ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально–науковий інститут Інформаційних технологій

Кафедра Інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи
на ступень вищої освіти бакалавр

на тему ««**Розробка програмного забезпечення для обліку відвідувачів виставки мовою C# з використанням SQL"»**

Виконав: студент 4 курсу, групи ПД - 42
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності)
Михайленко Вадим Дмитрович
(прізвище та ініціали)
Керівник Гаманюк І.М.
(прізвище та ініціали)
Рецензент
(прізвище та ініціали)

КИЇВ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально–науковий інститут Інформаційних технологій

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти - «Бакалавр»
Спеціальність - 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного
забезпечення
_____О.В.Негоденко
“____” _____ 2023 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Михайленко Вадим Дмитрович
(прізвище, ім’я, по батькові)

1. Тема роботи: «Розробка програмного забезпечення для обліку відвідувачів виставки мовою C# з використанням SQL»
Керівник роботи Гаманюк Ігор Михайлович.
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від «24» лютого 2023 року протокол No26.
2. Строк подання студентом роботи «01» червня 2023 року
3. Вхідні дані до роботи:
3.1.Офіційна документація Microsoft.
3.2.Наукова-технічна література
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
4.1. Введення в облік відвідувачів, аналіз основних понять,
Вивчення потреб і вимог.
4.2. Огляд існуючих засобів для обліку відвідувачів виставки
4.3. Технології та інструменти реалізації.
4.4. Розробка структури програмного забезпечення для обліку відвідувачів виставки.
4.5. Програмна реалізація застосунку
4.6. Висновки
5. Перелік графічного матеріалу

5.1. Титульний слайд
5.2. Мета, об’єкт та предмет роботи
5.3. Задачі дипломної роботи
5.4. Аналіз аналогів
5.5. Вимоги до програмного забезпечення
5.6. Програмні засоби реалізації
5.7. Архітектура застосунку
5.8. Діаграма варіантів використання
5.9. Схема бази даних
5.10. Екранні форми
5.11. Презентація взаємодії з програмою
5.12. Апробація результатів дослідження
5.13. Висновки

6. Дата видачі завдання «25» лютого 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Вибір теми бакалаврської роботи	25.02.2023	Виконано
2	Підбір науково-технічної літератури	26.02.2023 - 07.03.2023	Виконано
3	Дослідження аналогів та актуальності додатку	07.03.2023 – 19.03.2023	Виконано
4	Аналіз та вибір інструментів для розробки додатку	19.03.2023 – 04.04.2023	Виконано
5	Проектування та реалізація	04.04.2023 – 02.05.2023	Виконано
6	Висновки, оформлення роботи	02.05.2023 – 19.05.2023	Виконано
7	Передзахист	24.05.2023	
8	Захист роботи		

Студент _____ Михайленко В.Д.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Гаманюк І.М.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи: 50 с., 20 рис., 2 табл., 11 джерел.

Ключові слова: Програмне забезпечення, Облік, База даних, Visual Studio, C#, WPF, .Net, SQL Server

Об'єкт дослідження – процес обліку відвідувачів виставки.

Предмет дослідження – застосунок з обліку відвідувачів виставки.

Мета дослідження – спрощення процесу обліку відвідувачів виставки.

У роботі проведено аналіз існуючих додатків, таких як Excel, Eventbrite, Cvent Event Management та Cvent Express

В якості сервера БД було взято SQL Server та Entity Framework Core для роботи з нею.

Виконано опис технологій, які були використані у ході роботи, та розробки.

Додаток реалізований за допомогою технології WPF.

Здійснено аналіз сучасних додатків та технологій, які використовуються для обліку відвідувачів

На основі результатів виконаних досліджень розроблено додаток програмне забезпечення для обліку відвідувачів.

Упровадження розробленої програми дозволяє спростити процес обліку відвідувачів виставки.

Сфера Застосування - організації та управління виставками, конференціями, ярмарками і подібними подіями.

ЗМІСТ

ВСТУП

Актуальність теми - На сьогоднішній багато людей можуть спершу подумати, що в часи війни виставки є не на часі, але це не так. Виставки можуть використовуватись не тільки заради розваг, а і як благодійні акції, для привернення уваги і спонукаючи людей приєднуватися до зборів на підтримку армії або постраждалих. І роль програми для обліку відвідувачів виставки є дуже важливою. Програма для обліку відвідувачів допомагає організаторам зібрати інформацію, для оцінки ефективності благодійних виставок. За допомогою зібраної інформації можна аналізувати тенденції, визначити успішні благодійні ініціативи та вдосконалювати майбутні події.

Особливо важлива роль облік відвідувачів відіграє у забезпеченні безпеки. Це відбувається за допомогою встановити ідентичності відвідувачів та контролю доступу до виставки. Також облік відвідувачів під час воєнного стану дозволяє швидко зібрати інформацію про кількість людей, які перебувають на території чи в будівлі в разі виникнення надзвичайних ситуацій. Це важливо для організації евакуації та надання допомоги та пошуку постраждалих.

Об'єкт дослідження – процес обліку відвідувачів виставки.

Предмет дослідження – застосунок з обліку відвідувачів виставки

Мета дослідження – спрощення процесу обліку відвідувачів виставки

Завдання дослідження:

1. Аналіз предметної галузі.
2. Дослідження програмних забезпечень для обліку відвідувачів виставок.
3. Розробка вимог до програмного забезпечення на основі аналізу переваг та недоліків існуючих додатків.
4. Аналізувати технічні засоби, що використовуються для розробки та обрано необхідні для створення надійного додатку.
5. Проектування та розробка нового програмного забезпечення для обліку відвідувачів виставок, який буде мати потрібний функціонал для

використання організаторами виставок та надаватиме зручний доступ до інформації.

6. Налаштування та тестування системи.

Практична значущість результатів дослідження полягає у тому, що розроблене програмне забезпечення може бути застосоване для обліку відвідувачів на будь-яких виставках.

1. ВВЕДЕННЯ В ОБЛІК ВІДВІДУВАЧІВ, АНАЛІЗ ОСНОВНИХ ПОНЯТЬ, ВИВЧЕННЯ ПОТРЕБ І ВИМОГ

Перед початком вивчення потреб і вимог щодо обліку відвідувачів виставок необхідно проаналізувати основні поняття і визначення, що використовуються у даному дослідженні.

Аналіз основних понять допомагає визначити взаємозв'язок між поняттями і встановити логічну структуру дослідження. Він дозволяє ідентифікувати ключові аспекти, які слід враховувати при визначенні потреб і вимог.

Основні поняття і визначення:

- відвідувач: Особа, яка приходить на виставку або подібну подію з метою ознайомлення з продуктами, послугами або інформацією, які представлені на цій події.
- облік відвідувачів: обробка та збереження інформації про виставки та відвідувачів виставок з подальшою їх використання з метою підзвітності, збору статистичних даних, аналізу, оптимізації організації події…
- виставка: Організована подія, на якій публічно представляються різні види продукції, послуги або творів мистецтв з метою привернення уваги аудиторії, встановлення бізнес-контактів та досягнення певних комерційних або не комерційних цілей.
- Адміністратор виставки: Особа, відповідальна за організацію, керування та координацію виставки. Він виконує різноманітні обов'язки, пов'язані з підготовкою та веденням обліку виставки.

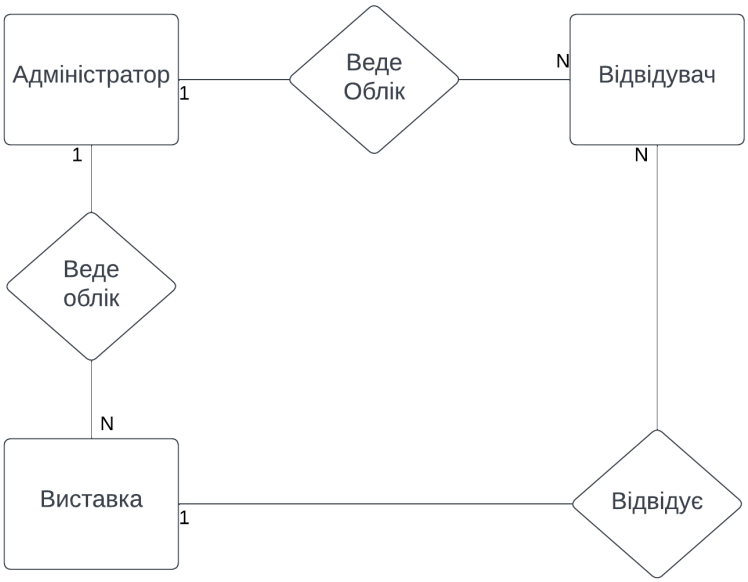


Рисунок 1.1 – Модель предметної галузі

Аналізуючи основні поняття, пов'язані з обліком відвідувачів виставок, було виявлено різноманітні потреби та вимоги:

1) можливість додавати, редагувати та видаляти виставки: Програма повинна мати інтерфейс, що дозволяє користувачам додавати нові виставки до системи, редагувати існуючі та можливість видаляти виставки, які більше не актуальні.
2) можливість додавати, редагувати та видаляти відвідувачів: Програма повинно забезпечувати можливість додавати нових відвідувачів до бази даних, включаючи їх особисті дані, контактну інформацію, час відвідування тощо.
3) налагоджена інтеграція даних між відвідувачами та виставками: Вимога полягає в забезпеченні механізмів для встановлення зв'язку між конкретними відвідувачами і виставками, які вони відвідують. Це означає, що система обліку повинна здатна збирати та обробляти дані з обох джерел, забезпечуючи повний інформаційний контекст.

Об'єктом дослідження є процес обліку відвідувачів виставок. Дослідження спрямоване на вивчення потреб і вимог щодо ефективного обліку, збору та аналізу

даних про відвідувачів з метою покращення швидкості, якості і організації виставкових подій.

Процес обліку включає систематичне фіксування та реєстрацію інформації про відвідувачів, які відвідують виставки з метою ознайомлення з продуктами, послугами або інформацією, що представлені на цих подіях. Облік відвідувачів є важливим аспектом організації виставкових подій.

Його метою є зручна реєстрація відвідувачів та виставок з використанням ефективних інструментів та технологій. Реєстрація включає збір інформації про відвідувачів, таку як особисті дані, контактна інформація, інтереси, а також деталі про виставку, включаючи назву, місце проведення, дату та час.

Після реєстрації відвідувачів та виставок, обробка цих даних стає необхідним етапом. Це включає створення бази даних, упорядкування інформації та забезпечення доступу до неї. Зручна обробка даних дозволяє організаторам ефективно керувати інформацією.

Методика дослідження включає наступні кроки:

1) аналіз вимог до програмного продукту, що включає вивчення особливостей процесу обліку, потреб користувачів, функціональних вимог та вимог до бази даних.
2) проектування бази даних, що включає розробку структури бази даних, яка буде використовуватися для зберігання інформації про відвідувачів та виставки.
3) написання програмного коду на мові C# для реалізації функціональності обліку відвідувачів, забезпечення зв'язку між програмним забезпеченням та базою даних, щоб забезпечити збереження та доступ до даних про відвідувачів та виставки
4) розробка інтерфейсу програми
5) тестування та налагодження, Перевірка правильності роботи програмного забезпечення шляхом виконання тестових сценаріїв та виявлення та усунення помилок

2 ОГЛЯД ІСНУЮЧИХ ЗАСОБІВ ДЛЯ ОБЛІКУ ВІДВІДУВАЧІВ ВИСТАВКИ

2.1 Види засобів для обліку відвідувачів

Засоби обліку відвідувачів можуть бути різноманітними, як застарілими традиційними методами, так і більш сучасними програмними рішеннями, забезпечуючи різні переваги ефективність. Ось декілька засобів обліку відвідувачів виставок:

1) ручний облік на листочках: Це метод, коли організатори виставки використовують листочки або блокноти для реєстрації відвідувачів. Для зручності вони можуть мати спеціальні поля або таблички На кожному листочку можуть бути зазначені основні дані про відвідувача, такі як ім'я, контактна інформація і, можливо, деякі додаткові деталі. Ці листочки пізніше можна обробити, скласти таблицю або використати для створення бази даних.

