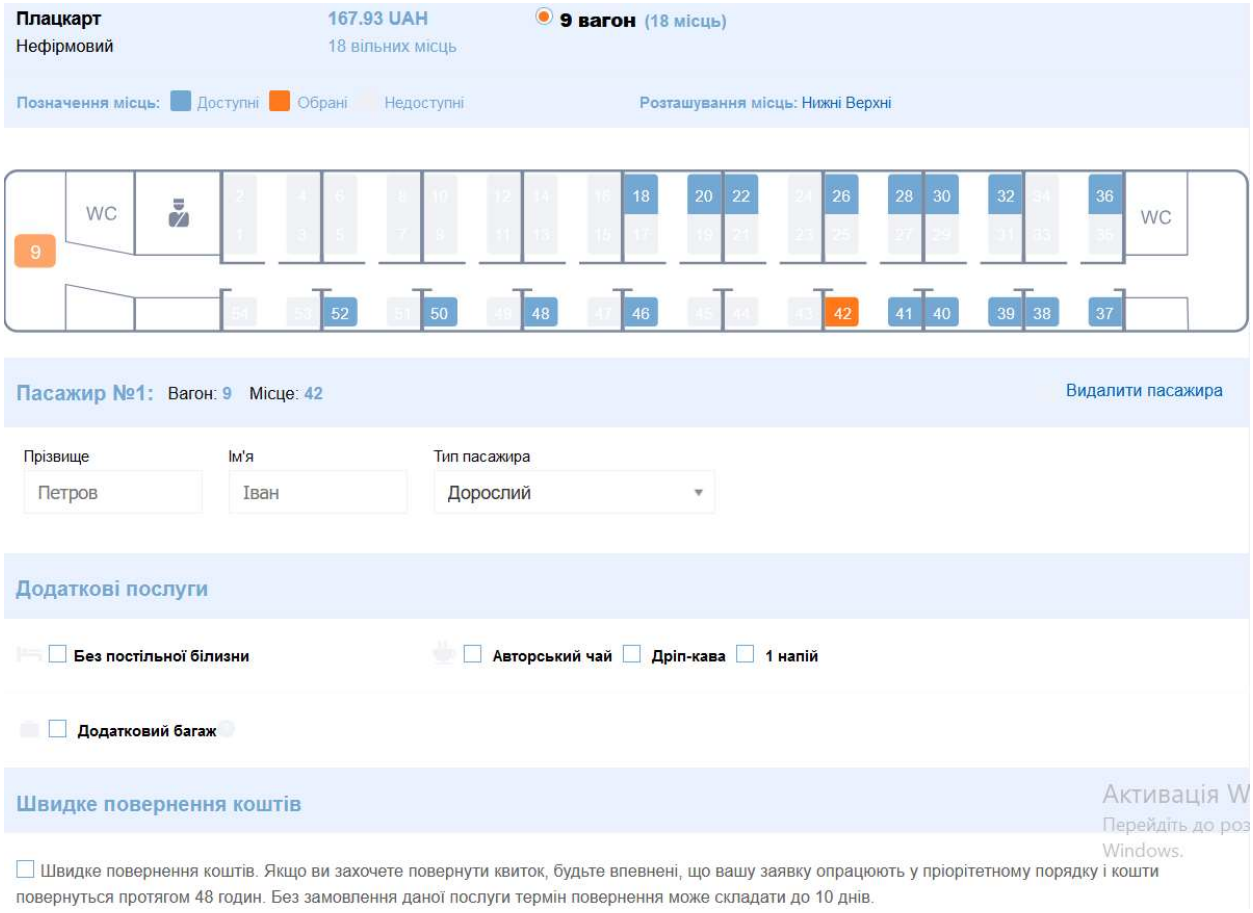


Рисунок 1.2. Інтерфейс купівлі квитків на Poizdato.net

Іншим поширеним сервісом бронювання квитків є Poizdato.net, хоч цей сайт більше відомий своїм онлайн-розкладом потягів. Загалом інтерфейс подібний до booking.uz.gov.ua, проте більш пристосований до людського ока; імовірно тут

використовується такий самий API. Втім я ніколи повноцінно не користувався цим сайтом, тому об’єктивно оцінити його не можу.

BlaBlaCar

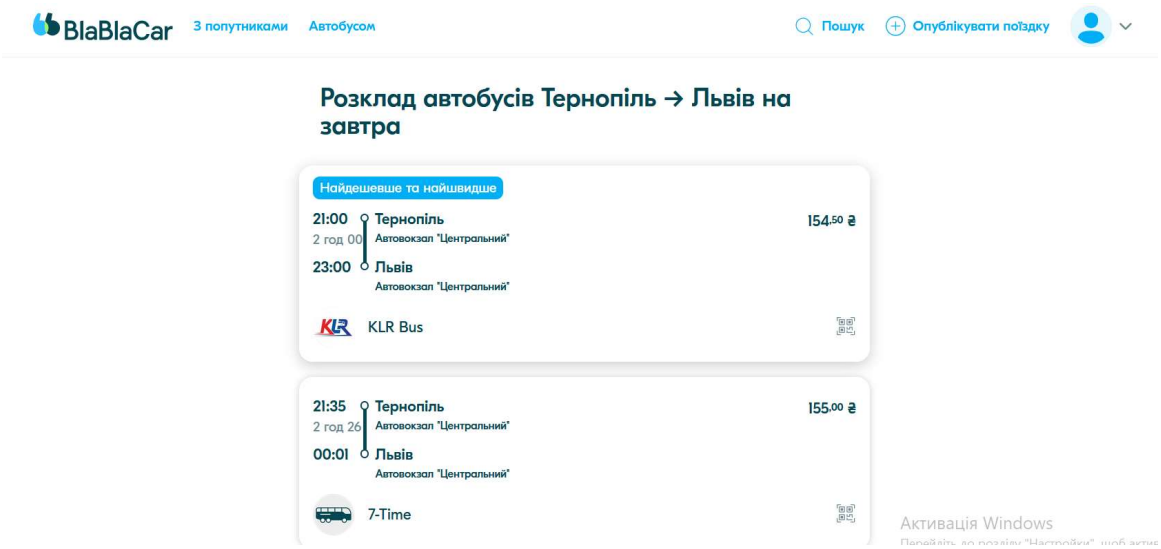


Рисунок 1.3. Інтерфейс купівлі квитків у BlaBlaCar

BlaBlaCar перш за все відомий як платформа для пошуку спільних попутників автомобілем, проте в останній час він купив декілька онлайн-сервісів бронювання автобусинх квитків і відповідно тепер на ньому можливо також купувати квитки на автобусі. Пасажири можуть обрати з широкого розмаїття автобусних перевізників різної ціни і рівня сервісу. Для кожного рейсу вказані додаткові опції які може мати автобус, як от WiFi чи наявність напоїв. Як перевізник, так і користувач може також скасувати свій квиток на рейс, проте у випадку користувача потрібно буде сплатити також сервісний збір. Оплата можлива через LiqPay.

1.2 Огляд середовища розробки та застосованих технологій

В процесі розробки системи бронювання були використані система керування базами даних MySQL версії 8.0.28 та середовище розробки Visual Studio Community 2019 з установленим .NET Framework 4.8. Для з'єднання програми з базою даних застосовується MySQL Connector .Net версії 8.0.18.

.NET Framework

.NET Framework — це платформа розробки програмного забезпечення, розроблена корпорацією Майкрософт, яка забезпечує середовище виконання та набір бібліотек і інструментів для створення та запуску програм в операційних системах Windows. Фреймворк містить різноманітні мови програмування, такі як C#, F# та Visual Basic, і підтримує низку типів додатків, включаючи настільні, веб-, мобільні та ігрові програми. .NET Framework включає два основні компоненти: середовище виконання Common Language Runtime (CLR) і бібліотеку класів .NET Framework. CLR відповідає за керування виконанням коду, написаного будь-якою з підтримуваних мов, тоді як бібліотека класів надає великий набір попередньо створених функцій і класів, які можна використовувати для створення широкого діапазону програм [1,6].

- Однією з ключових переваг .NET Framework є підтримка різноманітних мов програмування. Це означає, що розробники можуть вибрати мову, яка найкраще відповідає їхнім потребам і знанням, і водночас мати можливість використовувати той самий набір бібліотек та інструментів, що надаються фреймворком.
- Ще однією перевагою .NET Framework є підтримка різних типів програм. Фреймворк містить бібліотеки та інструменти для створення настільних, мобільних, ігрових та веб-додатків, що робить його універсальним вибором для розробників, які працюють над широким спектром проєктів.
- .NET Framework також надає ряд функцій, які допомагають покращити безпеку, надійність і продуктивність програм. До них належать такі функції,

як безпека доступу до коду, автоматичне керування пам’яттю та just-in-time (JIT) компіляція, що допомагає підвищити швидкість виконання програми [1].

- .NET Framework також розроблено для інтеграції з іншими технологіями Microsoft, такими як Microsoft SQL Server, Microsoft SharePoint та Microsoft Office, що спрощує створення програм, які чудово працюють з іншими продуктами Microsoft.

Втім, як і в інших програмних продуктів, у .NET Framework існують також і недоліки. Серед них: інсталяція .NET може вимагати значних ресурсів та займати багато місця, що дещо обмежує розробників у своїх можливостях; іноді потрібно використовувати конкретну версію .NET і може виникнути залежність від певної версії платформи; деякі мови, зокрема C#, можуть бути менш продуктивними порівняно з компільованими мовами тощо [6].

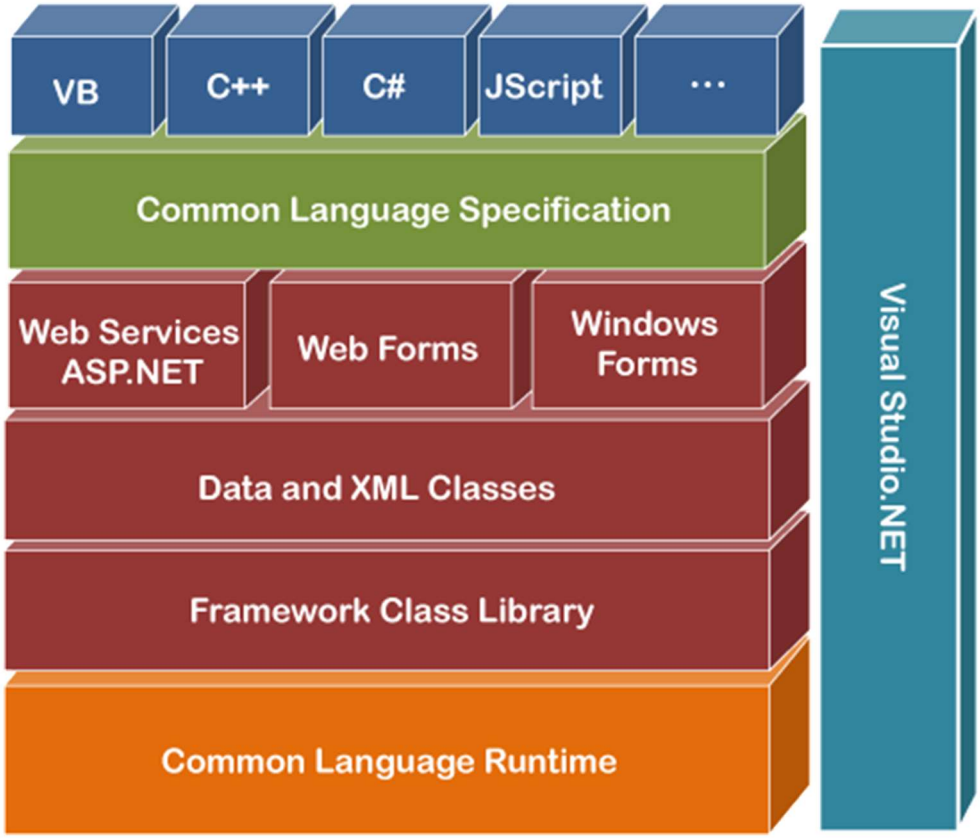


Рисунок 1.4. Архітектура .NET Framework.

Загалом .NET Framework — це потужна та універсальна платформа розробки, яка надає широкий спектр інструментів і бібліотек для створення та виконання програм в операційних системах Windows. Простіше кажучи, це віртуальна машина для компіляції та виконання програм, написаних різними мовами, такими як C#, VB.Net тощо [6].

.NET — це програмна основа, розроблена та розроблена корпорацією Microsoft. Першою версією .Net Framework була 1.0, яка вийшла в 2002 році. Він використовується для розробки додатків на основі форм, веб-додатків і веб-служб. На платформі .Net доступні різні мови програмування, найпоширенішими з яких є VB.Net і C#. Він використовується для створення програм для Windows, телефонів, Інтернету тощо. Він надає багато функцій, а також підтримує галузеві стандарти [1]. Індустрія постійно еволюціонує, і Microsoft розробляє нові технології, такі як .NET Core та .NET 5+, які вирішують деякі з цих питань та додають нові функції.

Visual Studio

Розробку на платформі .NET Framework я проводив в середовищі Visual Studio версії 2019. Visual Studio 2019 — це інтегроване середовище розробки (IDE) від компанії Microsoft, призначене для створення різноманітних програмних продуктів: настільних програм, GUI (графічного інтерфейсу користувача), консолі, веб-програм, мобільних програм, хмарних і веб-служб тощо. За допомогою цього IDE ви можете створювати керований та native код. Воно використовує різні платформи програмного забезпечення для розробки програмного забезпечення Microsoft, як-от магазин Windows, Microsoft Silverlight, Windows API тощо. Це не IDE для певної мови, оскільки ви можете використовувати його для написання коду на C#, C++, VB (Visual Basic), Python, JavaScript та багато інших мов. Воно доступне як для Windows, так і для macOS. Воно є одним із найпопулярніших інструментів розробки серед програмістів і розробників програмного забезпечення [7]. Ось кілька ключових характеристик Visual Studio 2019:

1. Visual Studio 2019 підтримує багато мов програмування, включаючи C#, VB.NET, F#, C++, Python, і інші – загалом 36 різних мов програмування. Це дозволяє розробникам використовувати ту мову, яка найкраще підходить для їхнього завдання.
2. В інструменті доступні шаблони для розробки різноманітних додатків, таких як веб-додатки, десктоп-додатки, мобільні додатки, служби та інше.
3. Visual Studio 2019 підтримує розробку для різних платформ, включаючи Windows, Android, iOS, Linux, Azure та інші.
4. Вбудовані інструменти для розробки:
    - Включає потужний інструмент для відлагодження коду, вивчення стеку викликів, дослідження змінних і багато іншого.
    - Має вбудовані інструменти для розробки інтерфейсів користувача (UI) для різних типів додатків.
    - Забезпечує інструменти для аналізу коду на предмет потенційних проблем і оптимізацій.
5. Visual Studio 2019 інтегровано з іншими інструментами Microsoft для керування версіями, спільної роботи, автоматизації збірки і розгортання, зокрема Team Foundation Server і Azure DevOps, що полегшує розробку в командному середовищі.
6. Visual Studio 2019 дозволяє розробникам створювати додатки для нових версій платформи .NET, таких як .NET Core і .NET 5+.