   Переваги підходу:

   - простота: ручний облік на листочках не вимагає спеціального програмного забезпечення, чи навичок користування. Він є дуже простим в застосуванні будь-кого, незалежно від досвідченості персоналу.
   - низька вартість: цей метод не потребує апаратури або купівлі програмного забезпечення. Це максимально доступний варіант ля будь-якої організації, незалежно від бюджету

   Недоліки підходу

   - великий обсяг роботи: При значній кількості відвідувачів виставки ручний облік на листочках може вимагати додаткових зусиль, часу та ресурсів. Організаторам доведеться вручну заповнювати, зберігати та обробляти

велику кількість листочків, що може бути незручним та призвести до затримок у роботі

- помилки та неточності: При ручному обліку на листочках існує великий потенціал для помилок та неточностей. Організатори можуть допускати помилки при введенні даних, записувати неправильну інформацію або пропускати деякі деталі. Це може призвести до неточного обліку відвідувачів та ускладнити подальшу обробку даних
- обмежені можливості аналітики: Ручний облік на листочках може обмежувати можливості проведення детального аналізу та витягу статистичних даних. Відсутність автоматизованих засобів обробки та аналізу даних ускладнює здійснення глибокого аналізу, виявлення тенденцій та залежностей між різними факторами.
- вразливість даних: Ручний облік на листочках може призводити до рику втрати даних. Листочки можуть бути загублені, пошкоджені або викрадені, що може призвести до втрати цінної інформації та порушення конфіденційності даних

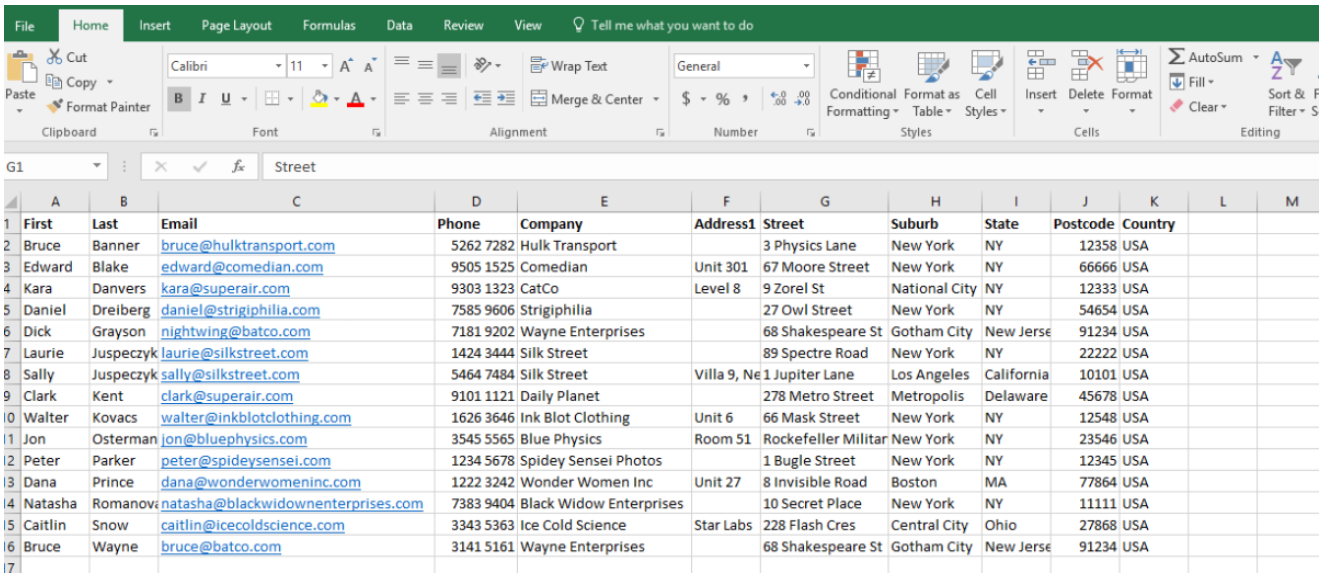
2) електронні таблиці: Багато організаторів виставок використовують електронні таблиці, такі як Microsoft Excel, для обліку відвідувачів. Вони створюють таблиці зі стовпцями, що відповідають різним атрибутам відвідувачів, наприклад, ім'я, прізвище, адреса, контактна інформація тощо. Організатори можуть вручну вводити дані у відповідні комірки таблиці, а потім здійснювати обробку та аналіз цих даних за допомогою функцій Excel.

Переваги підходу:

- більша ефективність і зручність: Використання електронних таблиць дозволяє значно полегшити процес обліку відвідувачів. За допомогою різних функцій та форматування, можна швидко і точно вводити дані про відвідувачів, а також легко редагувати та вносити зміни в існуючі записи. Також, електронні таблиці дозволяють автоматично виконувати розрахунки,

сортувати та фільтрувати дані, що сприяє швидкій і зручній обробці інформації.

- можливості аналізу та звітності: Електронні таблиці надають широкі можливості для аналізу даних та побудови звітів. Вони дозволяють використовувати різні функції, графіки та діаграми для візуалізації інформації та виявлення залежностей. За допомогою фільтрів та сортування, можна швидко відбирати певні групи відвідувачів для подальшого аналізу. Це допомагає організаторам виставок отримувати цінні усвідомлення про процес відвідування та ефективність заходу.
- електронне збереження та резервування даних: Електронні таблиці дозволяють зручно зберігати дані про відвідувачів. Вони можуть бути збережені на комп'ютері, в хмарних сховищах або на сервері, що забезпечує їх безпеку та доступність. Крім того, дані можна регулярно резервувати, що забезпечує захист від втрати інформації у випадку несподіваної поломки апаратного забезпечення або випадкового видалення.
- інтеграція з іншими програмами: Електронні таблиці, такі як Excel, часто підтримують інтеграцію з іншими програмними засобами. Наприклад, дані з електронних таблиць можуть бути експортовані в бази даних або імпортовані в інші програми для подальшої обробки чи аналізу. Це дозволяє розширити функціональність та використовувати дані з обліку відвідувачів в інших контекстах.



	A	B	C	D	E	F	G	H	I	J	K
1	First	Last	Email	Phone	Company	Address1	Street	Suburb	State	Postcode	Country
2	Bruce	Banner	bruce@hulktransport.com	5262 7282	Hulk Transport		3 Physics Lane	New York	NY	12358	USA
3	Edward	Blake	edward@comedian.com	9505 1525	Comedian	Unit 301	67 Moore Street	New York	NY	66666	USA
4	Kara	Danvers	kara@superair.com	9303 1323	CatCo	Level 8	9 Zorel St	National City	NY	12333	USA
5	Daniel	Dreiberg	daniel@strigiphilia.com	7585 9606	Strigiphilia		27 Owl Street	New York	NY	54654	USA
6	Dick	Grayson	nightwing@batco.com	7181 9202	Wayne Enterprises		68 Shakespeare St	Gotham City	New Jerse	91234	USA
7	Laurie	Juspeczyk	laurie@silkstreet.com	1424 3444	Silk Street		89 Spectre Road	New York	NY	22222	USA
8	Sally	Juspeczyk	sally@silkstreet.com	5464 7484	Silk Street	Villa 9, Ne	1 Jupiter Lane	Los Angeles	California	10101	USA
9	Clark	Kent	clark@superair.com	9101 1121	Daily Planet		278 Metro Street	Metropolis	Delaware	45678	USA
10	Walter	Kovacs	walter@inkblotclothing.com	1626 3646	Ink Blot Clothing	Unit 6	66 Mask Street	New York	NY	12548	USA
11	Jon	Osterman	jon@bluephysics.com	3545 5565	Blue Physics	Room 51	Rockefeller Militar	New York	NY	23546	USA
12	Peter	Parker	peter@spideysensei.com	1234 5678	Spidey Sensei Photos		1 Bugle Street	New York	NY	12345	USA
13	Dana	Prince	dana@wonderwomeninc.com	1222 3242	Wonder Women Inc	Unit 27	8 Invisible Road	Boston	MA	77864	USA
14	Natasha	Romanova	natasha@blackwidownenterprises.com	7383 9404	Black Widow Enterprises		10 Secret Place	New York	NY	11111	USA
15	Caitlin	Snow	caitlin@icecoldscience.com	3343 5363	Ice Cold Science	Star Labs	228 Flash Cres	Central City	Ohio	27868	USA
16	Bruce	Wayne	bruce@batco.com	3141 5161	Wayne Enterprises		68 Shakespeare St	Gotham City	New Jerse	91234	USA
17											

Рисунок 2.1 – Приклад таблиці для обліку відвідувачів

3) спеціалізоване програмне забезпечення: Існують програмні забезпечення, розроблені спеціально для обліку відвідувачів виставок. Ці програми забезпечують автоматизовану систему реєстрації, обробки та аналізу даних. Вони можуть містити різну функціональність для вирішення потреб користувачів, наприклад створення електронних реєстраційних форм, відстежування присутності, аналізу даних про аудиторію, генерації бейджів для відвідувачів та інше.

Переваги підходу:

- функціональність: Спеціалізоване програмне забезпечення для обліку відвідувачів виставок має широкий набір функцій та інструментів, спеціально розроблених для потреб організаторів виставок. Це може включати можливості реєстрації відвідувачів, керування даними, створення звітів і аналітики, інтеграцію з іншими системами, що дозволяє ефективно та зручно вести облік відвідувачів, оптимізувати робочі процеси та забезпечувати повну функціональність.

- автоматизація: Спеціалізоване програмне забезпечення дозволяє автоматизувати багато процесів пов'язаних з обліком відвідувачів. Наприклад, система може автоматично генерувати та роздруковувати бейджі для відвідувачів, надсилати автоматичні підтвердження реєстрації по електронній пошті, сканувати QR-коди або штрих-коди для швидкого входу на виставку та інше. Це дозволяє зменшити ручну працю та помилки, покращує ефективність процесу обліку та підвищує задоволення відвідувачів.
- централізоване база даних: Спеціалізоване програмне забезпечення зазвичай має централізоване зберігання даних, для зберігання всіх даних про відвідувачів. Це забезпечує їх доступність та надійність. Всі інформаційні дані можуть бути легко оновлювані, пошукові та аналізовані. Крім того, це дозволяє забезпечити конфіденційність даних та керувати рівнями доступу для різних користувачів.
- інтеграція та сумісність: Спеціалізоване програмне забезпечення для обліку відвідувачів виставок може бути інтегроване з іншими системами та засобами, що використовуються організаторами. Наприклад, з ним можна інтегрувати системи онлайн-реєстрації, системи продажу та бронювання квитків тощо. Це дозволяє автоматизувати обмін даними, забезпечити єдину інформаційну систему та зручність в роботі.

Недоліки підходу:

- вартість: Використання спеціалізованого програмного забезпечення може бути пов'язане зі значними витратами. Це може включати вартість придбання ліцензії, підписки на програму, налаштування та інтеграцію з іншими системами, навчання персоналу та технічну підтримку.
- навчання та впровадження: Використання спеціалізованого програмного забезпечення вимагає певного рівня навичок та навчання. Організаторам виставок може знадобитись час та ресурси на навчання персоналу та впровадження програми. Крім того, перехід зі старих методів обліку до

нового програмного забезпечення може вимагати адаптації та пристосування робочих процесів.

- залежність від технологій: Використання програмного забезпечення означає залежність від IT-інфраструктури. Якщо виникають проблеми з програмним забезпеченням, це може призвести до проблем у роботі обліку відвідувачів.ти.

2.2 Аналіз існуючих аналогів

2.2.1 Eventbrite

Рисунок 2.2 – Логотип Eventbrite

Eventbrite – це американська глобальна платформа для продажу квитків дозволяє користувачам переглядати, створювати та рекламувати заходи.

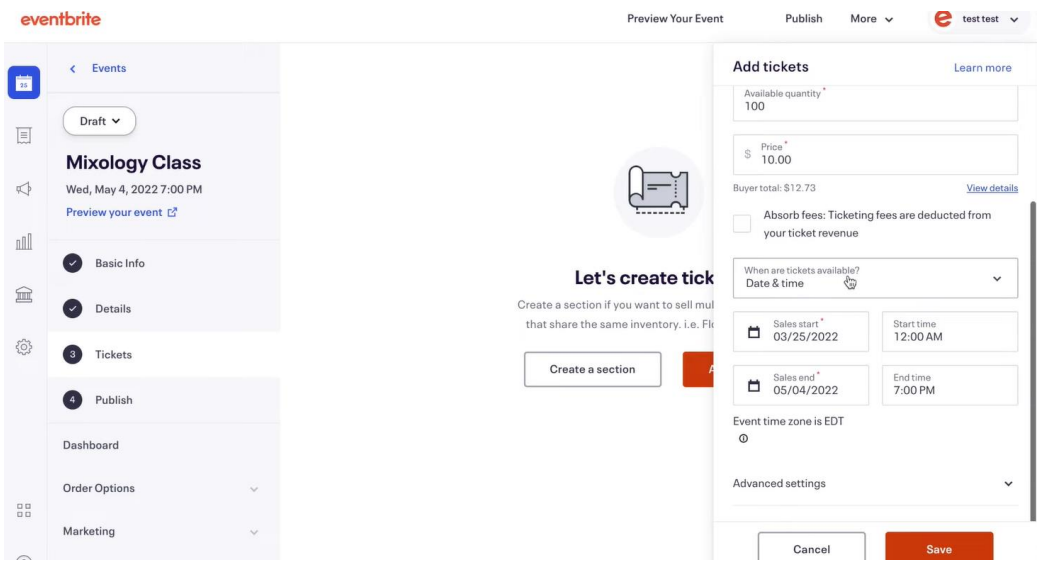


Рисунок 2.3 - Інтерфейс додатку Eventbrite

Eventbrite є популярною платформою для керування подіями та обліку відвідувачів. Які ж в неї основні переваи та недоліки?

Переваги:

- легкість використання: Eventbrite має інтуїтивний і простий інтерфейс, що дозволяє швидко створювати та налаштовувати події.
- продаж квитків онлайн: Платформа надає можливість продавати квитки на події в режимі онлайн, що спрощує процес реєстрації та збільшує охоплення аудиторії..
- інтеграція з соціальними мережами: Eventbrite дозволяє легко поширювати інформацію про події через соціальні мережі та залучати більше учасників.
- аналітика та звітність: Платформа надає інструменти для аналізу даних та генерації звітів про відвідувачів та продажі квитків та інші метрики , що допомагає зрозуміти ефективність подій та покращити маркетингові стратегії.

Недоліки:

- обмежена безкоштовна версія: Хоча Eventbrite має безкоштовний план, він має свої обмеження щодо функціональності, такі як обмежувати кількість подій, які можна створити, інструменти для маркетингу, наприклад запрошення за поштою, а також усі можливості для керування учасниками? Такі як додавання учасників, написання їм листів, перевірка…
- висока комісія: При продажу квитків через Eventbrite, організаторам доводиться платити комісію за кожен проданий квиток, що може вплинути на вартість події та прибутковість.

2.2.2 Cvent

Рисунок 2.4 – Логотип Cvent

Cvent це є публічною компанією програмного забезпечення як послуги, що пропонує програмні рішення для організаторів подій і маркетологів для онлайн-реєстрації подій, вибору місця проведення, управління подіями та маркетингу.

Компанія має два продукти, що надають рішення для упр подіями та реєстрації учасників: Cvent Event Management та Cvent Express. Ось докладнаінформація про кожен з них:

Можливості Cvent Event Management:

- розширена реєстрація учасників: За допомогою Cvent Event Management ви можете створювати настроювані реєстраційні форми для учасників, додавати різні поля для збору інформації, встановлювати ціни, обмеження та умови участі.
- повнофункціональне планування подій: Ви можете створювати детальний розклад подій, включаючи сесії, презентації, кавер-сторінки, обіди та інші активності. Також є можливість керувати локаціями та розміщенням елементів події на інтерактивних картах.
- розширене управління спікерами: Ви можете керувати спікерами та їх пропозиціями для участі у події, включаючи збір біографічних даних, фотографій та інших деталей.
- повнофункціональна промоція подій: За допомогою Cvent Event Management ви можете створювати веб-сайти подій, розсилати електронну пошту учасникам, використовувати інтеграцію з соціальними мережами для просування та привертання учасників.

- розширена звітність і аналітика: Ви отримуєте доступ до різноманітних звітів та аналітики, які допомагають вам оцінити ефективність подій, включаючи статистику участі, фінансову інформацію, дані про спікерів та інше.

Можливості Cvent Express:

- базова реєстрація учасників: Cvent Express дозволяє створювати прості реєстраційні форми для учасників та збирати основний набір інформації, такий як ім'я, контактні дані та преференції.
- базові функції планування: Ви можете створювати основний розклад подій з обмеженими опціями, такими як сесії та перерви.
- базове керування спікерами: Ви можете додавати інформацію про спікерів та їх сесії для відображення на веб-сайті події.
- базова промоція: Cvent Express дозволяє створювати прості веб-сайти подій та надсилати обмежену кількість електронних листів учасникам.
- базова звітність: Ви можете отримати доступ до основних звітів, таких як список учасників, реєстраційні дані та інші базові відомості.

Різниця між Cvent Event Management та Cvent Express полягає в рівні функціональності, обсязі можливостей та ціновій політиці. Cvent Event Management надає широкий спектр функцій та дозволяє керувати великими та складними подіями, тоді як Cvent Express є спрощеною версією з меншим функціоналом і підходить для менших подій або обмежених бюджетів.

Основні переваги Cvent включають:

- реєстрація та управління учасниками: Cvent дозволяє створювати власні сторінки реєстрації, керувати даними про учасників, включаючи персоналізовані поля, керувати розкладом подій і відстежувати присутність учасників.
- маркетинг та просування: Платформа надає інструменти для створення привабливих сторінок подій, розсилки електронних листів, просування в соціальних мережах та інші маркетингові активності для залучення більше учасників.

- управління локаціями та ресурсами: Cvent допомагає вибирати та керувати локаціями для подій, включаючи облік доступних місць, розміщення номерів готелів, вибір кейтерингу та інші деталі.
- мобільні додатки: Платформа надає можливість створювати спеціальні мобільні додатки для подій, що дозволяє учасникам отримувати актуальну інформацію, розклад подій, інтерактивні картки та інші корисні функції.

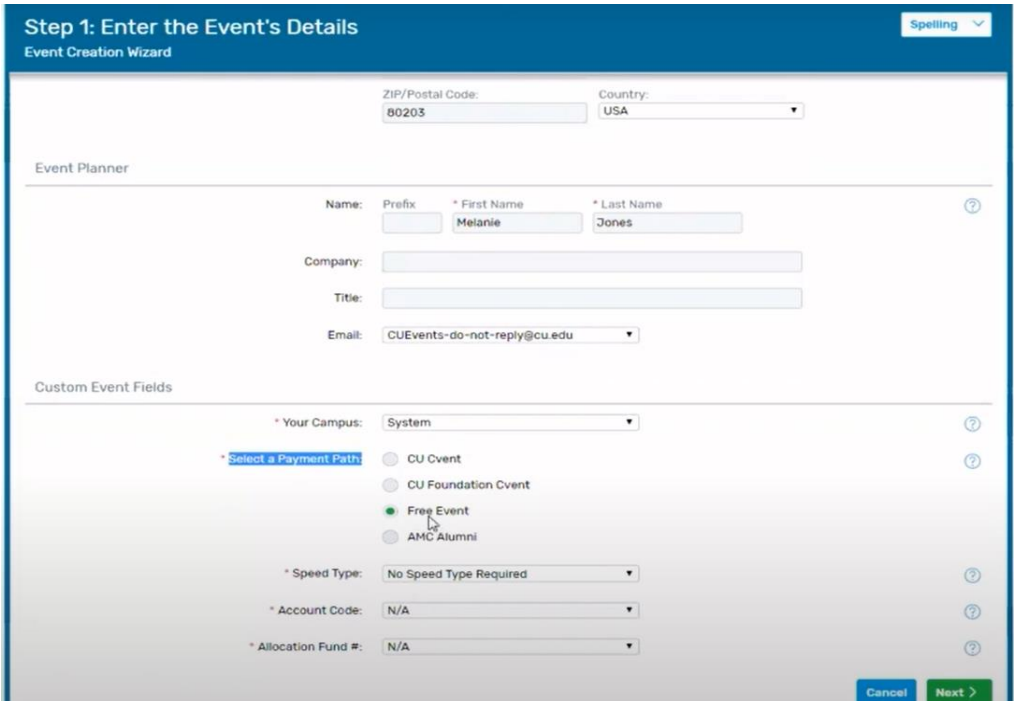


Рисунок 2.5 – Інтерфейс додатку Cvent

Таблиця 2.1 – Результат аналізу розглянутих додатків.

	Excel	**Eventbrite**	**Cvent**	**Розроблена програма**
Українська локалізація	+	-	-	+
Автоматизоване налаштування зберігання даних	-	+	+	+

Продовження таблиці 2.1

	Excel	**Eventbrite**	**Cvent**	**Розроблена програма**
Необмежена кількість подій	+	Лише у про версії	+	+
Автоматизована обробка даних	-	+	+	+
Автоматизована інтеграція даних	-	+	+	+
Модель реалізації	On-premises software	Software as a Service	Software as a Service	On-premises software

На основі аналізу розглянутих аналогів, було виділено основні функції та характеристики, які мають бути реалізовані у розроблюваній програмі для обліку відвідувачів виставки.

Функціональні вимоги:

- Можливість додавати, редагувати та видаляти виставки та відвідувачів: Програма має надавати можливість користувачам управляти відвідувачами та виставками, що проводяться.
- автоматизоване налаштування зберігання даних: Розроблена програма має мати можливість автоматичного налаштування зберігання даних у базі даних. Це забезпечить зручний та надійний механізм збереження інформації про відвідувачів виставок
- автоматизована обробка даних: Програма має забезпечувати можливість додавання нових записів до бази даних шляхом використання спеціальних форм для заповнення даних. Крім того, має бути реалізована зручна зміна та оновлення існуючих даних. Користувачі зможуть швидко знайти необхідну

інформацію за допомогою функції пошуку та експортувати дані в формат Excel для подальшого аналізу

- автоматизована інтеграція даних: Програма має підтримувати інтеграцію даних між таблицями, які стосуються виставок та їх відвідувачів. Це дозволить зручно відстежувати відвідувачів, що зареєструвалися на конкретну виставку.
- модель реалізації - On-premises software: Розроблена програма буде розгортатися як локальне програмне забезпечення (on-premises) на сервері або комп'ютері виставки. Це забезпечить контроль над даними та конфіденційністю, оскільки дані не будуть зберігатися в хмарі.
- можливість пошуку відвідувачів та виставок: Застосунок повинен мати функціонал для пошуку відвідувачів та виставок за різними критеріями, наприклад, за іменем, контактною інформацією або іншими характеристиками. Це дозволить користувачам швидко знаходити потрібних відвідувачів та виставок у системі.
- можливість експортування даних у форматі Excel: Програма повинна мати функцію експорту, яка дозволяє користувачам зберігати дані про виставки та відвідувачів у файлі Excel.