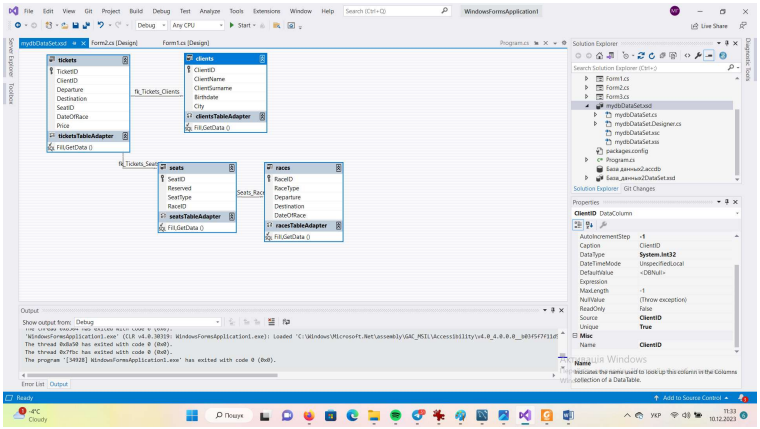


Рисунок 1.5. Розробка програми з використанням Visual Studio 2019.

Visual Studio 2019 є потужним та розширюваним інструментом для розробки програмного забезпечення, що застосовується в широкому спектрі проєктів та рішень.

MySQL

Розробка та відлагодження серверної частини (база даних) проводилася на платформі MySQL. MySQL — це реляційна система управління базами даних (СУБД або СКБД), яка використовує мову SQL (Structured Query Language) для зберігання, керування та взаємодії з даними. Розроблена спочатку шведською компанією MySQL AB, яку згодом придбала компанія Sun Microsystems, а потім Oracle Corporation. Вона відкрита, безкоштовна та широко використовується в різних веб-проєктах [8].

MySQL є одним із багатьох варіантів програмного забезпечення РСКБД. РСУБД і MySQL часто вважаються одним та тим самим через популярність MySQL. Кілька великих веб-програм, таких як Facebook, Twitter, YouTube, Google і Yahoo! усі використовують MySQL для зберігання даних. Незважаючи на те, що спочатку він був створений для обмеженого використання, тепер він сумісний з багатьма важливими обчислювальними платформами, такими як Linux, macOS, Microsoft Windows і Ubuntu. Порівняно з іншими системами керування базами даних MySQL має наступні переваги**:**

1. MySQL є відкритим програмним забезпеченням з ліцензією GPL, що дозволяє користувачам використовувати, змінювати і поширювати код безкоштовно.
2. MySQL підтримує різні платформи, включаючи Windows, Linux, macOS та інші.
3. Управління індексами та оптимізація запитів дозволяють MySQL працювати ефективно та швидко, особливо для великих обсягів даних.
4. Багата спільнота користувачів і розробників MySQL надає широкий спектр документації, підтримки та додаткових ресурсів.

5. MySQL підтримує різні методи реплікації та кластеризації, що дозволяє підвищити доступність та надійність.
6. Має багато різноманітних можливостей, включаючи тригери, сховища, збережені процедури та інші.

Поряд із цим MySQL має також і певні недоліки**:**

1. У порівнянні з деякими конкурентами, MySQL може не мати деяких розширених функцій, таких як повна підтримка OLAP.
2. MySQL не має повноцінної підтримки віконних функцій, що може бути важливим для певних аналітичних операцій.
3. При великому обсязі даних деякі типи JOIN можуть призводити до втрати продуктивності.
4. Інструменти моніторингу та управління базою даних у MySQL можуть бути менш деталізованими порівняно з деякими іншими системами управління базами даних.
5. Хоча MySQL має деякі засоби забезпечення безпеки, належна конфігурація та обслуговування дуже важливі для уникнення потенційних загроз безпеки.

Всі ці фактори повинні бути враховані в залежності від конкретного використання та вимог вашого проєкту. Кожен клієнт може зробити запит із графічного інтерфейсу користувача (GUI) на своїх екранах, і сервер видасть бажаний результат. Одними з найпопулярніших графічних інтерфейсів MySQL є MySQL Workbench (рис. 1.6), SequelPro, DBVisualizer і Navicat DB Admin Tool. Деякі з них безкоштовні, інші комерційні [8]. В даному випадку був обраний саме MySQL Workbench (рис. 1.6).

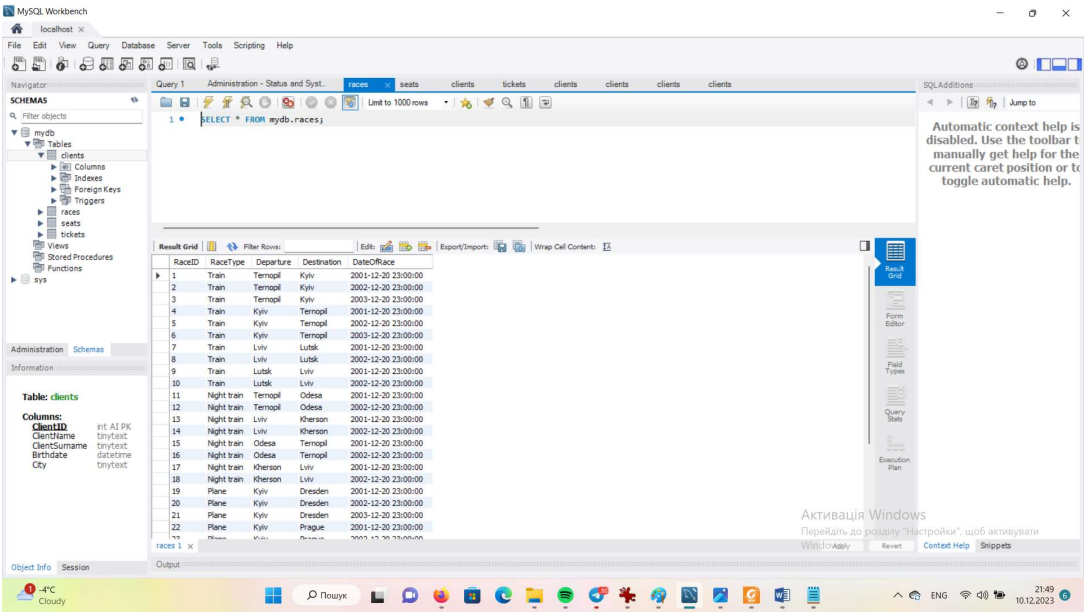


Рисунок 1.6. Розробка та відлагодження БД з використанням MySQL Workbench.

1.3. Вибір моделі розробки системи

Для найбільш ефективного проєктування заданої системи з виконанням всіх вимог варто правильно обрати методологію розробки програмного забезпечення. Методологія розробки програмного забезпечення (ПЗ) — це систематичний підхід до планування, створення, тестування та управління процесом розробки програмного продукту. Методології надають набір правил, процедур і процесів, які допомагають забезпечити ефективність та якість розробки програмного забезпечення. Ці підходи можуть бути адаптовані до конкретних потреб та характеристик проєкту [2].

Ось деякі популярні методології розробки ПЗ:

а) Каскадна модель (Waterfall) – лінійний процес розробки, де кожен етап виконується послідовно, один за одним.

1) Переваги: Простота управління, чітка документація, підходить для стабільних вимог.

2) Недоліки: Не гнучкий, важко адаптується до змін, ризик невідомості на ранніх етапах.

б) Ітеративна та інкрементальна розробка – Процес розробки поділяється на ітерації, кожна з яких пройде через усі фази розробки.
    1) Переваги: Гнучкість, здатність швидко реагувати на зміни, можливість поступового додавання функцій.
    2) Недоліки: Потребує багато уваги до управління версіями.

в) Scrum – фреймворк для розробки програмного забезпечення, оснований на ітераційному та інкрементальному підходах. Використовується в агільному управлінні проєктами.
    1) Переваги: Гнучкість, акцент на комунікації в команді, швидке впровадження змін.
    2) Недоліки: Не підходить для всіх типів проєктів, вимагає певного рівня самоорганізації в команді.

г) Kanban – використовується для візуалізації робочого процесу та оптимізації потоку роботи.
    1) Переваги: Гнучкість, підвищена прозорість, покращення ефективності робочого процесу.
    2) Недоліки: Може бути складним для впровадження великими командами.

д) DevOps – інтеграція розробки (Dev) та операцій (Ops) для поліпшення комунікації та автоматизації процесів розробки та впровадження.
    1) Переваги: Швидше впровадження змін, автоматизація тестування та впровадження.
    2) Недоліки: Вимагає значного зусилля для впровадження, відмінності в культурі та підходах.

е) RUP (Rational Unified Process) — різновид ітеративного та інкрементального фреймворку розробки програмного забезпечення, розроблений компанією Rational Software, яку згодом придбала IBM [2].
    1) Переваги: Підтримка моделювання, планування завдань та управління ризиками.

2) Недоліки: Не підходить для малих проєктів, значна ресурсоємність та складність документації.

Ці методології можуть використовуватися окремо або комбінуватися в залежності від конкретного проєкту та його вимог. Кожна з них має свої переваги та недоліки, і вибір залежить від конкретних обставин проєкту та уподобань команди розробників.

Для виконання даної роботи було обрано каскадну модель розробки, як таку, яка найбільше відповідає вимогам створення малих проєктів у відносно стислі терміни.

2. РОЗРОБКА МОДЕЛІ ТА ПРОГРАМНОГО КОМПЛЕКСУ

2.1 Розробка моделі предметної області системи бронювання квитків

Мета атестаційної роботи магістра — створення якісного необхідного програмного забезпечення, що повинне спростити бронювання квитку. Таким чином, однією з умов до цієї програми буде інтуїтивно зрозумілий інтерфейс.

Програма повинна давати можливість користувачу додавати та редагувати інформацію про клієнтів, їхню інформацію для ідентифікації, квитки, які вони купують, в тому числі місце відправки та призначення, їхню ціну задля змоги повернення, місця в потягах, а також збереження та видалення інформації в базах.

Для аналізу сфери діяльності необхідно оцінити як розроблювана системи може вплинути на бізнес і роботу компанії в цілому. Для цього створюється матриця аналізу підприємства. В неї вносяться всі ризики можливості від створення системи, сильні і слабкі сторони підприємства при її впроваджені в робочий цикл компанії. Знак «+» - означає позитивний ефект системи на роботу компанії, знак «-» - негативний вплив та можливу загрозу для подальшого функціонування.

Таблиця 2.1. Матриця аналізу підприємства

Сильні сторони		Можливості	
Збільшення зарплати та кваліфікації працівників галузі	+	Збільшення комфорту пасажирів шляхом пришвидшення видачі квитків	+
Збільшення контролю над бухгалтерією	+	Встановлення оперативного зв’язку між відділами	+

Продовження таблиці 2.1.

Слабкі сторони		Загрози	
Недостатньо кваліфікований персонал	-	Обмеження структурних нововведень в систему	-
Відсутність типової структури підприємства	-	Неможливість швидкої дії на непередбачувані ситуації без згоди керівництва	-
Постійна необхідність обслуговування системи	-	Зменшення кількості робочих місць	-

З отриманої таблиці можна судити про те наскільки вигідною є розробка даної системи автоматизації. З даних приведених вище можна судити про позитивний стрибок контролю над бухгалтерією та приведення праці касира до певної структурованої роботи. Серед негативних сторін можна визначити непідготовленість персоналу до нововведень та можливі кадрові перестановки що призведе до деякого сповільнення процесу роботи.

Отже, мета атестаційної роботи – це створення нового ефективного програмного забезпечення, яке має на меті спростити та пришвидшити бронювання квитка.

Ця програма повинна відповідати наступним вимогам:

- мати інтуїтивно зрозумілий інтерфейс, бути простою у використанні;
- мати актуальну базу та алгоритми роботи з нею;
- бути правильно спроєктована, правильно використовувати ресурси пристрою.

2.2 Проєктування бази даних

MySQL поєднує відомості з різних джерел в одній реляційній базі даних. Реляційна база даних є найпопулярнішим типом бази даних, яка зберігає та надає доступ до точок даних, пов’язаних одна з одною. У реляційних базах даних дані впорядковуються та зберігаються в таблицях, що складаються зі стовпців і рядків [8,9].

Кожен рядок у таблиці є записом з унікальним ідентифікатором, який називається первинним ключем. Стовпці таблиці містять атрибути даних. Оскільки кожен запис зазвичай має значення для кожного атрибута, стає легко формувати гнучкі зв’язки між точками даних. Давайте розглянемо основні поняття, пов’язані з реляційними базами даних:

- Первинний ключ — це унікальний ідентифікатор, який ідентифікує кожен рядок таблиці.
- Зовнішній ключ — це поле (або кілька полів) в одній таблиці, яке посилається на первинний ключ в іншій таблиці. Це допомагає встановити зв'язки між таблицями.
- Представлення — це віртуальна таблиця, яка не зберігає дані; замість цього вона представляє певні вихідні дані, які обчислюються з базових таблиць.
- Індекс — це структура даних, яка містить копію стовпця (або кількох стовпців) із таблиці бази даних, яка впорядкована для прискорення операцій пошуку бази даних у вихідному стовпці.

Існує три основних типи зв'язків між таблицями, які забезпечують абсолютну гнучкість моделі реляційної бази даних.

- Відношення один-до-одного – одному запису в таблиці відповідає лише один такий ж запис в іншій таблиці.
- Відношення один-до-багатьох – одному запису в таблиці відповідає декілька записів в іншій таблиці.

- Відношення багато-до-багатьох – багатьом записам в таблиці відповідає багато записів в іншій таблиці.

Структурована мова запитів (SQL) — це доменно-специфічна мова, яка використовується для зберігання, отримання та обробки даних — стандарт для реляційних баз даних. Клієнт і сервер використовують її для спілкування в середовищі реляційної бази даних. Всі інші реляційні бази даних також використовують синтаксис Structured Query Language. Програмне забезпечення реляційних баз даних часто пишеться іншими мовами програмування, але завжди використовує SQL як основну мову для взаємодії з базою даних [8,9].

SQL повідомляє серверу, що робити з даними. У цьому випадку оператори SQL можуть сказати серверу виконати певні операції:

- Маніпулювання даними – додавання, видалення, зміна, сортування та інші операції для зміни даних, значень або візуальних елементів.
- Ідентифікація даних – визначення типів даних, напр. зміна числових даних на цілі. Це також включає визначення схеми або зв'язку кожної таблиці в базі даних
- Контроль доступу до даних – забезпечення техніки безпеки для захисту даних. Це включає в себе рішення про те, хто може переглядати або використовувати будь-яку інформацію, що зберігається в базі даних.

SQL-запит — це запит інформації від реляційної БД. Користувачі пишуть і виконують запити, щоб отримати, додати, змінити та видалити дані з реляційних баз даних.

Створення бази даних — це процес, під час якого визначається структура бази даних, включаючи таблиці, поля та взаємозв'язки між ними [5]. Ось загальний огляд етапів створення бази даних:

1. Визначення вимог. Починається з визначення бізнес-вимог та потреб користувачів. Спілкування з зацікавленими сторонами допомагає з'ясувати, які дані необхідні та як вони будуть використовуватися. Загальні специфікації були визначені в розділі 2.1.1.

2. Вибір системи керування базами даних (СКБД). Обирається конкретна система керування базами даних. Вибір залежить від потреб проєкту та відомостей про конкретні СКБД, в даному випадку це MySQL.
3. Проєктування бази даних. Проєктування бази даних включає в себе створення схеми бази даних, визначення таблиць, полів, первинних та зовнішніх ключів. Розглядається логічна та фізична модель даних.
4. Створення таблиць та полів. Відповідно до розробленої схеми бази даних створюються таблиці з відповідними полями. Визначається тип даних для кожного поля (наприклад, текст, число, дата). В даній БД було створено спершу дві таблиці, які під час нормалізації було поділено до чотирьох.
5. Визначення відношень між таблицями. Встановлюються взаємозв'язки між таблицями, що визначають, як дані пов'язані між собою. Це може бути один до багатьох, багато до багатьох та ін. Для цієї бази даних використано взаємовідношення “один до багатьох”.
6. Нормалізація. Відбувається процес нормалізації, який дозволяє оптимізувати базу даних для уникнення дублювання даних та підтримання цілісності даних, про нього нижче.
7. Створення запитів. Створюються запити для взаємодії з даними, включаючи запити на вибірку, вставку, оновлення та видалення даних. В даному проєкті запити до бази даних будуть винесені в окрему програму з графічним інтерфейсом користувача задля зручності користування.
8. Створення процедур та тригерів (за потреби). Збережені процедури та тригери можуть бути створені для забезпечення виконання деяких дій на рівні бази даних, таких як автоматизовані обчислення чи перевірка правил цілісності. На даний проєкт такої потреби нема.
9. Заповнення бази даних даними (попереднє завантаження). Якщо це необхідно, база даних може бути наповнена початковими даними. В даному конкретному випадку БД буде частково заповнена з двох CSV-таблиць Races та Seats.

10. Тестування та оптимізація. Проводиться тестування для перевірки правильності роботи бази даних та оптимізації її продуктивності. Це буде здійснено пізніше, в процесі загального тестування отриманої системи.
11. Впровадження. База даних впроваджується в роботу, і її використання починається відповідно до потреб бізнесу.

Ці етапи можуть дещо варіюватися в залежності від конкретних потреб проєкту і обраної методології розробки програмного забезпечення [9].

Нормалізація бази даних — це процес організації даних в базі даних таким чином, щоб забезпечити ефективність та уникнути аномалій при зберіганні та оновленні даних. Цей процес розділяє таблиці бази даних на менші та більш узгоджені одиниці, що сприяє покращенню структури даних та зниженню повторення інформації. Нормалізація бази даних зазвичай виконується за допомогою набору правил, які визначають структуру та організацію даних в таблицях. Найбільш відомі форми нормалізації включають першу нормальну форму (1NF), другу нормальну форму (2NF), третю нормальну форму (3NF) та інші.

Основні правила нормалізації:

- Перша нормальна форма (1NF). Кожен рядок таблиці повинен містити атомарні значення, тобто значення, які не можна розбити на більш прості складові.
- Друга нормальна форма (2NF). Таблиці повинні бути в першій нормальній формі, і кожен неключовий стовпець повинен повністю залежати від кожного ключа. Це означає, що не повинно бути часткової залежності.
- Третя нормальна форма (3NF). Таблиці повинні бути в другій нормальній формі, і всі стовпці, які не є частиною ключа, повинні бути взаємно незалежними. Це означає, що не повинно бути транзитивної залежності.
- Інші форми нормалізації (BCNF, 4NF, 5NF і т. д.). Є більш високі форми нормалізації, такі як четверта та п'ята нормальні форми, які використовуються для більш глибокого розбиття даних в зусиллях уникнути певних аномалій.

При користуванні нормалізацією потрібно враховувати компроміс між оптимізацією для читання та оптимізацією для запису. Іноді велика кількість зв'язків між таблицями може впливати на продуктивність читання, але спрощує процес оновлення даних. Кращий підхід залежить від конкретних вимог конкретного додатку чи системи. Мною була для додатку виконана нормалізація до третьої нормальної форми як найбільш оптимальне в даному випадку рішення.

Структура бази даних визначає, як дані організовані та взаємодіють між собою в межах бази даних. Це описує, як створені та організовані таблиці, поля, відносини та інші елементи бази даних. При визначенні полів таблиці для кожного поля створюють унікальні ім'я, тип даних, якими буде надалі заповнене це поле. Для деяких типів (наприклад VARCHAR) вводять розмір поля (кількість байтів). Коли визначають тип даних, необхідно враховувати, значення які будуть введені в дану колонку. Зверніть увагу на тип поля «лічильник»: якщо значення іншого типу можуть повторюватися в межах одного поля, «лічильник» є результатом роботи арифметичної прогресії, тому його значення ніколи не дублюються [5].

Структура бази даних визначається на етапі проєктування бази даних та повинна відповідати потребам конкретного додатка або системи. Правильно спроєктована структура дозволяє ефективно зберігати, оновлювати та витягувати дані з бази даних.

Внаслідок кроків, описаних вище, було створено базу даних, яка вміщає в себе чотири таблиці, вони пов’язані між собою зв’язками “один до багатьох”. EER-діаграма цієї бази зображена на рисунку 2.1. База була нормалізована до третьої нормальної форми, є достатньо інформативною та відповідає вимогам, яка ставляться до системи.

Нижче наведено список створених таблиць та полів.

a) tickets – таблиця, яка містить в собі список квитків. Містить всі необхідні для квитків поля:
    1) TicketId – поле-лічильник, яке позначає номер квитка. Є унікальним ідентифікатором для цієї таблиці.

2) ClientId – поле, яке позначає ID клієнта, і про яке нижче. Цим ж полем вона пов’язана з таблицею Clients.
3) Departure – текстове поле, яке позначає точку відправлення рейсу.
4) Destination – текстове поле, яке позначає точку призначення рейсу.
5) SeatId – поле, яке позначає ID місця, і про яке нижче. Цим ж полем вона пов’язана з таблицею Seats.
6) DateOfRace – поле, яке позначає дату поїздки.
7) Price – числове (float) поле, яке містить ціну оплаченої поїздки.

б) races – містить всі необхідні для рейсів поля:
1) RaceId – поле-лічильник, яке позначає номерний ID рейсу. Є унікальним ідентифікатором для цієї таблиці.
2) Departure – текстове поле, яке позначає точку відправлення рейсу.
3) Destination – текстове поле, яке позначає точку призначення рейсу.
4) RaceType – текстове поле, яке визначає тип транспортного засобу, який застосовується, та уточнює його клас (приклад: літак першого класу).
5) DateOfRace – специфікує дату рейсу.

в) clients – містить персональні дані клієнтів:
1) ClientID – поле-лічильник, яке позначає номерний ID клієнта. Є унікальним ідентифікатором для цієї таблиці.
2) ClientName – текстове поле, що містить ім’я клієнта.
3) ClientSurname – текстове поле, що містить прізвище клієнта.
4) Birthdate – поле, яке містить дату народження клієнта.
5) City – текстове поле, яке містить назву міста, де проживає клієнт.

г) seats – містить дані про зайняте місце:

1) SeatId – поле-лічильник, яке позначає номер місця. Є унікальним ідентифікатором для цієї таблиці.
2) SeatType – текстове поле, яке містить тип місця (приклад: сидіння без столика).
3) Reserved – поле типу bool, яке позначає зайнятість місця. Змінюється автоматично, а також можна змінити вручну.
4) RaceId – поле, яке позначає ID рейсу, і про яке вище. Цим ж полем вона пов’язана з таблицею Races.

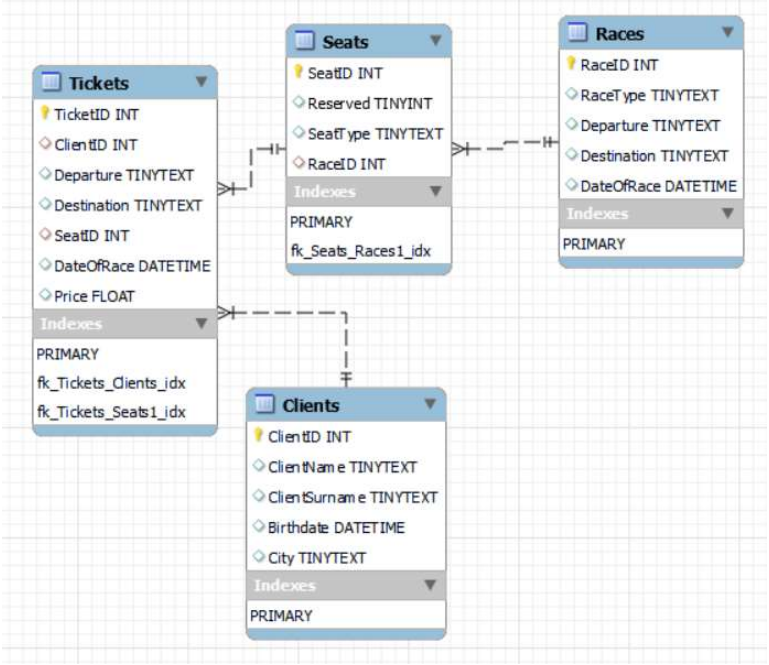


Рисунок 2.1. EER-діаграма створеної бази даних

2.3 Розробка бізнес-моделі системи

Розробка бізнес-моделі системи включає в себе розробку варіантів використання (use cases) є важливою частиною процесу розробки програмного забезпечення при аналізі вимог і допомагає створити систему, яка відповідає реальним потребам користувачів [2]. Варіанти використання є інструментом для

опису того, як користувачі будуть взаємодіяти з системою та як система відповідатиме на їхні дії. Основні причини розробки варіантів використання включають:

- Варіанти використання допомагають розробникам та аналітикам зрозуміти, які конкретні функціональність та властивості системи важливі для користувачів. Це дозволяє фокусуватися на розробці та вдосконаленні тих частин системи, які найбільше цікавлять користувачів.
- Варіанти використання визначають, як користувачі будуть взаємодіяти з системою, включаючи всі можливі сценарії та варіанти. Це допомагає визначити межі системи та обсяг функціоналу.
- Варіанти використання стають основою для визначення функціональності системи. Кожен варіант використання представляє собою конкретний сценарій взаємодії, що допомагає уточнити, як система повинна вести себе у конкретних ситуаціях.
- Через варіанти використання можна визначити потреби інфраструктури для системи, такі як обладнання, мережі та інші ресурси, необхідні для правильної роботи.
- Варіанти використання стають ефективним інструментом для спілкування з замовниками та розробниками. Ці документи допомагають уникнути непорозумінь та допомагають сторонам зрозуміти очікування та функціональність системи.
- Варіанти використання можуть служити основою для тестування системи. Кожен варіант використання може бути перевірений, щоб впевнитися, що система веде себе правильно в конкретних сценаріях.

В даній бізнес-моделі система представлена як визначена кількість користувачів (акторів), які взаємодіють з системою, та сценаріїв власне їхньої взаємодії з системою, себто власне варіантів використання. Кожен сценарій взаємодії з системою є довершеним алгоритмом дій. Це означає, що після того як система закінчить обробку запиту користувача, вона повинна повернутися в

початковий стан, в якому готова до виконання наступних запитів. В ході розробки були визначені наступні актори та варіанти використання:

Таблиця 2.2. Реєстр ролей

Код	Назва ролі	Опис ролі
К	Касир	Вводить дані про клієнтів. Оформлює їхнє замовлення та вносить його в базу даних. В разі потреби вносить зміни.

Таблиця 2.3. Сценарії використання

Код	Назва ролі	Обов'язковість	Опис сценарію
К1	Касир	Обов'язковий	Варіант використання дозволяє касиру реєструвати нові замовлення.
К2	Касир	Не обов'язковий	Варіант використання дозволяє касиру змінювати замовлення до початку його виконання.
К3	Касир	Не обов'язковий	Варіант використання дозволяє касиру видаляти скасовані замовлення.
К4	Касир	Обов'язковий	Варіант використання дозволяє касиру у разі виконання замовлення змінити стан місць для наступного рейсу.
К5	Касир	Обов'язковий	Варіант використання дозволяє додати нового клієнта до БД.

Нижче наведено детальний опис сценаріїв використання.

К1. Реєстрація замовлення

Даний сценарій використання дозволяє Касирові реєструвати і бронювати нові замовлення. Кожне замовлення в електронній формі містить дату необхідної поїздки, номер місця, його тип і і маршрут рейсу з вказівкою тривалості у часі. Для поїздок дата повинна носити обов'язковий характер.

К2. Редагування замовлення

Даний сценарій використання дозволяє Касирові змінювати дані замовлення при необхідності оновлення даних до його початку. Кожне замовлення в електронній формі може бути зміненим. Змінити можна дату поїздки, місце, клас тощо.