Не функціональні вимоги:

- легкість використання та інтуїтивний інтерфейс: Програма повинна бути простою та зрозумілою і мати легкий у використанні інтерфейс, що дозволить операторам швидко створювати виставки та реєструвати відвідувачів без потреби у вивчені складних інструкцій.
- українська локалізація: Програма повинна підтримувати українську мову для зручного користування та взаємодії з україномовними користувачами.

3 ТЕХНОЛОГІЇ ТА ІНСТРУМЕНТИ РЕАЛІЗАЦІЇ

3.1 Платформа .NET

.NET - це відкрита платформа, яка надає середовище для розробки та виконання програм. Вона має багатий вибір інструментів та бібліотек, що допомагають розробникам ефективно створювати програми різної природи - від десктопних додатків до веб-сайтів та хмарних додатків.

Платформа .NET складається з двох основних компонентів: CLR (Common Language Runtime) і BCL (Base Class Library).

CLR - це середовище виконання, яке забезпечує виконання програмного коду на платформі .NET. Вона відповідає за наступні завдання:

- компіляція та виконання: CLR забезпечує компіляцію програмного коду, написаного на мові програмування C# (або інших мов платформи .NET), в проміжний код, відомий як CIL (Common Intermediate Language). Після цього виконує цей проміжний код на віртуальній машині CLR.
- керування пам'яттю: CLR автоматично вирішує питання виділення та звільнення пам'яті, дозволяючи розробникам уникнути проблем, пов'язаних з ручним керуванням пам'яттю.
- керування типами: CLR забезпечує систему типів, що використовується в .NET. Вона відповідає за завантаження та виконання типів, перевірку типів під час виконання та управління взаємодією між типами.
- обробка виключень: Вона відстежує виключення, керує перехопленням та передачею виключень до відповідних обробників і забезпечує відновлення стану програми після виникнення виключення.

Тож можна виділити такі її плюси як:

- Підтримка декількох мов
- Кросплатформність
- Вбудовані механізми оптимізації

BCL - це стандартна бібліотека класів, які надають базові функціональні можливості для розробки програм на платформі .NET. Вона включає різноманітні класи, інтерфейси та типи даних, які можна використовувати у своїх програмах.

BCL надає реалізацію різних завдань, таких як робота з колекціями, робота з файлами та директоріями, мережеве взаємодія та багато іншого. Вона є фундаментом для розробки програм на платформі .NET і надає розробникам широкі можливості для створення різноманітних додатків.

Обидва ці компоненти є важливою частиною екосистеми .NET та сприяють ефективній та швидкій розробці програмного забезпечення.

Завдяки платформі .NET, розробники мають можливість створювати кросплатформові додатки, які можуть працювати на різних операційних системах, таких як Windows, macOS та Linux. Зокрема, з'явилися такі ініціативи як .NET Core та ASP.NET Core, що дозволяють створювати ефективні та масштабовані веб-додатки, які можуть працювати на різних платформах.

3.2 Мова програмування C#

C# - це строго типізована об'єктно-орієнтована мова програмування, розроблена компанією Microsoft, яка поєднує в собі потужність і гнучкість для створення різноманітних програмних рішень. Вона є однією з основних мов програмування на платформі .NET і широко використовується для розробки різноманітних програм, включаючи настільні програми, веб-застосунки, мобільні додатки та інше.

Ця мова програмування підтримує концепції об'єктно-орієнтованого програмування (ООП), які дозволяють розробникам створювати структуровані та модульні програми, що допомагає у побудові великих, але гнучких, з можливостю розширення, програм.

C# має Сі-подібний синтаксис, схожий на C, C++ та java, що робить її зрозумілою та легким для опанування багатьом розробникам, які мають досвід

роботи з цими мовами. Вона використовує фігурні дужки для визначення блоків коду та точки з комою для розділення операторів.

C# постійно розвивається, і з кожною наступною версією збагачується новими функціональними можливостями.

3.3 Мова програмування SQL

SQL - це мова програмування, призначена для управління реляційними базами даних. Вона надає розробникам значний набір функціональностей для створення, модифікації та операцій з даними в базах даних.

У реляційній моделі СУБД дані організовані у вигляді набору таблиць, а взаємозв'язки між таблицями встановлюються за допомогою ключів. Ключі визначають унікальні ідентифікатори записів у таблицях та використовуються для забезпечення зв'язків між різними таблицями. Це дозволяє ефективно управляти та організовувати великі обсяги даних у базі даних.

Однією з причин популярності SQL є його простота та легкість в освоєнні. Синтаксис SQL базується на зрозумілих людям словах та фразах, що дозволяє розробникам легко виразити свої наміри та завдання щодо роботи з даними. SQL є декларативною мовою, що означає, що розробник описує, що потрібно зробити, а не як це зробити, а виконання операцій покладається на систему управління базою даних (СУБД).. Це робить SQL простим у використанні та зрозумілим.

Крім того, SQL є стандартом, що розробляється та підтримується міжнародною організацією ISO (International Organization for Standardization). Це означає, що SQL може бути використана з різними СУБД, що дозволяє розробникам переносити свій код між різними платформами без необхідності повторного написання.

Мова SQL має стандартний набір команд та операцій, які використовуються для виконання різних завдань, таких як вибірка даних (SELECT), вставка (INSERT), оновлення (UPDATE), видалення (DELETE), створення таблиць та інших об'єктів бази даних (CREATE), зміна схеми бази даних (ALTER) та багато іншого

Все це разом дозволило SQL стати найпопулярнішою мовою для роботи з базами даних

3.4 MS SQL Server

Microsoft SQL Server (MS SQL Server) - це реляційна система управління базами даних, розроблена компанією Microsoft. Вона надає середовище для збереження, керування та обробки даних.

З плюсів Microsoft SQL Serverможна виділити наступні переваги:

- забезпечення стабільної та надійної роботи з базою даних. Він має механізми для забезпечення цілісності даних, виявлення та виправлення помилок, а також механізми резервного копіювання та відновлення для забезпечення безпеки даних.
- підтримка можливості горизонтального та вертикального масштабування, дозволяючи обробляти великі обсяги даних та підтримувати багатопотоковий доступ, що дозволяє збільшувати обсяги даних та кількість оброблюваних запитів без значного падіння продуктивності.
- розширені функціональні можливості для роботи з базами даних. Він підтримує широкий спектр операцій та функцій SQL, таких як складні запити, агрегація даних, з'єднання таблиць, виклик хранимих процедур та багато іншого. Він також має вбудовані інструменти для оптимізації запитів, моніторингу продуктивності та аналізу даних.
- має вбудовані механізми захисту даних, які дозволяють забезпечувати конфіденційність та доступ до даних тільки авторизованим користувачам.

3.5 SQL Server Management Studio

SQL Server Management Studio це програма, розроблена компанією Microsoft, призначеним для адміністрування та керування базами даних Microsoft SQL Server.

SSMS надає такі можливості:

- підключатися до локальних або віддалених серверів SQL Server, використовуючи різні методи аутентифікації
- надає зручний спосіб для створення, зміни та видалення об'єктів бази даних
- дозволяє виконувати SQL-запити безпосередньо з інтерфейсу, що дозволяє вам переглядати та редагувати дані в базі даних

Завдяки графічному інтерфейсу, SQL Server Management Studio дозволяє розробникам і адміністраторам зручно працювати з базами даних та виконувати різноманітні операції.

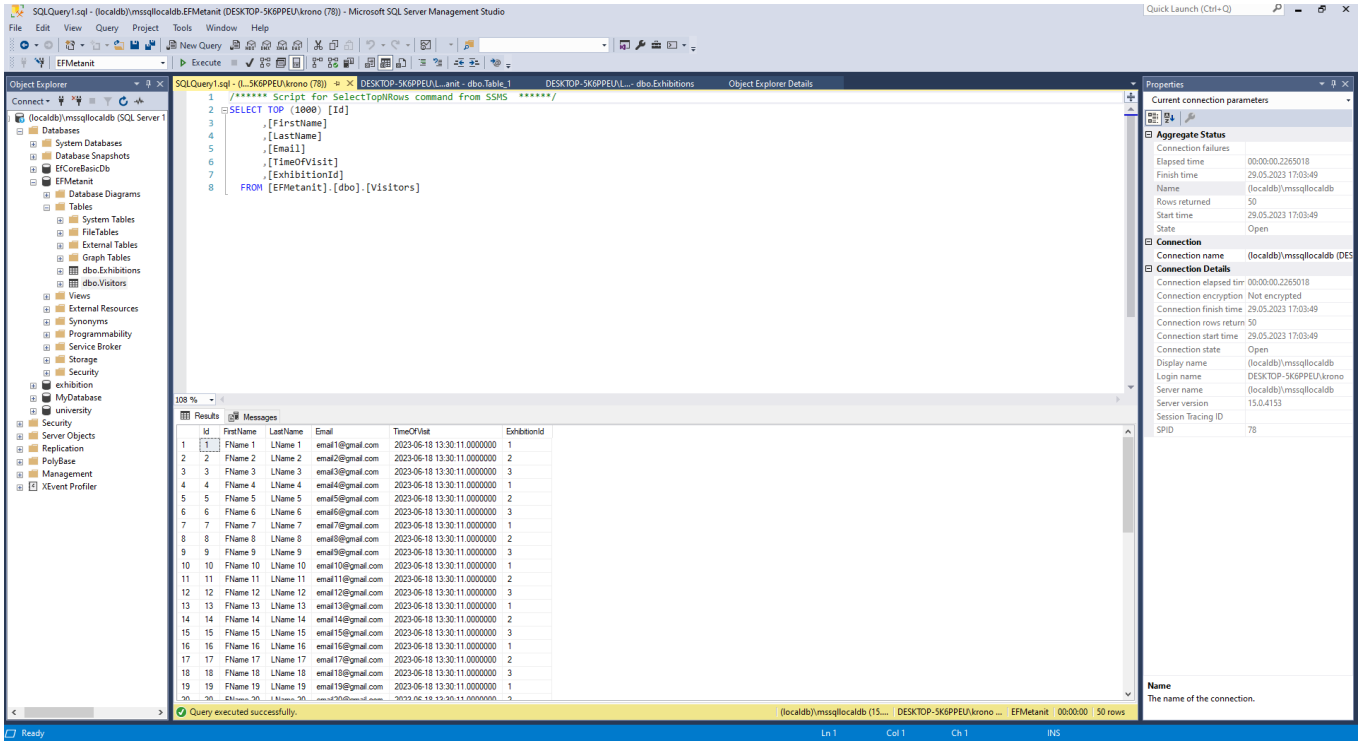


Рисунок 3.1 – Інтерфейс програми SQL Server Management Studio

3.6 Entity Framework Core

Entity Framework Core (EF Core) - це сучасний ORM (Object-Relational Mapping) фреймворк для роботи з базами даних у .NET-розробці. Він надає зручний спосіб взаємодії з реляційними базами даних, використовуючи об'єктно-орієнтований підхід.

Проблема взаємодії між парадигмами ООП та реляційними базами даних полягає в тому, що об'єктно-орієнтовані мови програмування, такі як C#, мають різні парадигми та поняття для роботи з даними, такі як класи, об'єкти. У той час як реляційні бази даних працюють з табличною структурою, запитами SQL і наборами даних.

Entity Framework Core вирішує цю проблему, перетворюючи дані з реляційної бази даних в об'єкти, з якими зручно працювати у контексті ООП. EF Core дозволяє описувати таблиці бази даних за допомогою класів, які називаються класами сутностей. Кожен клас сутності відповідає одній таблиці в базі даних, а властивість класу сутності відповідає певному стовпцю таблиці і містить дані, що зберігаються в цьому стовпці.

У рамках Entity Framework Core (EF Core) надається можливість автоматичного та явного встановлення відповідності між класами, які використовуються у додатку, і таблицями бази даних, що також відоме як маппінг.

- автоматичний маппінг даних використовує конвенції для встановлення відповідності між назвами класів сутностей і таблицями бази даних, а також між назвами властивостей класів сутностей і стовпцями таблиць.
- явний маппінг даних використовує набір атрибутів, для налаштування та вказівки назви таблиці або стовпця, з якою пов’язані клас або властивість. Або використовуючи ланцюжкові методи Fluent API для опису конфігурації та правил маппінгу між об'єктно-орієнтованою моделлю і базою даних.

Такий підхід дозволяє забезпечити зручну взаємодію між об'єктно-орієнтованою парадигмою програмування та реляційними базами даних.

Entity Framework (EF) надає три основні підходи для роботи з даними:

- Code-First: Цей підхід передбачає розробку моделі даних (сутностей) в першу чергу без прямого зв'язку з базою даних. Розробник створює класи сутностей, визначає їх відношення та правила мапінгу до таблиць бази даних за допомогою атрибутів або Fluent API. При першому запуску програми EF Core автоматично створює необхідну базу даних та таблиці з урахуванням визначеної моделі.

- Database-First: Цу підхід до розробки програмного забезпечення, коли спочатку створюється база даних, а потім модель даних генерується автоматично на основі цієї бази. Розробник визначає структуру бази даних, таблиці, колонки, відносини між ними тощо, і на основі цієї структури генерується модель даних. Зазвичай це виконується за допомогою інструментів EF Core для згенерування моделі даних з бази. Це включає створення класів сутностей на основі таблиць, визначення відношень між ними та можливість додавати додаткову логіку до згенерованих класів..
- Model-First: Цей підхід передбачає розробку моделі даних у спеціальному редакторі. Розробник створює сутності, визначає їх властивості, відношення та правила мапінгу. За допомогою інструментів EF Core, база даних та таблиці можуть бути згенеровані на основі визначеної моделі.

Кожен з цих підходів має свої переваги та підходить для різних сценаріїв розробки. Вибір підходу залежить від ваших вимог, наявності бази даних або потреби в моделюванні даних перед створенням бази.

Ця ORM-технологія дозволяє автоматично створювати, змінювати або видаляти таблиці бази даних на основі змін у моделі даних. Розробникам не потрібно вручну писати складні SQL-запити, а замість цього вони можуть працювати з об'єктами та логікою своїх додатків на більш вищому рівні абстракції. Це дозволяє зберігати структуру бази даних у синхронізації з моделлю даних.

Крім того, вона надає розширений набір методів для виконання запитів до бази даних. Це включає фільтрацію, сортування, групування та інші операції. Всі ці методи можна виконувати за допомогою LINQ або методів-розширень.

Тому в цілому EF Core є потужним інструментом для розробки додатків, які взаємодіють з базами даних. Він спрощує роботу з даними, надає вищий рівень абстракції і забезпечує швидку розробку та підтримку додатків.

3.7 Visual Studio

Visual Studio є одним з найпопулярніших інтегрованих середовищ розробки (IDE), розроблених компанією Microsoft. Воно надає розробникам широкий набір інструментів і можливостей для створення різноманітних типів програмного забезпечення, включаючи десктопні додатки, веб-додатки, мобільні додатки та багато інших.

Завдяки Visual Studio розробники можуть працювати над проектами, для різних платформ, включаючи Windows, Android, iOS, веб-додатки та хмарні сервіси. Це дозволяє розробникам працювати над проектами різного типу,

Однією з ключових переваг Visual Studio є його зручність та інтуїтивний інтерфейс. Завдяки багатофункціональному середовищу розробки, Visual Studio надає розробникам широкий спектр інструментів для покращення продуктивності. Його можливості включають вбудовані засоби відлагодження, автоматичне завершення коду та підказки, що полегшують процес написання коду та відлагодження програм.

Під час роботи з Visual Studio, ви можете викликати контекстну довідку, навівши курсор на функцію, клас або інший елемент коду. Це відкриє вікно з коротким описом та прикладами використання цього елемента.

У цьому середовищі розробки програмного забезпечення доступна функція IntelliCode, яка використовує машинне навчання та штучний інтелект для аналізу написаного коду та надання рекомендацій. IntelliCode може пропонувати шаблони коду, автоматичні завершення, підказки та оптимізації, сприяючи покращенню продуктивності та ефективності розробників.

Крім того, Visual Studio має інтеграцію з NuGet - менеджером пакетів для платформи .NET. NuGet дозволяє легко встановлювати, оновлювати та керувати залежностями та сторонніми бібліотеками у проекті. Ви можете швидко шукати та встановлювати NuGet пакети безпосередньо з Visual Studio, що спрощує управління залежностями та розширює можливості вашого проекту

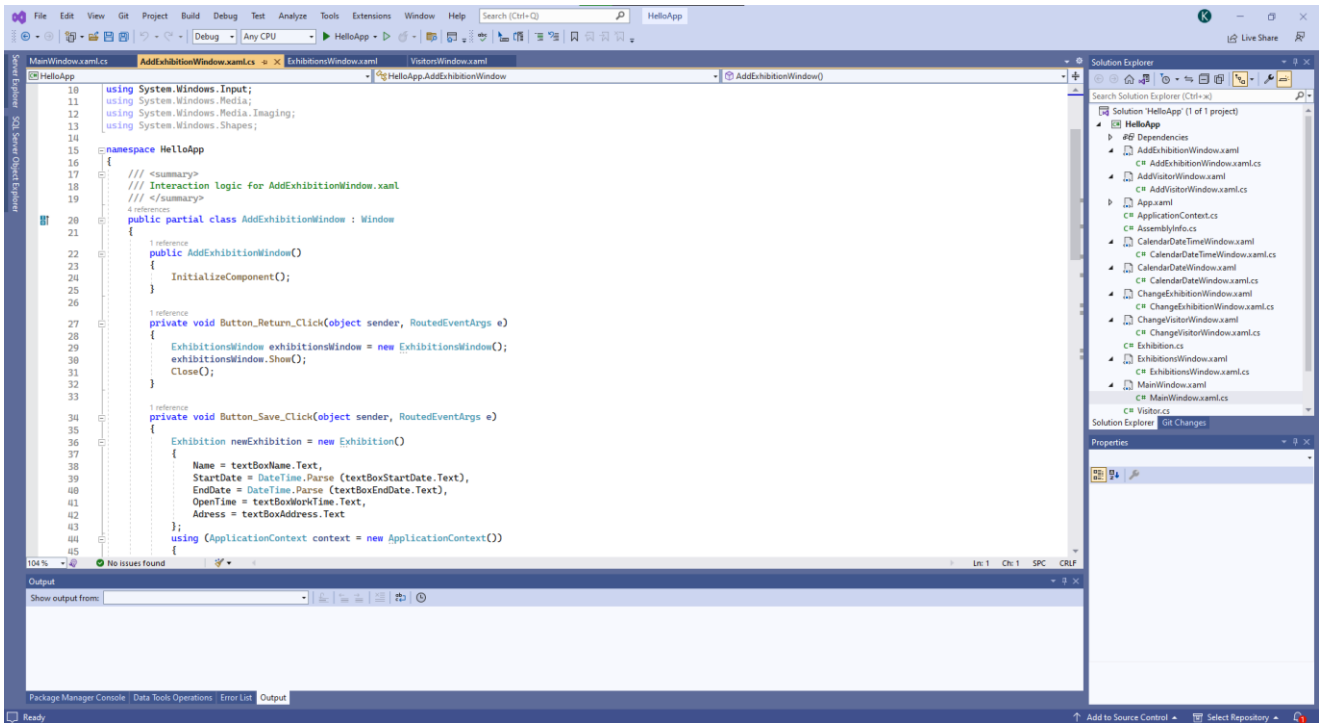


Рисунок 3.2 – Інтерфейс програми Visual Studio 2022

Також Visual Studio надає велику гнучкість у налаштуванні свого інтерфейсу, щоб ви могли адаптувати його до своїх потреб і вимог. Ви можете додавати, видаляти та закріплювати різні інструменти, панелі та вікна в своєму інтерфейсі, щоб забезпечити зручну робочу область.

3.8 WPF

Windows Presentation Foundation (WPF) - це технологія розробки інтерфейсу користувача, яка є частиною платформи .NET. Вона дозволяє розробляти різноманітні програми для Windows з інтерактивним інтерфейсом.

Розробка програм з використанням Windows Presentation Foundation (WPF) передбачає два основні способи побудови вікон та інтерфейсу користувача: декларативний підхід з використанням мови розмітки XAML та інтерактивний підхід з використанням інтерфейсу Visual Studio.

У Windows Presentation Foundation (WPF) є два основних підходи до побудови інтерфейсу користувача:

1) перший підхід полягає в декларативному описі вікон та елементів інтерфейсу за допомогою мови розмітки XAML (eXtensible Application Markup Language). За допомогою XAML, ви можете створювати структуру інтерфейсу, розміщувати елементи, налаштовувати їх властивості та встановлювати взаємодію між ними.
2) другий підхід до роботи з інтерфейсом полягає у використанні інтерактивного інтерфейсу Visual Studio. Ви можете перетягувати та розташовувати елементи з панелі інструментів на вікно, налаштовувати їх властивості безпосередньо у візуальному режимі. Цей підхід дозволяє вам швидко будувати та налаштовувати інтерфейс за допомогою інтуїтивних засобів.

В розробці на WPF можна поєднувати обидва підходи - інтерактивний та декларативний - в залежності від потреб проекту та етапу роботи.

Починаючи розробку, інтерактивний підхід з використанням інтерфейсу Visual Studio дозволяє швидко створити загальну структуру інтерфейсу та розмістити прості елементи шляхом перетягування їх з панелі інструментів на вікно. Це швидкий спосіб прототипування та отримання загального вигляду інтерфейсу. Ви можете швидко експериментувати з розміщенням елементів, змінювати їх розміри та властивості, а все це без необхідності писати власноруч код.

Після створення загальної структури інтерфейсу можна перейти до декларативного підходу, використовуючи XAML. В XAML ви можете детально настроїти вигляд інтерфейсу, задати властивості елементів, визначити їх поведінку та взаємодію. Ви можете додавати анімацію, стилізацію, зв'язування даних та багато інших функцій, що дозволяють створити більш складні та гнучкі інтерфейси.