К3 Видалення замовлення

Даний сценарій використання дозволяє Касирові скасовувати замовлення на бажання клієнта. Кожне замовлення в електронній формі може бути скасоване до свого початку.

К4. Зміна стану місць

Даний сценарій використання дозволяє Касирові після здійснення поїздки змінити стан місць без скасування замовлення, тим самим надаючи можливість касирові звільнити їх для наступної поїздки чи позначити неробочими. Кожне місце має свій унікальний ідентифікатор, що дозволяє розрізняти їх та виявляти необхідні.

К5. Додати клієнта

Даний сценарій використання дозволяє додавати Касирові нового клієнта до списку, уможливлюючи клієнтові таким чином здійснювати замовлення на певні поїздки. У разі зміни даних у клієнта інформація в базі може бути також змінена.

Розроблена діаграма варіантів використання в проєктованій системі автоматизації бронювання квитків зображена на рисунку 2.2.

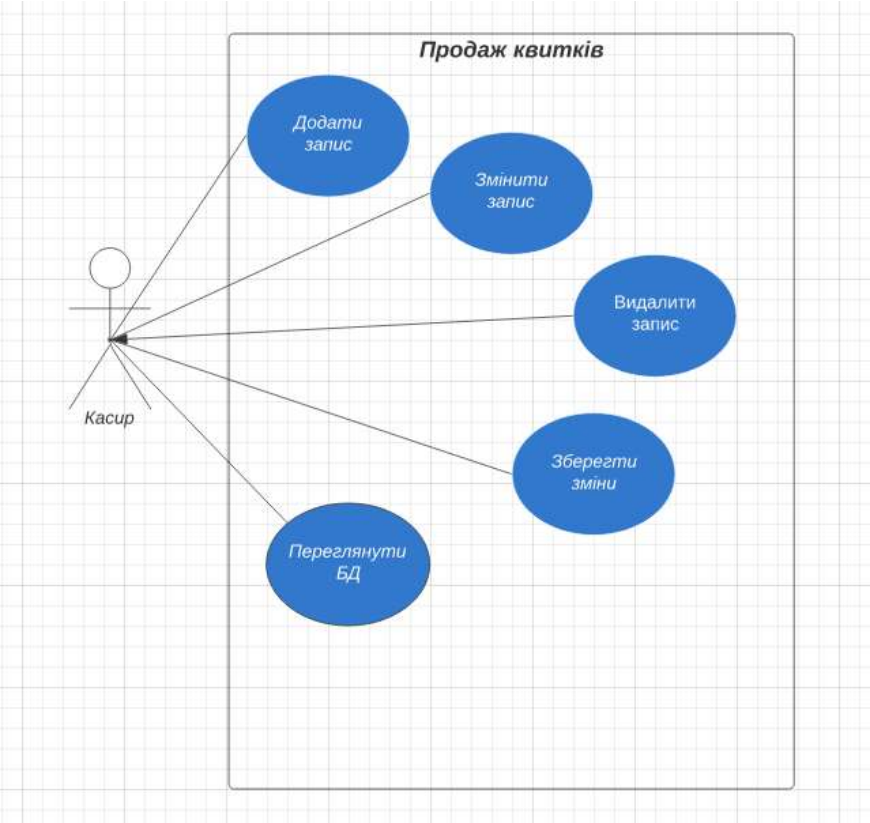


Рисунок 2.2. Діаграма варіантів використання

2.4 Розробка архітектури системи

Архітектура системи повинна зображувати саму структуру програми і зазвичай візуалізується діаграмою класів, відповідно діаграми класів у проєктуванні застосовуються найчастіше. Діаграма класів є одним із видів діаграм UML (Unified Modeling Language) і використовується для моделювання структури об'єктно-орієнтованого програмного забезпечення. Діаграма класів відображає класи, їх атрибути, методи та взаємовідношення між ними. Вона є потужним інструментом для візуалізації структури програми та розуміння взаємодій між об'єктами [2].

Основні елементи діаграми класів включають:

а) Класи – основний елемент діаграми класів, характеристика певного об’єкта. Кожен клас представляє собою шаблон для створення об'єктів. У діаграмі

класів клас зображується у вигляді прямокутника, розділеного на три секції: назва класу, атрибути та методи, про них піде нижче.

1) Атрибути – представляють дані, які належать класу. Вони зазвичай зображуються в другій секції прямокутника під назвою класу. Наприклад, якщо у класі "Квиток", атрибутами можуть бути "Місце", "Пасажир" та "Дата".
2) Методи – представляють функції або операції, які можуть бути викликані об'єктами класу. Вони зображуються в третій секції прямокутника. Наприклад, у класі "Квиток" можуть бути методи "Видача", "Повернення" та "Отримання Інформації".

б) Взаємовідношення – показують зв'язки між класами. Є різні типи взаємовідношень, про них мова нижче.

1) Асоціації – вказують на те, що об'єкти одного класу можуть взаємодіяти з об'єктами іншого класу. Вони представляються лініями, що з'єднують класи.
2) Узагальнення – вказує на те, що один клас успадковує або розширює функціональність іншого класу. Це представляється стрілкою, яка вказує на базовий клас.
3) Агрегація та композиція – вказують на те, як один клас пов'язаний з іншим, де композиція є більш тісним інтерфейсом, де один об'єкт є частиною іншого.

Діаграми класів використовуються під час аналізу та проєктування систем, щоб уточнити структуру програми та полегшити розуміння моделі об'єктів. Створення діаграм класів є важливим етапом у процесі розробки програмного забезпечення, оскільки вони полегшують розуміння та спілкування стосовно структури та функціоналу системи. Загалом, створення діаграм класів має кілька цілей та важливих використань у процесі розробки програмного забезпечення:

- Діаграми класів допомагають в аналізі вимог до системи. Вони дозволяють ідентифікувати класи, які будуть присутні в системі, та з'ясувати, як вони будуть взаємодіяти між собою та з користувачем.
- Діаграми класів стають ефективним інструментом комунікації між розробниками, архітекторами, менеджерами та замовниками. Вони надають зрозуміле та візуальне представлення структури системи.
- Діаграми класів є ключовим інструментом для моделювання структури програми в об'єктно-орієнтованому програмуванні. Вони дозволяють розробникам та архітекторам програмного забезпечення визначити класи, їх атрибути та методи, а також взаємозв'язки між класами, а пізніше також розподілити функціональність між класами та створити основну архітектуру системи.
- Діаграми класів є корисним інструментом під час фази програмування. Вони можуть служити як довідковий матеріал для розробників, які працюють над конкретними частинами системи. В майбутньому вони стають частиною документації проєкту. Вони допомагають документувати структуру програми, зв'язки та логіку взаємодії між класами.
- Діаграми класів можуть бути використані для визначення тестових сценаріїв та валідації системи. Це дозволяє перевірити, чи відповідає реалізація системи визначеній структурі та взаємодії класів.

Створення діаграм класів включає в себе кілька кроків і використання спеціальних інструментів для моделювання, таких як CASE (Computer-Aided Software Engineering) або графічних редакторів, що підтримують UML (Unified Modeling Language). Нижче наведено загальний процес створення діаграми класів:

- Ідентифікація класів, які будуть присутні в системі, та їх характеристик. Для кожного класу варто визначити атрибути (властивості) та їх типи даних, методи (функції або операції), які можуть бути викликані. Вказуйте аргументи, типи повернених значень та інші деталі.

- Встановлення взаємовідношення між класами. Використовуйте асоціації для визначення, як один клас пов'язаний з іншим. Розгляньте агрегацію, композицію та інші типи взаємовідношень. Якщо в системі є використання узагальнення, встановіть взаємозв'язок між базовим та похідним класами. Використовуйте стрілки для вказівки напрямку узагальнення.
- Використання спеціальних інструментів для моделювання, таких як CASE-інструменти або графічні редактори, що підтримують UML. Ці інструменти надають можливість створювати діаграми класів, зберігати їх, а також взаємодіяти з іншими елементами моделі.
- Збереження діаграми та її документація. Додайте коментарі або опис до класів, атрибутів, методів та взаємовідношень. Це полегшить розуміння діаграми іншим розробникам. Це може включати створення описового документа, що пояснює кожен елемент та зв'язок на діаграмі.

Важливо відзначити, що розробка діаграми класів - це ітеративний процес, і вона може змінюватися під час подальшого розвитку проєкту чи деталізації вимог. UML добре підходить для моделювання як логічних, так і фізичних схем баз даних. Багато систем, як от система бронювання квитків, мають у своєму складі збережені об’єкти. Це означає, що вони можуть бути поміщені у базу даних.

Створюючи діаграми класів UML варто пам’ятати, що кожна із них – це лише графічне зображення статичного дизайну системи. Жодна діаграма класів не зобов’язана включати все, що стосується дизайну системи. Але вона надає користувачеві повну інформацію, необхідну для статичного представлення системи; хоча кожна з них представляє лише один її аспект [3].

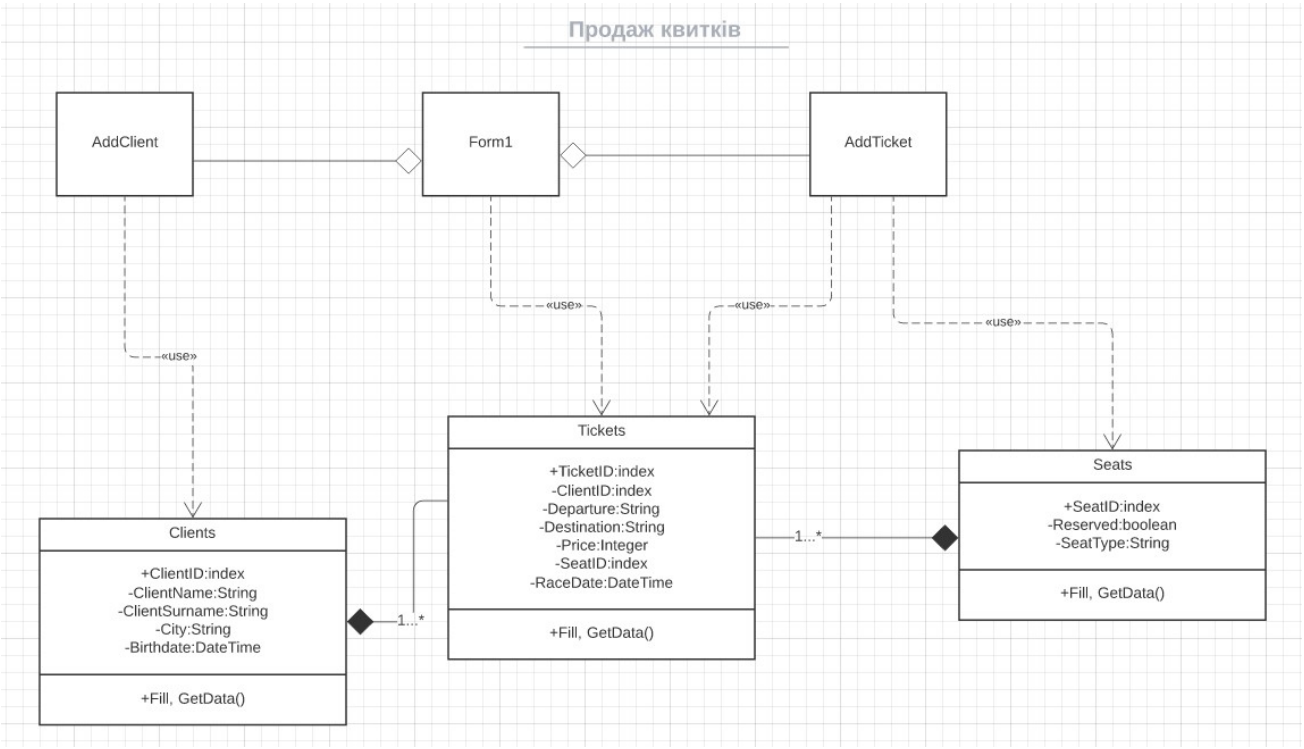


Рисунок 2.3. Діаграма класів

Наведена на рисунку діаграма класів містить в собі наступні класи.

а) AddClient, Form1 та AddTicket – класи графічного інтерфейсу Windows Forms, які створюються конструктором на етапі розробки, а тому детально розглядати тут не будемо.

б) Tickets

1) Поля даних, збіжні з полями відповідної таблиці, і які до них прив’язані.
2) void Fill(Tickets) – заповнити відповідну таблицю БД введеними в формі даними.
3) void GetData() – заповнити відповідний екземпляр класу відповідним записом БД.

в) Clients

1) Поля даних, збіжні з полями відповідної таблиці, і які до них прив’язані.
2) void Fill(Clients) – заповнити відповідну таблицю БД введеними в формі даними.

3) void GetData() – заповнити відповідний екземпляр класу відповідним записом БД.

г) Seats

1) Поля даних, збіжні з полями відповідної таблиці, і які до них прив’язані.
2) void Fill(Seats) – заповнити відповідну таблицю БД введеними в формі даними.
3) void GetData() – заповнити відповідний екземпляр класу відповідним записом БД.

3. КОНСТРУЮВАННЯ СИСТЕМИ

3.1. Реалізація елементів інтерфейсу доступу до БД

За допомогою майстра імпорту даних в MySQL Workbench заповнюю базу даних з двох таблиць: Seats.csv та Races.csv. В VS під'єдную базу даних. Створюю проєкт Windows Forms Application, де визначаємо три вікна. На першому додаю інструмент tabControl і створюємо дві вкладки: «Tickets» і «Clients» , на кожній зі вкладок додаємо кнопки «Add» і «Close» [4].

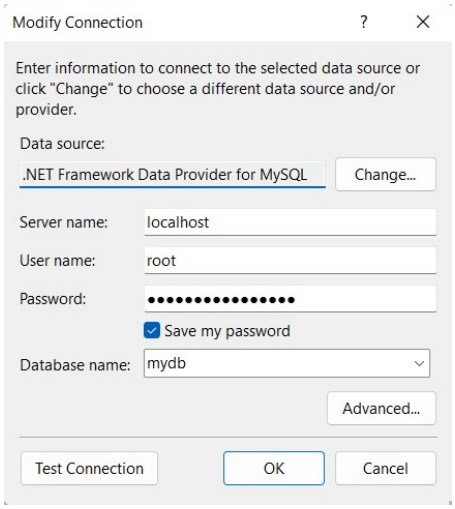


Рисунок 3.1. Вікно встановлення з'єднання з сервером MySQL

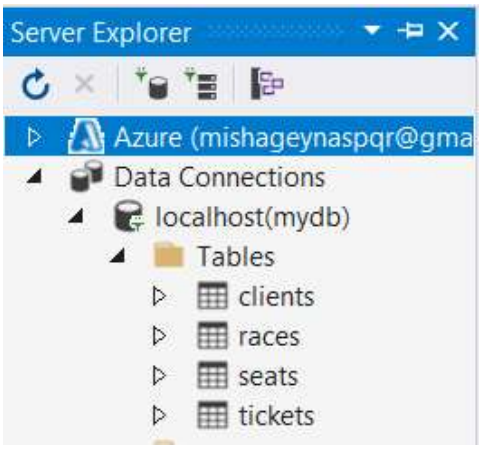


Рисунок 3.2. Під'єднана база даних

На кожну зі вкладок додаю по datagrid'у, які під'єдную до відповідних таблиць бази даних «Tickets» і «Clients» та налаштовую.

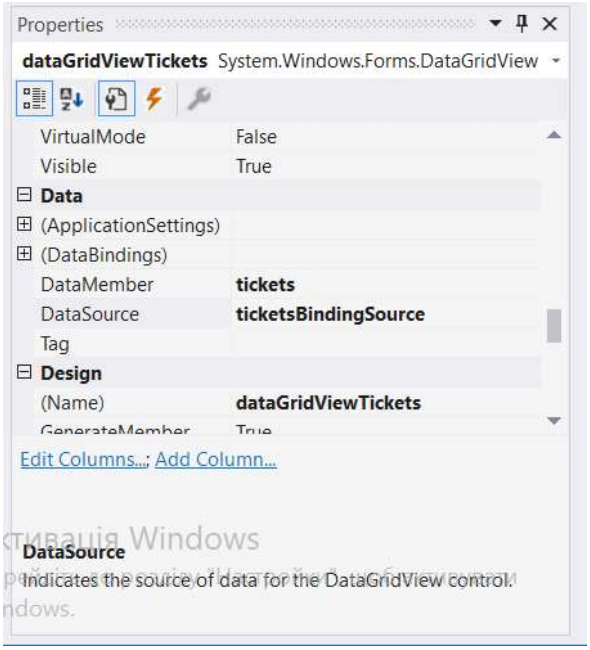


Рисунок 3.3. Під’єднання бази даних до datagrid’у.

Для кожної з вкладок створюю по формі діалогового вікна для додавання елементів у відповідну таблицю, до кнопок «Add» у кожній прописую відповідний код для підключення цих вікон. Під час перевірки працездатності код спрацював успішно.

```
private void btnAdd_Click(object sender, EventArgs e)
{
    Form add = new AddTicket();
    add.Owner = this;
    add.Show();
}

private void btnAddClient_Click(object sender, EventArgs e)
{

    Form cli = new AddClient();
    cli.Owner = this;
    cli.Show();
}
```

Рисунок 3.4. Методи відкриття допоміжних вікон.

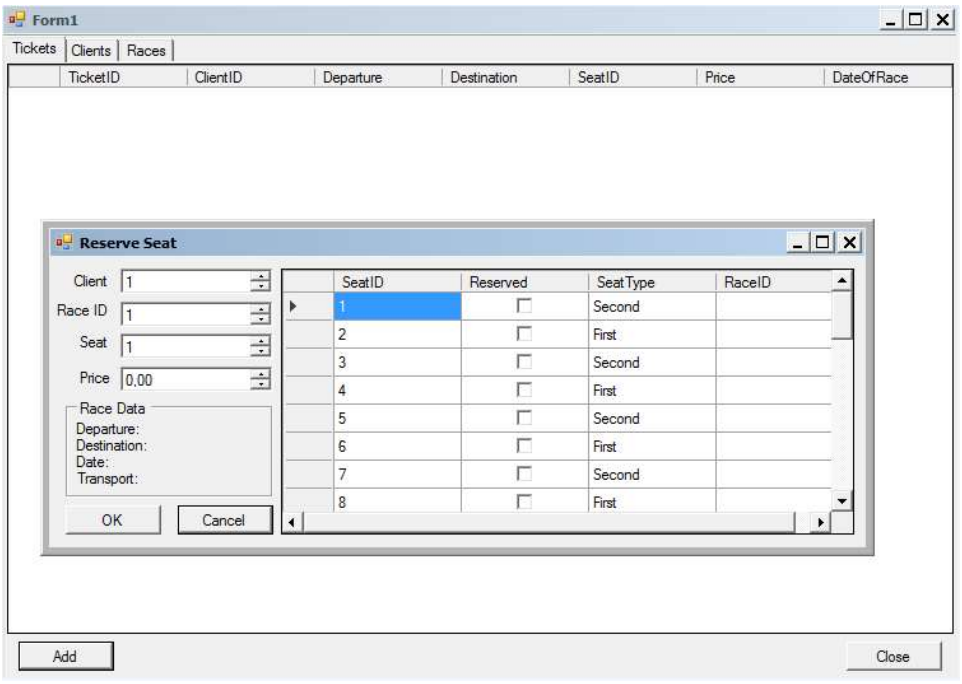


Рисунок 3.5. Вікно бронювання користувача.

Переходжу до роботи з діалоговими вікнами «AddTicket» та «AddClient». Тут додаємо до полів «Client» і «Price» NumericUpDown'и, до яких під'єднуємо відповідні поля з БД. До інших полів доєднуємо «TextBox'и», а до поля з датою – dateTimePicker.

До вікон додаємо метод Load() для передачі таблиць одна в одну. До кнопки «ОК» діалогового вікна «AddTicket» прописуємо відповідний код для додання запису в БД. Аналогічно і з кнопкою «ОК» діалогового вікна «AddClient».

```
private void Form1_Load(object sender, EventArgs e)
{
    this.racesTableAdapter1.Fill(this.mydbDataSet.races);
    this.clientsTableAdapter1.Fill(this.mydbDataSet.clients);
    this.ticketsTableAdapter1.Fill(this.mydbDataSet.tickets);

    dataGridViewTickets.Refresh();
    dataGridViewClients.Refresh();
    dataGridView1.Refresh();
}
```

Рисунок 3.6. Метод Load().

```
private void button2_Click(object sender, EventArgs e)
{
    try
    {
        string msg = "Do you want to add new ticket?";
        string caption = "Message";
        MessageBoxButtons buttons = MessageBoxButtons.YesNo;
        MessageBoxIcon ico = MessageBoxIcon.Question;

        DialogResult result;
        result = MessageBox.Show(this, msg, caption, buttons, ico);

        if (result == DialogResult.Yes)
        {
            try
            {
                numericUpDownClientID.Update();
                numericUpDownRaceID.Update();
                numericUpDownSeatID.Update();
                numericUpDownPrice.Update();

                Form1 main = this.Owner as Form1;
                DataRow Row = main.mydbDataSet.tickets.NewRow();
                int rc = main.dataGridViewTickets.RowCount + 1;
                int race = Convert.ToInt32(numericUpDownRaceID.Value) - 1;


                Row[0] = rc;
                Row[1] = numericUpDownClientID.Value;
                Row[2] = main.mydbDataSet.races.Rows[race][2];
                Row[3] = main.mydbDataSet.races.Rows[race][3];
                Row[4] = numericUpDownSeatID.Value;
                Row[5] = main.mydbDataSet.races.Rows[race][4];
                Row[6] = numericUpDownPrice.Value;

                main.mydbDataSet.tickets.Rows.Add(Row);
                main.mydbDataSet.tickets.AcceptChanges();
                ticketsTableAdapter1.Update(main.mydbDataSet.tickets);
                main.dataGridViewTickets.Refresh();

                this.Close();
            }
            catch (Exception ex)
            {
                string err = "Fill all fields properly!" + '\n' + ex;
                string c = "Error";
                buttons = MessageBoxButtons.OK;
                ico = MessageBoxIcon.Question;

                result = MessageBox.Show(this, err, c, buttons, ico);
            }
        }
        else
        {
            this.Close();
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Data not saved" + ex.Message.ToString(), "Error",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }

}
```

Рисунок 3.7. Метод додавання до бази даних нового запису.