Така послідовність дозволяє спочатку швидко розмістити та налаштувати елементи за допомогою інтерфейсу Visual Studio, а потім, коли необхідно здійснити

більш глибоке налаштування та контроль над інтерфейсом, переходити до використання XAML. Такий підхід дозволяє поєднувати швидкість та зручність інтерактивного підходу з потужністю та гнучкістю декларативного підходу.

4 РОЗРОБКА СТРУКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОБЛІКУ ВІДВІДУВАЧІВ ВИСТАВКИ

4.1 Моделювання системи

4.1.1 Діаграма прецедентів

Діаграма прецедентів - це графічний інструмент, який допомагає описати функціональність системи з точки зору її користувачів. Вона використовується в аналізі та проектуванні програмного забезпечення для визначення вимог до системи та уточнення її функцій.

Зазначена діаграма сприяє полегшенню комунікації між розробниками програмного забезпечення та замовниками, допомагає уточнити вимоги до системи та визначити необхідний функціонал.

Процес створення діаграми прецедентів сприяє кращому розумінню того, як користувачі будуть взаємодіяти з системою та які конкретні функції вона має надавати. Відображення цих сценаріїв допомагає уточнити потреби користувачів та гарантує успішну реалізацію системи з урахуванням їх очікувань та вимог.

Основні складові діаграми прецедентів:

- актори: Актори представляють ролі або зовнішніх сутностей, які взаємодіють з системою. У випадку програмного забезпечення для обліку відвідувачів виставок акторами можуть бути адміністратори виставок.
- прецеденти: Прецеденти описують конкретні дії або функціональність, яку система надає користувачам. Наприклад, додавання нової виставки, реєстрація відвідувача, генерація звіту тощо..

Мета створення діаграми прецедентів полягає у визначенні основних сценаріїв взаємодії з системою. Діаграма прецедентів дозволяє ідентифікувати ключові функції, які система повинна надавати для задоволення потреб адміністраторів виставок.

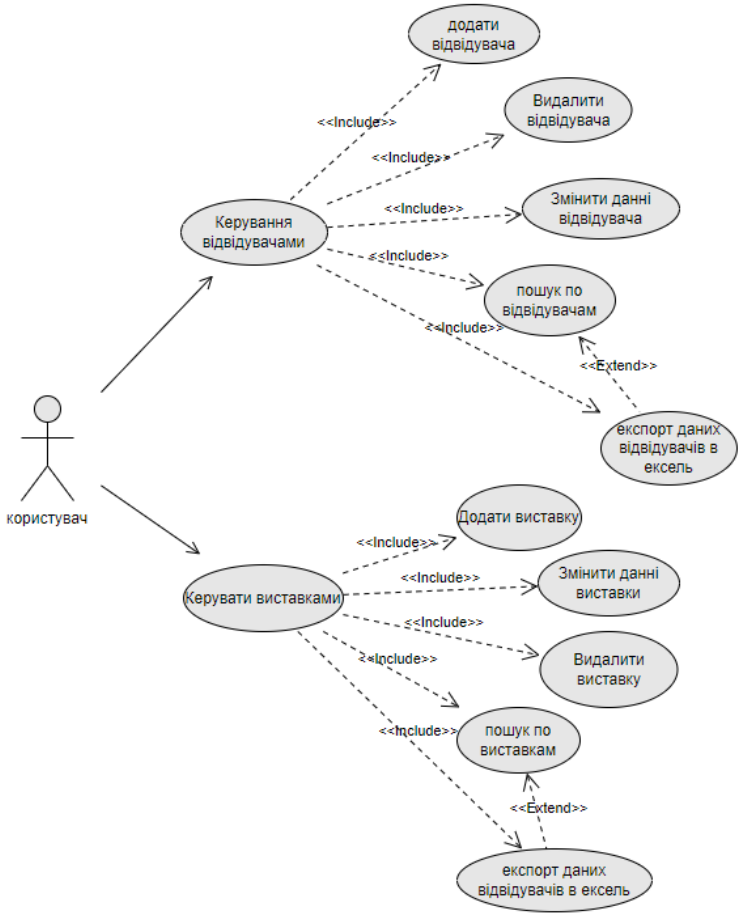


Рисунок 4.1 - UML Діаграма прецедентів системи

Опис прецедентів:

1) керування відвідувачами – що включає в себе:
   - додати відвідувачами: Адміністратор виставки додає нового відвідувача, вписує дані та зберігає зміни.
   - видалити відвідувача: Адміністратор виставки видаляє відвідувача та зберігає зміни.
   - змінити дані відвідувача: Адміністратор виставки змінює дані відвідувача, оновлює дані та зберігає зміни.
   - пошук по відвідувачам: Адміністратор виставки шукає відвідувачів що мають конкретні дані з додатковою можливістю експорту обраних даних в ексель.
   - експорт даних відвідувачів в ексель: Адміністратор виставки експортує дані про всіх відвідувачів в ексель.

2) керування виставками – що включає в себе:
   - додати виставку: Адміністратор виставки додає нову виставку, вписує дані та зберігає зміни.
   - видалити виставку: Адміністратор виставки видаляє виставку та всіх пов’язаних з нею відвідувачів і зберігає зміни.
   - змінити дані виставки: Адміністратор виставки змінює дані виставки, оновлює дані та зберігає зміни.
   - пошук по відвідувачам: Адміністратор виставки шукає виставки що мають конкретні дані з додатковою можливістю експорту обраних даних в ексель.
   - експорт даних відвідувачів в ексель: Адміністратор виставки експортує дані про всі виставки в ексель.

4.1.2 Діаграма станів

Діаграма станів, також відома як діаграма переходів станів, є графічним інструментом, який використовується для моделювання поведінки системи або об'єкту через його різні стани та переходи між ними. Це дозволяє розробникам та аналітикам більш чітко розуміти, як система працює і які є її можливі стани.

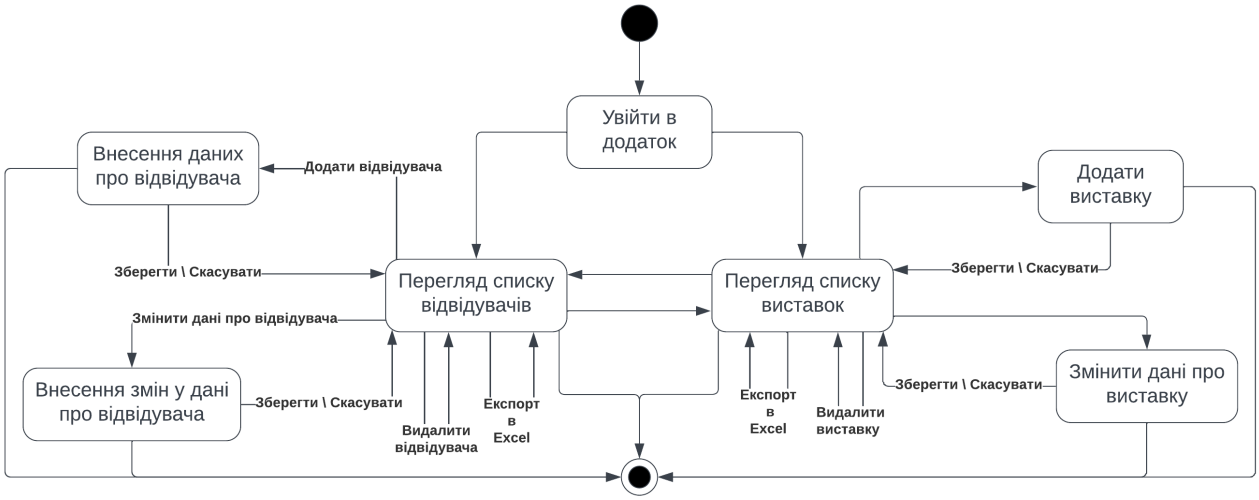


Рисунок 4.2 – UML Діаграма станів

4.1.3 Діаграми діяльності

Діаграма діяльності є одним із видів UML-діаграм. Вона використовується для моделювання послідовності дій або процесів у системі. Цей інструмент дозволяє графічно зобразити послідовність дій, розгалуження та паралельні процеси, що відбуваються в системі.

Одна з основних функцій діаграми діяльності полягає в тому, що вона допомагає зрозуміти, як система взаємодіє з акторами або іншими системами. Таким чином, діаграма діяльності сприяє кращому розумінню процесів, що відбуваються у системі.

Подальшою користністю діаграми діяльності є її здатність до візуалізації процесів в системі. Завдяки графічному зображенню дій та кроків, діаграма діяльності дозволяє зрозуміти послідовність виконання, а також ідентифікувати розгалуження та паралельні процеси. Це допомагає уточнити та усвідомити кроки, які потрібно виконати для досягнення конкретних цілей у системі.

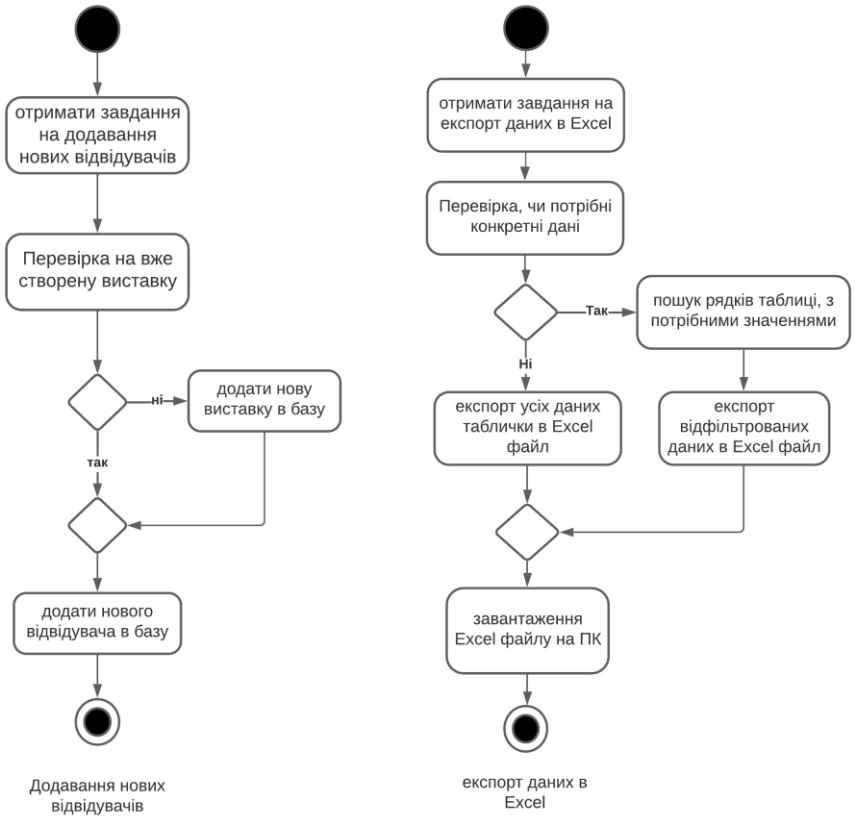


Рисунок 4.3 - UML діаграми діяльності

4.2 Організація даних

Так як для обліку у програмі використовується SQL Server, тому наші дані будуть збережені в реляційній базі даних. Структура такої бази даних складається з кількох важливих елементів, які забезпечують організацію та доступ до даних.

Першим елементом є таблиці, які виступають основними одиницями структури реляційної бази даних. Кожна таблиця складається зі збору рядків та стовпців.

Стовпці, які визначають поля кожної таблиці, задають типи даних, які можуть бути збережені в кожному полі. Та мають імена які використовуються для посилання на конкретні поля або атрибути даних в таблиці. Імена стовпців мають відображати призначення конкретного поля, наприклад, "ім'я", "прізвище", "дата_народження" тощо.

Кожний рядок таблиці представляє окремий запис або об'єкт, який містить дані про певну сутність або подію. Для ідентифікації кожного запису в таблицю використовуються ключі.

Ключі це спеціальні стовпці, які мають унікальні значення для кожного запису в таблиці. Окрім забезпечення цілісності даних, ключи також використовуються для встановлення зв’язків між таблицями

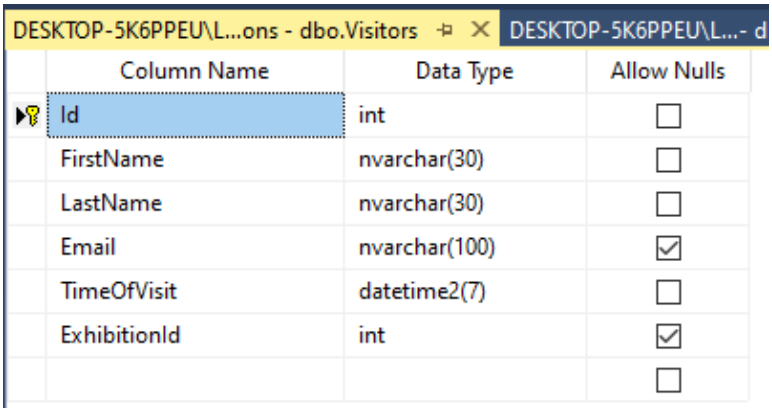


Рисунок 4.4 – Вигляд спроектованої таблиці бази даних, на прикладі таблиці відвідувачів

На рисунку 4.4 показані налаштування таблиці на прикладі таблиці відвідувачів. Ця таблиця містить назви стовпців, з яких вона складається та типів даних які вони можуть зберігати. Крім того, на цій таблиці можна побачити, які рядки не можуть мати значення "null".

Важливою частиною структури реляційної бази даних є зв’язування таблиць. Це означає, що поля однієї таблиці можуть посилатися на поля іншої таблиці, утворюючи таким чином зв'язки між ними. Ці зв'язки визначаються за допомогою зовнішніх ключів, які вказують на відповідність між значеннями у полях двох таблиць.

Зв'язування таблиць спрощує роботу з базою даних, дозволяючи ефективно організувати, зберігати та отримувати дані. Воно сприяє цілісності даних і допомагає забезпечити правильність зв'язків між різними сутностями у системі.

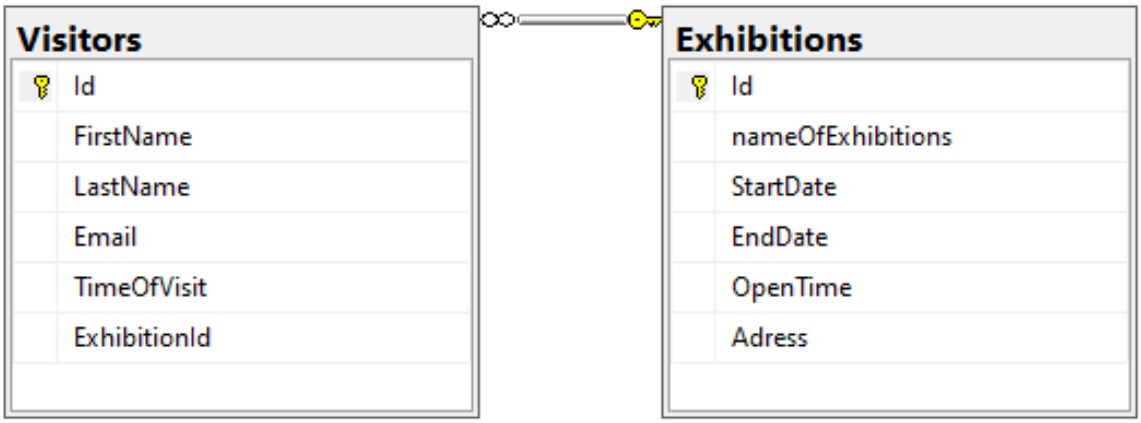


Рисунок 4.5 – Зв’язок між двома таблицями

На рисунку 4.5 представлено зв'язок між двома таблицями, який ілюструє вигляд зв'язку між ними. Зокрема, можна спостерігати, що таблиця "Visitors" зв’язана з таблицею "Exhibitions". У нашому випадку, ми використовуємо рядок "ExhibitionId" як посилання на первинний ключ таблиці "Exhibitions". В такому контексті "ExhibitionId" називається зовнішнім ключем або Foreign key.

Зовнішній ключ це поле або набір полів в одній таблиці, яке посилається на первинний ключ іншої таблиці. Зв'язок між таблицями встановлюється шляхом

використання значення зовнішнього ключа в одній таблиці, яке збігається зі значенням первинного ключа в іншій таблиці.

Зв'язування таблиць дозволяє створювати багатоваріантні запити до бази даних, отримувати пов'язану інформацію з декількох джерел і забезпечувати цілісність даних. Це розширює можливості бази даних і дозволяє ефективно зберігати та отримувати зв'язану інформацію.

4.3 Використання EF Core та SQL Server для операцій з даними

Database-First підхід є одним з поширених методів при розробці бази даних з використанням Entity Framework Core. При його застосуванні спочатку визначається схема бази даних, а потім генерується модель даних на основі цієї схеми.

Перевагою підходу Database-First є те, що розробнику необхідно лише визначити структуру бази даних, і EF автоматично створює класи моделі та контекст бази даних на основі цієї структури. Це дозволяє ефективно використовувати існуючу базу даних без необхідності переписування коду з нуля. На початку розробки саме такий підхід і застосовувався.

Для легкого створення моделі даних та контексту бази даних на основі існуючої бази даних, EF Core надає механізм, відомий як Reverse Engineering. Цей механізм дозволяє швидко створити модель даних без необхідності вручну визначати кожен клас та властивість

За допомогою Reverse Engineering, EF Core може проаналізувати структуру існуючої бази даних і автоматично згенерувати модель даних, яка відповідає цій структурі. Це означає, що класи, що представляють таблиці бази даних, та властивості цих класів, які відповідають стовпцям таблиць, будуть автоматично створені за вас.

Завдяки Reverse Engineering, процес створення моделі даних стає значно простішим та швидшим. Ви можете заощадити час і зусилля, оскільки не потрібно в ручну визначати кожен клас та властивість в моделі. Замість цього, EF Core

самостійно виявляє структуру бази даних та створює модель, що відповідає цій структурі.

Якщо вже є створена та визначена база даних, тоді Reverse Engineering виконується у декілька кроків:

1) створити проект
2) додати через NuGet Package Manager 2 пакети:
   - Microsoft.EntityFrameworkCore.SqlServer - забезпечує функціональність Entity Framework для роботи з MS SQL Server
   - Microsoft.EntityFrameworkCore.Tools - необхідний для створення класів по базі даних, тобто reverse engineering
3) відкрити Package Manager Console
4) виконати команду Scaffold-DbContext, передавши параметри підключення до сервера та назви бази даних.

Після виконання цих дій буде додано у проект класи сутності, а також клас контексту даних.

Однак, підхід Database-First має деякі недоліки. Зміни в базі даних можуть бути складними, оскільки вони можуть вимагати ручного оновлення моделі даних. Також, в даному підході модель даних залежить від бази даних, що може обмежити переносимість та гнучкість системи.

Для полегшення використання додатку та розв'язання недоліків підходу Database-First, було вирішено реалізувати механізми Code-First. Підхід Code-First дозволяє визначати модель даних безпосередньо в коді додатку, а EF автоматично створює базу даних на основі цієї моделі. Це забезпечує більшу гнучкість і незалежність моделі від бази даних, а також спрощує роботу зі змінами структури даних.

Для реалізації Code-First підходу, було використано Fluent API, який надає можливість визначати додаткові правила та налаштування для моделі даних. Крім того, метод EnsureCreated в EF Core дозволяє автоматично генерувати базу даних з використанням визначеної моделі, якщо база даних ще не створена. Це спрощує

використання додатку та дозволяє автоматично створювати базу даних для різних користувачів програми без необхідності ручного налаштування.

```
modelBuilder.Entity<Visitor>()
    .ToTable("Visitors");

modelBuilder.Entity<Visitor>()
    .HasKey(x => x.Id);

modelBuilder.Entity<Visitor>()
    .Property(x => x.FirstName)
    .HasMaxLength(30)
    .IsRequired();

modelBuilder.Entity<Visitor>()
    .Property(x => x.LastName)
    .HasMaxLength(30)
    .IsRequired();


modelBuilder.Entity<Visitor>()
    .Property(x => x.Email)
    .HasMaxLength(100);

modelBuilder.Entity<Visitor>()
    .Property(x => x.Email)
    .HasMaxLength(100);
```

Рисунок 4.6 – Приклад використання ланцюжкових методів Fluent API, для ручного налаштування відповідності моделі даних.

Fluent API дозволяє вам легко налаштувати різні правила для ваших властивостей. Ви можете використовувати методи ланцюжків для створення послідовно написаних правил валідації, наприклад максимальну довжину або ім’я таблиці, налаштування яких продемонстровано на рисунку 4.6.

На відміну від Data Annotation, Fluent API дозволяє вам тримати код вашої моделі більш чистим та зрозумілим. Замість розкидування налаштування по атрибутам, все згруповано в одному місці.

```
modelBuilder.Entity<Visitor>(entity =>
{
    entity.HasIndex(e => e.ExhibitionId, "IX_Visitors_ExhibitionId");

    entity.HasOne(d => d.Exhibition)
        .WithMany(p => p.Visitors)
        .HasForeignKey(d => d.ExhibitionId)
        .OnDelete(DeleteBehavior.Cascade);
});
```

Рисунок 4.7 – Приклад налаштування відношення між таблицями OneToMany

На рисунку 4.7 продемонстровано налаштування відношення OneToMany, вказання зовнішніх ключей та налаштування каскадного видалення всіх зв’язаних записів з конкретною виставкою.

Реалізація механізмів Code First дало нам дуже зручна функціональність. При першому запуску програми EF Core перевірить, чи існує база даних, згідно з конфігурацією підключення. Якщо база даних не існує, EF Core автоматично створить нову базу даних, з урахуванням визначеної моделі, на локальному сервері користувача.

5 ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

5.1 Концептуальний підсумок

Після аналізу аналогічних програм, їх функціоналу та умов війни, в яких ми зараз знаходимось - було прийняте рішення про розробку програмного продукту у форматі "On-premises software". Що означає, що програма буде встановлюватись і запускатись безпосередньо на локальних серверах або комп'ютерах користувачів. Вона не потребує прямого підключення до Інтернету і надає можливість користувачам повністю контролювати та зберігати свої дані власними силами.

Це має свої певні переваги, такі як:

- незалежність від Інтернету: При відсутності Інтернет-з'єднання, програма все одно продовжує працювати, оскільки дані зберігаються локально. Це дає користувачам можливість здійснювати облік відвідувачів навіть у випадку обмеженого або жодного доступу до Інтернету
- збереження продуктивності: При відключенні електроенергії чи проблемах з Інтернет-з'єднанням, On-premises software дозволяє продовжувати роботу без значних перебоїв, використовуючи заряд ноутбука. Ви можете продовжувати виконувати облік відвідувачів, вносити зміни та здійснювати необхідні операції, забезпечуючи неперервну продуктивність.
- збереження даних під контролем: У випадку втрати електроенергії, On-premises software дозволяє зберігати дані локально, у вашому власному середовищі. Це дозволяє уникнути потенційних ризиків втрати даних або несанкціонованого доступу до них.