Варто зазначити те, яким чином влаштований код збереження під час виходу з програми: в обробник подій програми додано наступний код, який виводить діалогове вікно про збереження. Якщо натиснуто «Так» – зміни зберігаються в окремий XSD-файл, якщо «Ні» – програма просто закривається.

```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    try
    {
        string msg = "Do you want to save the data?";
        string caption = "Save";
        MessageBoxButtons buttons = MessageBoxButtons.YesNo;
        MessageBoxIcon ico = MessageBoxIcon.Question;

        DialogResult result;
        result = MessageBox.Show(this, msg, caption, buttons, ico);

        if (result == DialogResult.Yes)
        {
            this.clientsBindingSource.EndEdit();
            this.clientsTableAdapter1.Update(this.mydbDataSet.clients);

            this.ticketsTableAdapter1.Fill(this.mydbDataSet.tickets);

            this.clientsBindingSource.EndEdit();
            this.clientsTableAdapter1.Update(this.mydbDataSet.clients);

            this.clientsTableAdapter1.Fill(this.mydbDataSet.clients);

            mydbDataSet.WriteXmlSchema("mydbDataSet.xsd");

            MessageBox.Show("Data saved", "Saved",
            MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        else
        {
            return;
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Data not saved" + ex.Message.ToString(), "Error",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

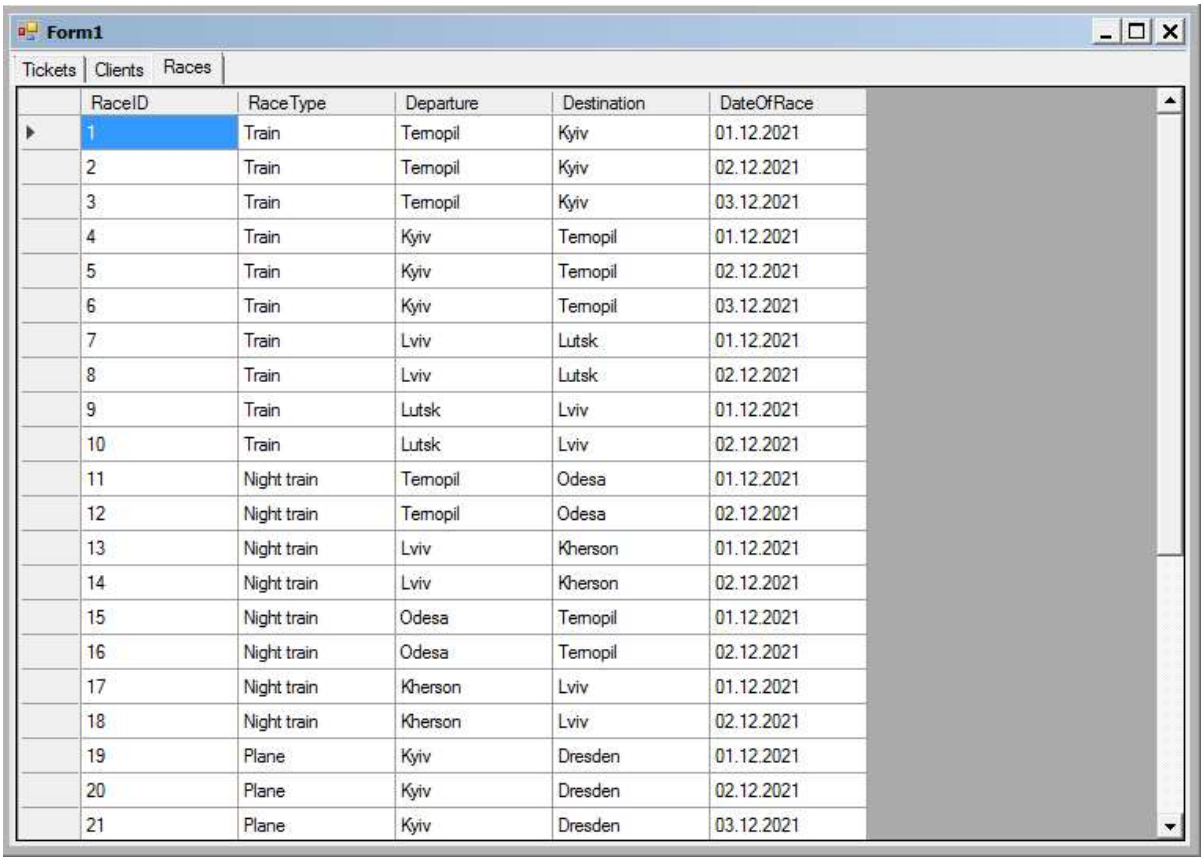
Рисунок 3.8. Метод збереження та закриття програми.

3.2. Тестування та оцінка якості створеної системи

Загальна мета тестування системи - це забезпечити якість та надійність програмного забезпечення перед його впровадженням в роботу або поширенням серед користувачів. Тестування системи включає в себе проведення різноманітних випробувань та перевірок для забезпечення того, що програмне забезпечення працює правильно, ефективно та відповідає вимогам. Основні цілі тестування системи включають:

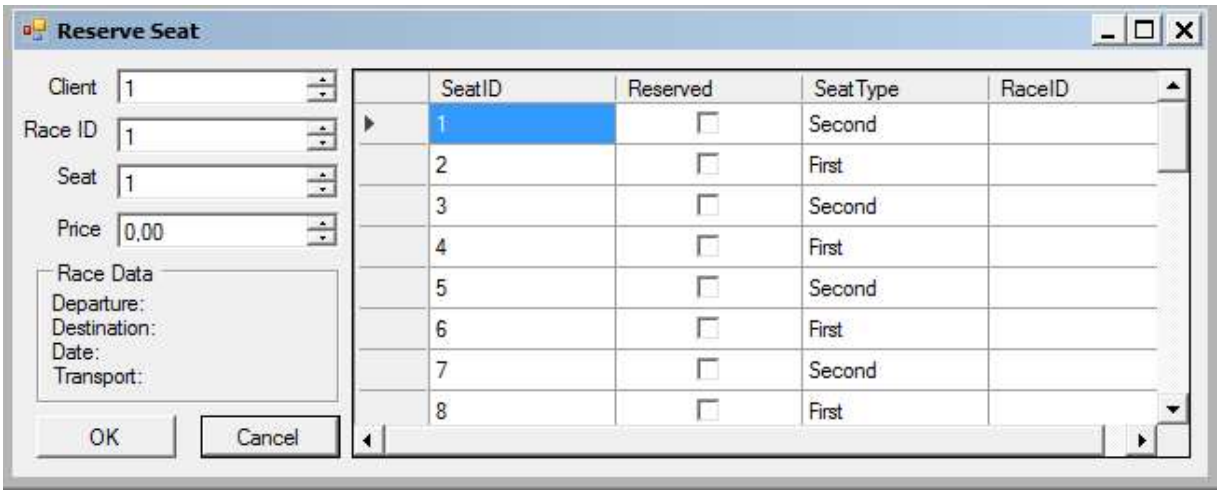
- Перевірка, чи правильно програмне забезпечення виконує всі функції та операції, які вимагаються в технічних та функціональних вимогах, та чи відповідає воно очікуванням користувачів. Визначення та усунення помилок, які можуть впливати на правильність та надійність програми. Це дозволяє покращити якість продукту та забезпечити кращий досвід для користувачів.
- Вимірювання продуктивності, швидкодії та витрат ресурсів (таких як пам'ять та процесорний час). Це важливо аби переконатися, що програма працює ефективно та не використовує забагато ресурсів.
- Визначення, чи може програмне забезпечення працювати стійко та безперебійно в різних умовах. Тестування на надійність включає в себе ідентифікацію та усунення проблем, які можуть виникнути в результаті довготривалого використання або несподіваних ситуацій.
- Перевірка вразливостей та ефективності заходів безпеки для запобігання несанкціонованому доступу, атакам та витокам конфіденційної інформації.
- Перевірка, як програмне забезпечення взаємодіє з іншими компонентами, операційними системами та апаратурою. Визначення сумісності допомагає уникнути проблем зі сумісністю та забезпечити, що програма може працювати на різних платформах.
- Тестування інтерфейсу користувача та інших аспектів зручності використання для розуміння, що програма є зрозумілою та легкою у використанні.

Установка програми полягає в розгортанні БД та установленні виконуваного файлу з кількома супутніми бібліотеками. Далі запускаємо програму запуском виконуваного файлу. Як видно, відкрилися дві порожні таблиці. Третя ж таблиця була вже попередньо заповненою списком наявних рейсів.



RaceID	RaceType	Departure	Destination	DateOfRace
1	Train	Ternopil	Kyiv	01.12.2021
2	Train	Ternopil	Kyiv	02.12.2021
3	Train	Ternopil	Kyiv	03.12.2021
4	Train	Kyiv	Ternopil	01.12.2021
5	Train	Kyiv	Ternopil	02.12.2021
6	Train	Kyiv	Ternopil	03.12.2021
7	Train	Lviv	Lutsk	01.12.2021
8	Train	Lviv	Lutsk	02.12.2021
9	Train	Lutsk	Lviv	01.12.2021
10	Train	Lutsk	Lviv	02.12.2021
11	Night train	Ternopil	Odesa	01.12.2021
12	Night train	Ternopil	Odesa	02.12.2021
13	Night train	Lviv	Kherson	01.12.2021
14	Night train	Lviv	Kherson	02.12.2021
15	Night train	Odesa	Ternopil	01.12.2021
16	Night train	Odesa	Ternopil	02.12.2021
17	Night train	Kherson	Lviv	01.12.2021
18	Night train	Kherson	Lviv	02.12.2021
19	Plane	Kyiv	Dresden	01.12.2021
20	Plane	Kyiv	Dresden	02.12.2021
21	Plane	Kyiv	Dresden	03.12.2021

Рисунок 3.9. Таблиця рейсів



SeatID	Reserved	SeatType	RaceID
1	☐	Second	
2	☐	First	
3	☐	Second	
4	☐	First	
5	☐	Second	
6	☐	First	
7	☐	Second	
8	☐	First	

Рисунок 3.10. Вікно бронювання місця

Щоб додати необхідну інформацію в базу, натискаємо на кнопку «Додати» на головній формі, перед нами з’являється спливаюче вікно в якому заповнюємо поля необхідною інформацією і натискаємо «Додати», опісля чого підтверджуємо дію.

Окрім додавання, в даній програмі застосовується властивість елементу datagridview до редагування записів як з таблиці, а також їх видалення. Для видалення необхідного запису в базі натискаємо безпосередньо на саму стрічку яку бажаємо видалити, після чого тиснемо клавішу delete. Для редагування запису клацаємо двічі по потрібній комірці, після чого змінюємо вміст.

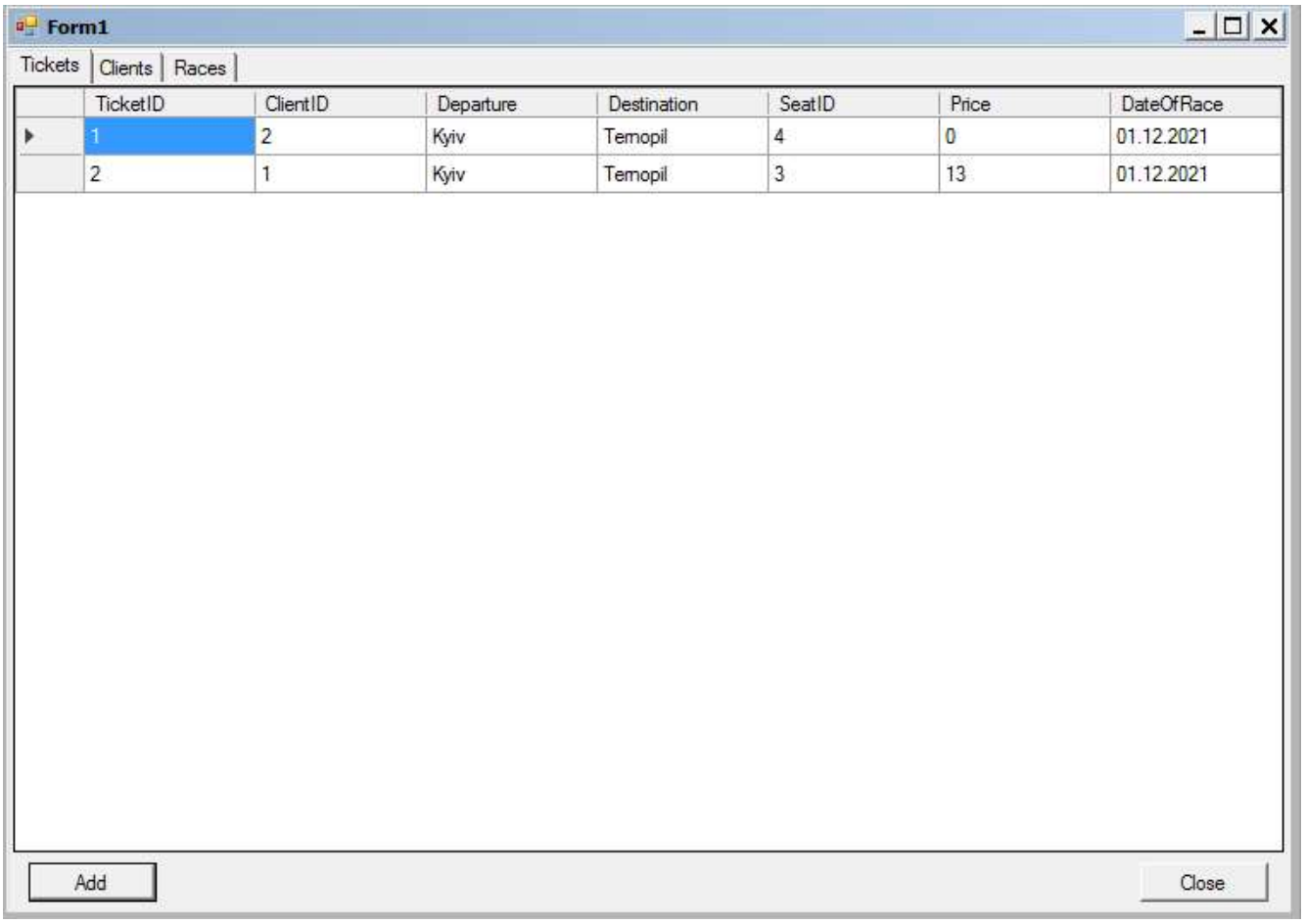


Рисунок 3.11. Частково заповнена таблиця

Аналогічні дії проводимо з таблицею клієнтів.

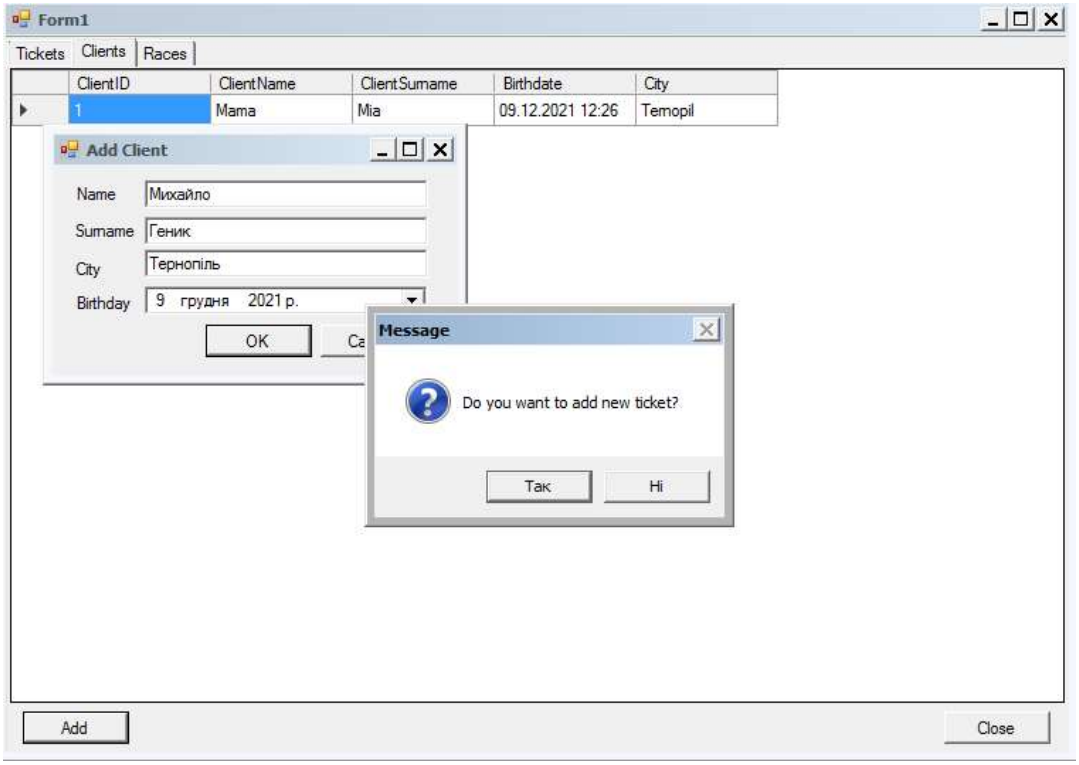


Рисунок 3.12. Підтвердження додання клієнта

3.3. Результат розробки системи бронювання квитків

В результаті атестаційного проєктування було створено програмне забезпечення – базу даних системи бронювання квитків та програмну оболонку до неї. База даних містить чотири таблиці tickets, clients, seats і races, які взаємопов’язані між собою зв’язком “один до багатьох”. Програма доступу до цієї бази даних являє з себе програму перегляду та редагування таблиці з трьома вкладками, одна з яких тільки для читання. Програму спроєктовано з урахуванням об’єктно-орієнтованого підходу та каскадної методології розробки.

Під час тестування програма проявила загальну працездатність та відповідність поставленим вимогам щодо бронювання квитків. Всі виявлені дрібні помилки, виявлені під час тестування, були усунуті.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Охорона праці

Зазвичай процес розробки системи бронювання квитків є довготривалим та виснажливим, оскільки доводиться працювати з багатьма технологіями та великим обсягом даних, через це існує ймовірність припуститися помилки. Усім добре відомо, що одним із головних методів забезпечення ефективної роботи з комп'ютером є забезпечення належних умов праці, так як при недостатньому освітленні, просторі чи неякісному дисплеї, людина відчуватиме фізичний та з часом психологічний дискомфорт, який обов'язково відобразиться на результатах роботи. Для того щоб запобігти цьому, потрібно дотримуватись ряду наказів та стандартів.

В наказі № 207 від 14.02.2018 НПАОП 0.00-7.15-18 [10] описується частина вимог, яких потрібно дотримуватись при роботі з екранними пристроями. Відповідно до третьої частини наказу, робоче місце працівника має відповідати наступним вимогам:

1. Робоче місце має мати достатні розміри, щоб працівник мав простір для зміни робочого положення та рухів.
2. Усе випромінювання від екранних пристроїв має бути знижене до гранично допустимого рівня з погляду безпеки та охорони здоров'я працівників.
3. Усі елементи робочого місця та їх розташування мають відповідати ергономічним, антропологічним, психофізіологічним вимогам, а також характеру виконуваних робіт.
4. Освітлення робочого місця має створювати відповідний контраст між екраном і навколишнім середовищем та відповідати вимогам ДСанПІН 3.3.2.007- 98.
5. Мікроклімат виробничих приміщень має підтримуватись на постійному рівні та відповідати вимогам Санітарних норм мікроклімату виробничих приміщень ДСН 3.3.6.042-99 [11].

6. Робочий стіл чи поверхня повинні бути достатнього розміру та мати поверхню з низькою відбивною здатністю, бути гнучкою під час розміщення екрана, клавіатури, документів чи устаткування.
7. Робоче крісло має бути стійким і дозволяти працівнику легко рухатися та займати зручне положення. Сидіння має регулюватися по висоті, спинка сидіння – по висоті, та з можливістю нахилу. Для зручності слід передбачати підніжку для тих, кому це необхідно.

Також потрібно пам’ятати про безпеку, відповідно до четвертої частини наказу [10], потрібно дотримуватись наступних мінімальних вимог безпеки:

1. Перед початком роботи необхідно очищати екранні пристрої від пилу та інших забруднень.
2. Після закінчення роботи пристрої слід відключати від живлення.
3. При виникнення аварійної ситуації необхідно в той же час відключити пристрій від електричної мережі.
4. Не допускається:
    - Ремонтувати чи виконувати технічне обслуговування, і налагодження екранних пристроїв на робочому місці працівника під час роботи з екранними пристроями;
    - Вимикати захисні пристрої чи проводити зміни у конструкції та складі екранних пристроїв або їх технічне налагодження;
    - Працювати з несправними екранними пристроями, у яких під час роботи виникають нехарактерні сигнали, мигання та інші несправності.
5. Під час виконання робіт з комп’ютером, пов’язаних з нервово-емоційним напруженням мають дотримуватися оптимальні умови мікроклімату відповідно до вимог ДСН 3.3.6.042-99 [11].

Відповідно до третьої частини наказу [10], робоче місце транспортного працівника – користувача системи бронювання квитків має відповідати наступним вимогам:

1. Екранні пристрої не мають бути джерелом ризику для працівників.

2. Усе випромінювання має бути зведене до мінімального рівня з погляду безпеки і охорони здоров’я працівників.
3. Символи на дисплеї мають бути чіткими, відповідного розміру. Між символами і рядками символів повинна бути правильна відстань.
4. Зображення на дисплеї має бути стабільним, без миготінь або інших видів несправності.
5. Яскравість та контрастність символів має легко регулюватися, а також швидко адаптуватися до навколишніх умов.
6. Під час вибору монітора, слід надавати перевагу тим пристроям, які мають можливості повороту та нахилу екрану.
7. При потребі монітор може бути закріпленим на окремому столі чи підставці.
8. При виборі монітора надавайте перевагу дисплеям з матовим покриттям, щоб мінімізувати відблискування або відбивання світла.
9. При виборі клавіатури, слід надавати перевагу тій, яка відкидається і є автономною, щоб працівник міг вибрати зручну робочу позу й уникнути втоми рук.
10. Поверхня клавіатури має бути матовою, щоб уникнути віддзеркалювання.
11. Устаткування, яке входить до робочої станції, не повинно виділяти надлишкового тепла.
12. Під час розробки, вибору, замовлення та модифікації програмного забезпечення, а також під час розробки завдань, що передбачають використання устаткування з екранними пристроями, роботодавець має керуватися таким програмним забезпеченням, яке відповідає розв’язуваним завданням і є простим у використанні, а де необхідно - адаптованим до рівня знань і досвіду працівника.

Отже, для безпечної та ефективної роботи користувача системи бронювання квитків забезпечено належні умови праці, починаючи від робочого місця та його оснащення, та закінчуючи мікрокліматом робочого середовища, відповідно до вимог чинного законодавства.

4.2 Безпека в надзвичайних ситуаціях

Ефективність роботи людини з комп’ютером значною мірою визначається функціональним станом людини. Психофізіологічні та емоційні перенапруження, втома людини-оператора можуть призвести в комп’ютеризованих системах керування до помилок і як наслідок – до значних економічних втрат [14].

Згідно зі статистичними даними від 40 до 75% аварій літаків зумовлено людським фактором [12]. Відмови комп’ютеризованої системи керування рухом залізничного транспорту, на гірничо-збагачувальних комбінатах з вини операторів становлять понад 50% їх загальної кількості, причому значна їх частина спричинена невідповідністю функціонального стану оператора складності виконуваної роботи.

Трудова діяльність користувачів комп’ютерів відбувається у певному виробничому середовищі, яке впливає на їх функціональний стан. Найбільш значимі – фізичні фактори виробничого середовища, до яких належать електромагнітні хвилі різних частотних діапазонів, електростатичні поля, шум, параметри мікроклімату та ціла низка світлотехнічних показників [14].

Трудовий процес суттєво впливає на психофізіологічні можливості користувачів комп’ютерів, оскільки їх діяльність характеризується значними статичними фізичними навантаженнями; недостатньою руховою активністю; напруженнями сенсорного апарату, вищих нервових центрів, які забезпечують функції уваги, мислення, регуляції рухів. Окрім того, трудовий процес користувачів комп’ютерів відзначається значними інформаційними навантаженнями.

Професійні якості та виробничий досвід, які визначають внутрішні засоби діяльності, обумовлюють надійну та безпомилкову діяльність користувачів комп’ютерів, дозволяють знаходити безпечні методи розв’язання виробничих завдань навіть у нестандартних ситуаціях [14].

Зовнішні засоби діяльності, які в основному визначаються ергономічними показниками щодо організації робочого місця, формою та параметрами його елементів, просторового розташування основного і допоміжного устаткування, можуть суттєво знизити фізичні та психофізіологічні навантаження, що діють на користувачів комп’ютерів.

Оскільки робота користувачів комп’ютерів найчастіше проходить за активної взаємодії з іншими людьми, то виникають питання раціоналізації міжособистісних стосунків. Цей комплекс питань порушує як психологічні, так і соціально-психологічні аспекти трудових взаємовідносин, які також є факторами "ризику", що відчутно впливають на функціональний стан користувачів комп’ютерів.

Визначення та вивчення факторів, що впливають на функціональний стан користувачів комп'ютерів дозволить виділити основні причини виникнення станів напруженості, стомлення, стресу і здійснити відповідні профілактичні заходи [14].

Отже, до основних факторів, що впливають на функціональний стан користувачів комп’ютера належать:

1. середовище – характеризується такими шкідливими факторами:

1.2 фізичні: електромагнітні хвилі різних частотних діапазонів, електростатичні поля, шум, параметри мікроклімату та ряд світлотехнічних показників;

1.3 хімічні: пил, шкідливі хімічні речовини, які виділяються при роботі принтера і копіювальної техніки;

1.4 біологічні: підвищений вміст в повітрі патогенних мікроорганізмів, особливо у приміщенні з великою кількістю працюючих, при недостатній вентиляції, особливо у період епідемій;

1.5 психофізіологічні: напруження зору та уваги, інтелектуальні та емоційні навантаження, тривалі статичні навантаження і монотонність праці.

2. трудовий процес - характеризується значними статичними фізичними навантаженнями; недостатньою руховою активністю; напруженнями сенсорного апарату, вищих нервових центрів, які забезпечують функції уваги, мислення, регуляції рухів. Окрім того, трудовий процес користувачів комп'ютерів відзначається значними інформаційними навантаженнями;

3. внутрішні засоби діяльності – це професійні риси та виробничий досвід, які обумовлюють надійну та безпомилкову діяльність користувачів комп'ютерів, дозволяють знаходити безпечні методи розв'язання виробничих завдань навіть у нестандартних ситуаціях;
4. зовнішні засоби діяльності - визначаються ергономічними показниками щодо організації робочого місця, форми та параметрів його елементів, просторового розташування основного і допоміжного устаткування, які можуть суттєво знизити фізичні та психофізіологічні навантаження, що діють на користувачів комп'ютерів;
5. соціально-психологічні фактори трудових взаємовідносин.

У професійних операторів частіше зустрічаються порушення органів зору, опорно-рухового апарату, центральної нервової, серцево-судинної, імунної та статевої систем, захворювання шкіри. Зафіксована значна кількість скарг операторського персоналу на загальне недомагання, передчасне стомлювання, головний біль, порушення функцій органів зору, які здійснювали несприятливий психофізіологічний вплив на самопочуття та працездатність операторів [14].

Сучасна професія користувача ВДТ належить до розумової праці, яка характеризується: високою напруженістю зорових функцій; одноманітною позою; великою кількістю стереотипних висококоординованих рухів, що виконуються лише м'язами кистей рук на фоні малої загальної рухової активності; значним нервовоемоційним компонентом, особливо в умовах дефіциту часу; роботою з великими масивами інформації, що викликає активізацію уваги та інших вищих психічних функцій. Крім того, при роботі з дисплеями на електронно-променевих трубках виникає вплив на користувача цілої низки факторів фізичної природи — електростатичні поля, радіочастотне та рентгенівське випромінювання тощо [14].

Діяльність професіоналів можна поділити на три групи:

1. Діяльність, яка пов'язана з виконанням нескладних багаторазово повторюваних операцій, що не вимагають великого розумового

напруження. Наприклад, робота операторів комп'ютерного набору, працівників довідкових служб.

2. Діяльність, яка пов'язана із здійсненням логічних операцій, що постійно повторюються. Це робота інженера-економіста, інженера-проектувальника, оператора автоматизованого виробництва.
3. Діяльність, коли в процесі роботи необхідно приймати рішення за відсутності заздалегідь відомого алгоритму. Наприклад, робота інженера програміста, диспетчерів руху залізничного транспорту, аеропортів тощо.

У користувачів, які інтенсивно використовують комп'ютер в умовах значних розумових напружень досить часто (40—70%) виникають психологічні та поведінкові порушення (нервозність, роздратування, тривога, нерішучість, замкнутість тощо). Серед користувачів ВДТ в США і Європі значного поширення набуло специфічне захворювання, яке отримало назву синдром комп'ютерного стресу (СКС). СКС супроводжується головним болем, запаленням очей, алергією, роздратованістю, млявістю і депресією. Інформаційне перевантаження користувачів ВДТ супроводжується низкою специфічних захворювань, які називають інформаційними. Першим симптомом їх є головний біль. Дослідження, проведені в США, Німеччині, Швейцарії та інших країнах, показали, що робота з обслуговування ВДТ супроводжується підвищеним напруженням зору, інтенсивністю і монотонністю праці, збільшенням статичних навантажень, нервово-психічним напруженням, впливом різного виду випромінювань та ін.

Внаслідок цього серед операторів ВДТ, як зазначають фахівці Всесвітньої організації охорони здоров'я, частіше, ніж в інших групах працюючих, трапляються такі професійні захворювання, як передчасна стомлюваність, погіршення зору, м'язові і головні болі, психічні й нервові розлади, хвороби серцево-судинної системи, онкологічні захворювання та ін. Вважається, що стан організму операторів ВДТ визначається комплексним впливом факторів трудового процесу і середовища, значення яких є неоднаковим. На операторів з малим стажем роботи на ВДТ домінуючий вплив чинять фактори середовища, а на операторів зі стажем понад 5 років - фактори трудового процесу. Комп'ютерний зоровий синдром

(КЗС) - комплекс порушень здоров'я, який може виникати у користувачів персональних комп'ютерів (ПК) [13]. У користувачів ПК дуже поширені кон'юнктивіти і блефарити, патогенетично пов'язані з КЗС.

Синдром розвивається при умові, що робоче місце організовано неправильно – у користувача незручне крісло, відсутні пюпітри для паперів, підставки для ніг та кистей рук, не встановлена висота і нахил монітора відносно очей, відстань від очей до екрана. За таких умов тіло людини при роботі займає вимушене положення: спина статично напружена, шия витягнута, плечі жорстко фіксовані. Напружені м'язи погіршують кровотік у сонних артеріях, а недостатнє кровозабезпечення головного мозку веде до очманіння, появи головного болю. На фоні шийного остеохондрозу з'являється відчуття випирання очних яблук, туману в очах, мушок та райдужних кіл у полі зору. Розвитку КЗС сприяє поганий мікроклімат приміщення, значна загальна іонізація та мікробне забруднення, а також куріння [14].

Національною радою з наукових досліджень США для стану зорового дискомфорту був уведений термін "астенопія", який означає "будь-які суб'єктивні зорові симптоми чи емоційний дискомфорт, що є результатом зорової діяльності". Симптоми астенопії були класифіковані на "очні" (біль, печія та різь в очах, почервоніння повік та очних яблук, ломота у надбрівній частині тощо) та "зорові" (пелена перед очима, мерехтіння, швидка втома під час зорової роботи та ін.).

Таким чином, на користувача комп'ютера впливає комплекс факторів. Урахування ступеня та якості впливу цих факторів на функціональний стан дозволяють розробити заходи та засоби щодо забезпечення безпеки, підвищення працездатності та збереження здоров'я користувачів комп'ютерів [14].

ВИСНОВКИ

В ході виконання роботи мовою C# в середовищі Visual Studio 2019 мною створено систему бронювання квитків. Система складається з серверної частини – реляційної бази даних на базі СКБД MySQL, та клієнтської – графічної оболонки на основі Windows Forms як частини .NET Framework. Також продемонстровано на прикладі можливості наведених інтегрованих середовищ розробки Microsoft Visual Studio та MySQL Workbench та мови моделювання UML.

Додаток проєктувався згідно вимог, за допомогою методології водоспаду, що найкраще підходить для виконання простих проєктів. Написана програма виконує усі потрібні функції.

- Створено підключення БД до Windows Forms.
- Зроблено форми для введення даних в БД.
- Організовано запис та зчитування даних.

Ця програма відповідає наступним вимогам:

- має інтуїтивно зрозумілий інтерфейс, є простою у використанні;
- має актуальну базу та алгоритми роботи з нею;
- є правильно спроєктована, правильно використовує ресурси пристрою, є не ресурсомісткою.

Розроблена програма повністю відповідає технічному завданню, умови якого вказані в додатку А. Під час тестування програми неполадок не виявлено, а виявлені дрібні помилки були оперативно усунуті. Демонстраційна версія додатку працює як слід.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Коноваленко І. В. Програмування мовою C# 7.0 : навчальний посібник / І. В. Коноваленко, П. О. Марущак, В. Б. Савків. – Тернопіль : ТНТУ ім. І. Пулюя, 2017. – 300 с.

2. Петрик М. Р., Мудрик І. Я. Проєктування програмного забезпечення на основі об’єкнто-орієнтованого аналізу вимог та інструментальних засобів розробки IBM Rational Software Architect / М. Р. Петрик, І. Я. Мудрик – Тернопіль : ТНТУ ім. І. Пулюя, 2022. – 56 с.

3. Об'єктно–орієнтоване програмування (ООП) [Електронний ресурс]. Програмування по українськи. – 2011 – Режим доступу до ресурсу: http://programming.in.ua/programming/basisprogramming/25–oop.html.

4. C# для початківців. [Електронний ресурс]. Все для програмування. – 2019 – Режим доступу до ресурсу: http://programer.in.ua/index.php/pochatkivtsiu/rozrobka–ihor–dlia–pochatkivtsiv–na–c/200–urok1–c–windows–forms.

5. Проектування бази даних. [Електронний ресурс]. Інформаційно–освітній портал кафедри інноваційних та інформаційних технологій в освіті Навчально–наукового інституту педагогіки, психології, підготовки фахівців вищої кваліфікації Вінницького державного педагогічного університету імені Михайла Коцюбинського. – 2015 – Режим доступу до ресурсу: http://ito.vspu.net/ENK/2015–2016/Programming_SQL/rob_stud/lucenko/ctvor_BAZU.htm – 2016/

6. Introduction to .NET Framework [Електронний ресурс]. GeeksforGeeks – 2023 – Режим доступу до ресурсу: https://www.geeksforgeeks.org/introduction-to-net-framework/.

7. Introduction to Visual Studio [Електронний ресурс]. GeeksforGeeks – 2023 – Режим доступу до ресурсу: https://www.geeksforgeeks.org/introduction-to-visual-studio/.

8. What Is MySQL and How Does It Work [Електронний ресурс]. Hostinger – 2023 – Режим доступу до ресурсу: https://www.hostinger.com/tutorials/what-is-mysql.

9. What is RDBMS [Електронний ресурс]. DevArt – 2023 – Режим доступу до ресурсу: https://www.devart.com/what-is-rdbms/.

10. Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями [Електронний ресурс]. Верховна Рада України – 2018. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0508-18#Text.

11. Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99 [Електронний ресурс]. Верховна Рада України – 1999. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/va042282-99#Text.

12. Основні причини аварій літаків [Електронний ресурс]. Суспільне – 2020. – Режим доступу до ресурсу: https://suspilne.media/7904-letiti-bez-strahu-vidpovidaemo-na-najposirenisi-zapitanna-pro-aviakatastrofi/.

13. Комп'ютерний зоровий синдром [Електронний ресурс]. Linza UA – 2018. – Режим доступу до ресурсу: https://linza.com.ua/uk/articles/blog/kompyuternyy-zritelnyy-sindrom-simptomy-i-lechenie/.

14. Стручок В. С. Методичний посібник для здобувачів освітнього ступеня «магістр» всіх спеціальностей денної та заочної (дистанційної) форм навчання «БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ» / В.С. Стручок – Тернопіль: ФОП Паляниця В. А., –156 с. Режим доступу до документу: https://elartu.tntu.edu.ua/handle/lib/39196.

ДОДАТКИ

ДОДАТОК А

Лістинги коду

Лістинг 1. Скрипт для створення бази даних системи

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET           @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET                                  @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN
_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUB
STITUTION';

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
DROP SCHEMA IF EXISTS `mydb` ;

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET
utf8 ;
USE `mydb` ;

-- -----------------------------------------------------
-- Table `mydb`.`Clients`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`Clients` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Clients` (
  `ClientID` INT NOT NULL AUTO_INCREMENT,
  `ClientName` TINYTEXT NULL DEFAULT NULL,
  `ClientSurname` TINYTEXT NULL DEFAULT NULL,
  `Birthdate` DATETIME NULL DEFAULT NULL,
  `City` TINYTEXT NULL DEFAULT NULL,
  PRIMARY KEY (`ClientID`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4;


-- -----------------------------------------------------
```

```
-- Table `mydb`.`Races`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`Races` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Races` (
  `RaceID` INT NOT NULL AUTO_INCREMENT,
  `RaceType` TINYTEXT NULL DEFAULT NULL,
  `Departure` TINYTEXT NULL DEFAULT NULL,
  `Destination` TINYTEXT NULL DEFAULT NULL,
  `DateOfRace` DATETIME NULL DEFAULT NULL,
  PRIMARY KEY (`RaceID`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4;


-- -----------------------------------------------------
-- Table `mydb`.`Seats`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`Seats` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Seats` (
  `SeatID` INT NOT NULL AUTO_INCREMENT,
  `Reserved` TINYINT(0) ZEROFILL NULL DEFAULT NULL,
  `SeatType` TINYTEXT NULL DEFAULT NULL,
  `RaceID` INT NULL DEFAULT NULL,
  PRIMARY KEY (`SeatID`),
  INDEX `fk_Seats_Races1_idx` (`RaceID` ASC) VISIBLE,
  CONSTRAINT `fk_Seats_Races1`
    FOREIGN KEY (`RaceID`)
    REFERENCES `mydb`.`Races` (`RaceID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4;


-- -----------------------------------------------------
-- Table `mydb`.`Tickets`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`Tickets` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Tickets` (
  `TicketID` INT NOT NULL AUTO_INCREMENT,
  `ClientID` INT NULL DEFAULT NULL,
  `Departure` TINYTEXT NULL DEFAULT NULL,
```

```
  `Destination` TINYTEXT NULL DEFAULT NULL,
  `SeatID` INT NULL DEFAULT NULL,
  `DateOfRace` DATETIME NULL DEFAULT NULL,
  `Price` FLOAT NULL DEFAULT NULL,
  PRIMARY KEY (`TicketID`),
  INDEX `fk_Tickets_Clients_idx` (`ClientID` ASC) VISIBLE,
  INDEX `fk_Tickets_Seats1_idx` (`SeatID` ASC) VISIBLE,
  CONSTRAINT `fk_Tickets_Clients`
    FOREIGN KEY (`ClientID`)
    REFERENCES `mydb`.`Clients` (`ClientID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Tickets_Seats1`
    FOREIGN KEY (`SeatID`)
    REFERENCES `mydb`.`Seats` (`SeatID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Лістинг 2. Обробники подій в основному вікні

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

    private void Form1_Load(object sender, EventArgs e)
    {
        this.racesTableAdapter1.Fill(this.mydbDataSet.races
);
        this.clientsTableAdapter1.Fill(this.mydbDataSet.cli
ents);
        this.ticketsTableAdapter1.Fill(this.mydbDataSet.tic
kets);

        dataGridViewTickets.Refresh();
        dataGridViewClients.Refresh();
```

```
            dataGridView1.Refresh();
        }

        private void btnAdd_Click(object sender, EventArgs e)
        {
            Form add = new AddTicket();
            add.Owner = this;
            add.Show();
        }

        private void btnAddClient_Click(object sender,
EventArgs e)
        {
            Form cli = new AddClient();
            cli.Owner = this;
            cli.Show();
        }

        private void Close2_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void Close1_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void Form1_FormClosing(object sender,
FormClosingEventArgs e)
        {
            try
            {
                string msg = "Do you want to save the data?";
                string caption = "Save";
                MessageBoxButtons buttons =
MessageBoxButtons.YesNo;
                MessageBoxIcon ico = MessageBoxIcon.Question;

                DialogResult result;
                result = MessageBox.Show(this, msg, caption,
buttons, ico);

                if (result == DialogResult.Yes)
                {
```

```
                this.clientsBindingSource.EndEdit();
                this.clientsTableAdapter1.Update(this.mydbD
ataSet.clients);

                this.ticketsTableAdapter1.Fill(this.mydbDat
aSet.tickets);

                this.clientsBindingSource.EndEdit();
                this.clientsTableAdapter1.Update(this.mydbD
ataSet.clients);

                this.clientsTableAdapter1.Fill(this.mydbDat
aSet.clients);

                mydbDataSet.WriteXmlSchema("mydbDataSet.xsd
");

                MessageBox.Show("Data saved", "Saved",
MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            else
            {
                return;
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Data not saved" +
ex.Message.ToString(), "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
    }
}
```

Лістинг 3. Обробники подій у вікні додання квитків

```
public partial class AddTicket : Form
{
    public AddTicket()
    {
        InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
```

```
        this.Close();
    }

    private void button2_Click(object sender, EventArgs e)
    {
        try
        {
            string msg = "Do you want to add new ticket?";
            string caption = "Message";
            MessageBoxButtons buttons =
MessageBoxButtons.YesNo;
            MessageBoxIcon ico = MessageBoxIcon.Question;

            DialogResult result;
            result = MessageBox.Show(this, msg, caption,
buttons, ico);

            if (result == DialogResult.Yes)
            {
                try
                {
                    numericUpDownClientID.Update();
                    numericUpDownRaceID.Update();
                    numericUpDownSeatID.Update();
                    numericUpDownPrice.Update();

                    Form1 main = this.Owner as Form1;
                    DataRow Row =
main.mydbDataSet.tickets.NewRow();
                    int rc =
main.dataGridViewTickets.RowCount + 1;
                    int race =
Convert.ToInt32(numericUpDownRaceID.Value) - 1;

                    Row[0] = rc;
                    Row[1] = numericUpDownClientID.Value;
                    Row[2] =
main.mydbDataSet.races.Rows[race][2];
                    Row[3] =
main.mydbDataSet.races.Rows[race][3];
                    Row[4] = numericUpDownSeatID.Value;
                    Row[5] =
main.mydbDataSet.races.Rows[race][4];
                    Row[6] = numericUpDownPrice.Value;
```

```
                    main.mydbDataSet.tickets.Rows.Add(Row);
                    main.mydbDataSet.tickets.AcceptChanges(
);
                    ticketsTableAdapter1.Update(main.mydbDa
taSet.tickets);
                    main.dataGridViewTickets.Refresh();

                    this.Close();
                }
                catch (Exception ex)
                {
                    string err = "Fill all fields
properly!" + '\n' + ex;
                    string c = "Error";
                    buttons = MessageBoxButtons.OK;
                    ico = MessageBoxIcon.Question;

                    result = MessageBox.Show(this, err, c,
buttons, ico);
                }
            }
            else
            {
                this.Close();
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Data not saved" +
ex.Message.ToString(), "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
    }

    private void AddTicket_Load(object sender, EventArgs e)
    {
        ticketsTableAdapter1.Fill(this.mydbDataSet.tickets)
;
        clientsTableAdapter1.Fill(this.mydbDataSet.clients)
;
        seatsTableAdapter1.Fill(this.mydbDataSet.seats);

    }
```

```
    private void numericUpDownRaceID_ValueChanged(object
sender, EventArgs e)
    {
        Form1 main = this.Owner as Form1;
        int race =
Convert.ToInt32(numericUpDownRaceID.Value) - 1;

        this.labelDep.Text = "Departure: " +
main.mydbDataSet.races.Rows[race][2];
        this.labelDes.Text = "Destination: " +
main.mydbDataSet.races.Rows[race][3];
        this.labelDat.Text = "Date: " +
main.mydbDataSet.races.Rows[race][4];
        this.labelTra.Text = "Transport: " +
main.mydbDataSet.races.Rows[race][1];
    }
}
```

Лістинг 4. Обробники подій у вікні додання клієнтів

```
public partial class AddClient : Form
{
    public AddClient()
    {
        InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        this.Close();
    }

    private void button2_Click(object sender, EventArgs e)
    {
        try
        {
            textBoxName.Update();
            textBoxSur.Update();
            textBoxCity.Update();
            dateTimePicker1.Update();

            string msg = "Do you want to add new client?";
            string caption = "Message";
            MessageBoxButtons buttons =
MessageBoxButtons.YesNo;
```

```
            MessageBoxIcon ico = MessageBoxIcon.Question;

            DialogResult result;
            result = MessageBox.Show(this, msg, caption,
buttons, ico);

            if (result == DialogResult.Yes)
            {
                try
                {
                    if (textBoxName.Text != "" ||
textBoxSur.Text != "" || textBoxCity.Text != "")
                    {
                        Form1 main = this.Owner as Form1;
                        DataRow nRow =
main.mydbDataSet.clients.NewRow();
                        int rc =
main.dataGridViewClients.RowCount + 1;

                        nRow[0] = rc;
                        nRow[1] = textBoxName.Text;
                        nRow[2] = textBoxSur.Text;
                        nRow[3] = dateTimePicker1.Value;
                        nRow[4] = textBoxCity.Text;

                        main.mydbDataSet.clients.Rows.Add(n
Row);
                        main.mydbDataSet.clients.AcceptChan
ges();
                        main.dataGridViewClients.Refresh();

                        this.Close();
                    }
                }
                catch (Exception ex)
                {
                    string err = "Fill all fields
properly!" + '\n' + ex;
                    string c = "Error";
                    buttons = MessageBoxButtons.OK;
                    ico = MessageBoxIcon.Question;

                    result = MessageBox.Show(this, err, c,
buttons, ico);
                }
```

```
            }
            else
            {
                this.Close();
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Data not saved" +
ex.Message.ToString(), "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
    }
}
```

ДОДАТОК Б

Тези конференції

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

XI НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

13-14 грудня 2023 року

ТЕРНОПІЛЬ
2023

УДК 004.4

Геник М.І.
Тернопільський національний технічний університет ім. І. Пулюя, Тернопіль

РОЗРОБКА СИСТЕМИ БРОНЮВАННЯ КВИТКІВ НА БАЗІ ТЕХНОЛОГІЙ MYSQL ТА .NET

Henyk M.I.

DEVELOPMENT OF THE TICKET BOOKING SYSTEM BASED ON MYSQL TA .NET TECHNOLOGIES

Розвиток комп'ютерних систем дав змогу автоматизувати роботу різноманітних професій в транспортній галузі, серед яких досить важливу роль займає продаж квитків. З поширенням використання баз даних стало можливим зменшити кількість паперової бухгалтерії при оформленні квитків та перевести даний процес в електронну форму. Це, в свою чергу, зменшило час оформлення квитка та створило можливість передачі даних про оформлені квитки на інші станції маршруту та до головного офісу впродовж доби, коли їх було оформлено. Внаслідок цього підвищується ефективність роботи транспортних підприємств та економляться їхні ресурси.

Проте аби максимально підвищити ефективність використання наявних ресурсів та максимізувати ефективність самої системи потрібно також правильно обрати технології для втілення проєкту цієї системи. Відповідно обраними технологіями стали MySql від Oracle Technologies та .Net від Microsoft Corporation. Перевагою MySql є простота використання та дешевизна розробки на ньому, тоді як перевагами .Net є широке поширення його серед комп'ютерних систем та відповідна кросплатформовість, а також безкоштовність розробки на ньому. Ще одною відчутною перевагою фреймворку .Net є наявність власного інтерфейсу програмування для графічних застосунків Windows Forms, що дозволяє створити високоефективні додатки з власним графічним інтерфейсом користувача (GUI).

Це все загалом дозволяє ефективно створити високопродуктивну систему, використавши мінімальну кількість ресурсів, яка є у наявності. Ключові слова: бази даних, MySql, .Net, продаж квитків.

Література

1. Сфера застосування БД, класифікація БД. [Електронний ресурс]. Учбові Матеріали. – URL: http://um.co.ua/8/8-5/8-52848.html/. Процитовано 5 грудня 2023.
2. MySQL advantages and disadvantages. [Електронний ресурс]. W3Schools Blog. – URL: https://www.w3schools.blog/mysql-advantages-disadvantages. – 2023.
3. George Toman, Explore the Pros and Cons of .NET Development. [Електронний ресурс]. Code4Nord. – URL: https://www.code4nord.com/pros-and-cons-net-development/. – 2023.
4. Що таке Windows Forms. [Електронний ресурс]. Ciksiti. – URL: https://ciksiti.com/uk/chapters/29676-what-is-windows-forms. – 2023.

ДОДАТОК В

Диск з матеріалами кваліфікаційної роботи