Цільова аудиторія:

- організатори виставок та заходів: У ситуації воєнного стану, коли необхідно забезпечити безпеку та контроль за відвідувачами, програма для обліку відвідувачів може бути використана організаторами виставок, заходів та демонстрацій. Вона дозволить відслідковувати кількість відвідувачів, контролювати доступ для забезпечувати безпеку приміщення.

- благодійні організації: Благодійні організації, які проводять акції зі збору коштів на підтримку армії, постраждалих або інших соціальних потреб, можуть використовувати програму для обліку відвідувачів на своїх заходах. Вони можуть аналізувати дані про відвідувачів, забезпечувати безпеку та створювати звітність для оцінки успішності акцій та залучення донорів.

5.2 Розробка інтерфейсу

Зважаючи на вибір розробки десктопного застосунку, вибір фреймворку для графічного інтерфейсу є важливим етапом. Є два найпопулярніші фреймворки для створення графічного інтерфейсу користувача з використанням мови програмування C#: Windows Forms та Windows Presentation Foundation (WPF).

Windows Presentation Foundation (WPF) - це інструментарій користувача для створення десктопних програм, який був представлений Microsoft у 2006 році. WPF був створений з метою заміни WinForms, який був основним інструментарієм користувача для десктопних програм Windows з 2002 року. Хоча WinForms і залишається у використанні й досі, WPF надає більш сучасний, візуально привабливий інструментарій користувача, побудований на основі DirectX.

Їх спільні риси:

- використання мови розмітки XAML (eXtensible Application Markup Language) для опису інтерфейсу: Windows Forms і WPF обидва підтримують використання XAML, яка є мовою розмітки, що дозволяє описувати структуру і вигляд інтерфейсу додатка. XAML дозволяє відокремити дизайн інтерфейсу від логіки програми, спрощуючи роботу з додатком.
- використання обробки подій для реагування на взаємодію користувача з інтерфейсом: Windows Forms і WPF надають можливості для обробки подій, які виникають під час взаємодії користувача з інтерфейсом. Це дозволяє реагувати на дії користувача, такі як натискання кнопок, введення тексту, вибір пунктів меню тощо. Розробники можуть призначати обробники подій, які виконують певні дії при настанні певних подій.

- мова програмування C#: Обидва фреймворки - Windows Forms і WPF - можуть бути використані для розробки десктопних додатків з використанням мови програмування C#.
- інтеграція з Visual Studio: для розробки додатків на Windows Forms або WPF, ви можете використовувати Visual Studio - інтегроване середовище розробки (IDE) від Microsoft Він надає зручний інтерфейс користувача, автодоповнення коду, візуальний дизайнер і багато інших функцій для покращення продуктивності розробки.

Відмінності

Одна з ключових відмінностей між WPF і WinForms - це підхід до роботи з графікою. WPF використовує систему графіки на основі векторів, що дозволяє створювати високоякісні, масштабовані зображення, які легко анімувати та змінювати.

WinForms, натомість, використовує систему графіки на основі растру, що може призводити до зображень нижчої якості та обмежених можливостей анімації.

Інша відмінність між цими фреймворками полягає в підтримці сучасних принципів дизайну інтерфейсу. WPF має підтримку функцій, таких як зв'язування даних, стилі та шаблони, що спрощує створення стильних, сучасних інтерфейсів.

WinForms, в свою чергу, обмежений в підтримці цих функцій, що може ускладнювати створення сучасних інтерфейсів без використання власного коду.

WinForms, вважається більш легким та продуктивним інструментарієм користувача. Він може забезпечувати реагуючі інтерфейси навіть на системах з обмеженими можливостями, але не надає такого рівня візуальної привабливості, як WPF.

І враховуючи такі переваги WPF як:

- сучасні та візуально привабливі інтерфейси
- підтримка високоякісної графіки та анімації
- більша майбутньоорієнтованість програми

Було надано перевагу Windows Presentation Foundation.

Основна ідея роботи з WPF полягає в описі інтерфейсу користувача за допомогою мови розмітки XAML (eXtensible Application Markup Language). XAML дозволяє відокремити дизайн інтерфейсу від логіки програми, що дозволяє розробникам та дизайнерам працювати над проектом паралельно і спільно.

Windows Presentation Foundation (WPF) є технологією, що базується на .NET Framework і використовується для розробки сучасних інтерактивних десктопних додатків у Windows. У роботі з WPF ключову роль відіграють вікна і код-бек файли. Суть роботи в WPF полягає в створенні вікон та визначенні їх зовнішнього вигляду та структури за допомогою XAML (Extensible Application Markup Language). XAML файл визначає розміщення елементів інтерфейсу, їх властивості, стилі, шаблони та анімацію. Вікна можуть містити кнопки, тексти, списки, таблиці та інші елементи управління, які відображаються на екрані.

Однак, окрім зовнішнього вигляду, важливо також визначити поведінку вікна та його логіку. Для цього використовуються code-behind файли, які автоматично генеруються Visual Studio при створенні нового вікна WPF. Ці файли містять C# код, в якому описується логіка програми, включаючи обробники подій, властивості, методи та інші функції, пов'язані з вікном. У код-бек файлі можна оголошувати змінні для елементів управління, визначати обробники подій, звертатися до властивостей та методів елементів інтерфейсу, а також виконувати різні дії при взаємодії з користувачем. Наприклад, при натисканні на кнопку можна виконати певну дію, валідувати введені дані або змінювати властивості інших елементів.

5.3 Структура програми та функціонал.

5.3.1 Діаграма класів

Діаграма класів є важливим інструментом для опису архітектури програмної системи. Вона дозволяє візуалізувати структуру системи, компоненти та взаємозв'язки між ними з точки зору класів і їх властивостей

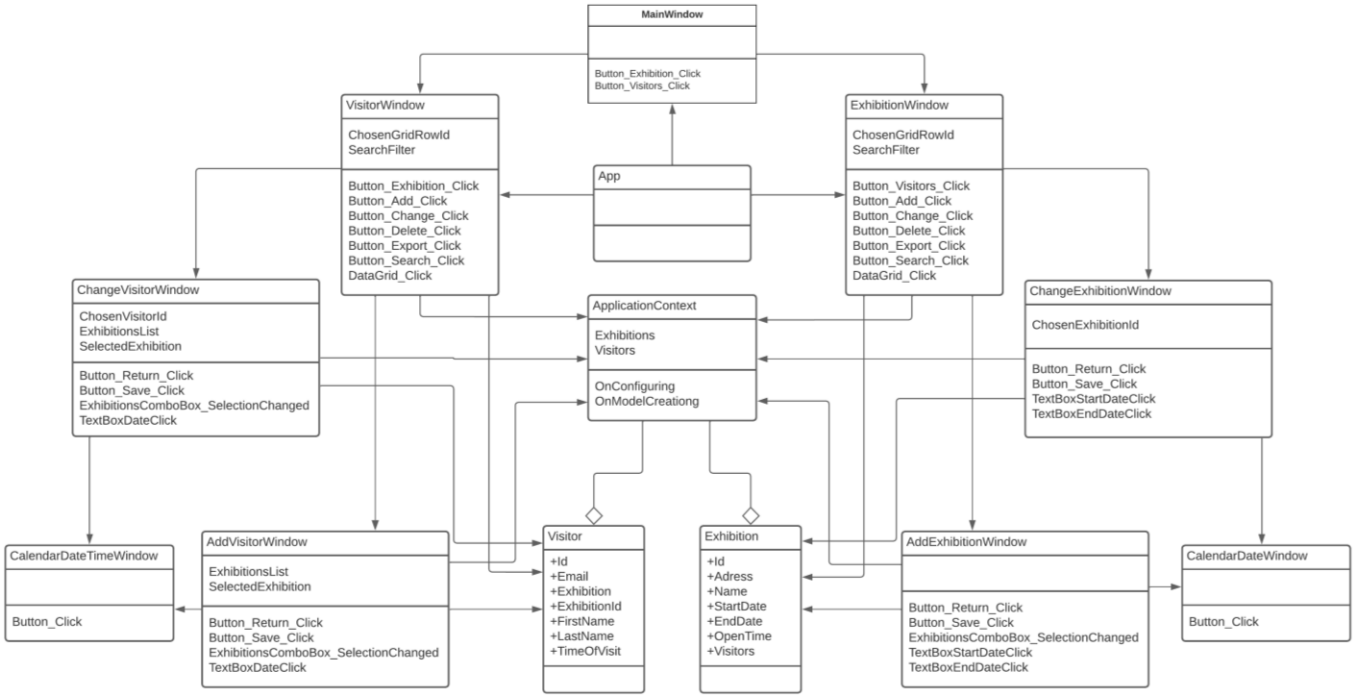


Рисунок 5.1 – UML Діаграма класів

5.3.2 Діаграма пакетів

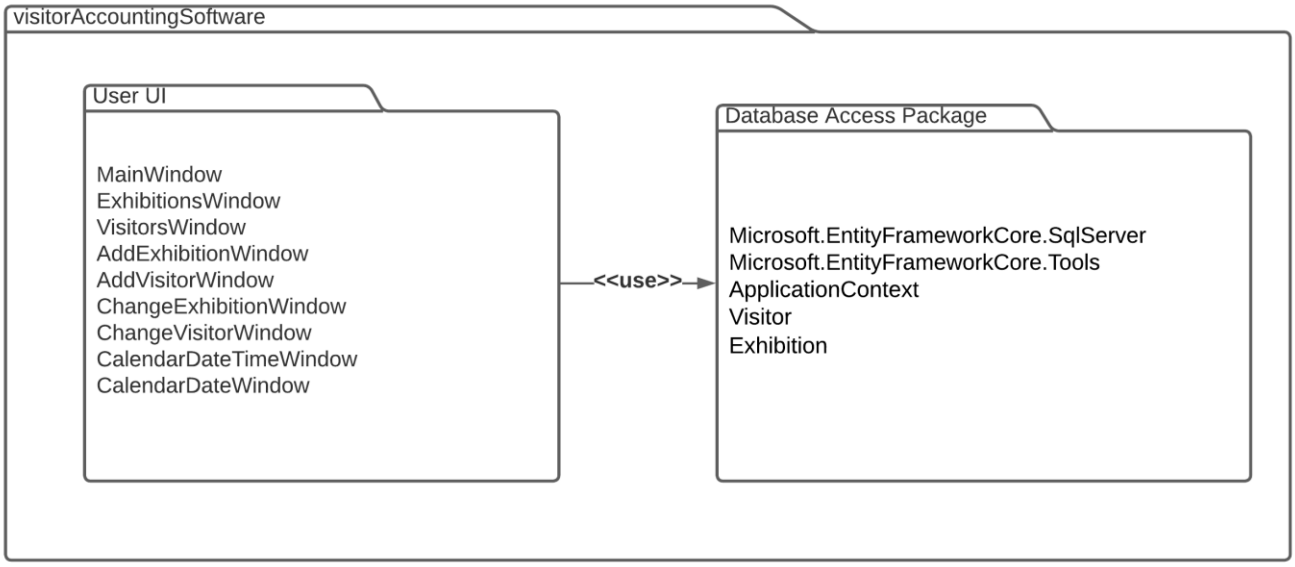


Рисунок 5.1 – UML діаграма пакетів програми

Опис діаграми пакетів:

Пакет: user UI

Опис: Містить класи, пов'язані з користувацьким інтерфейсом програми. Цей пакет відповідає за візуальну складову програми і забезпечує інтерфейс для взаємодії з користувачем.

Включені класи:

- MainWindow: Клас, що представляє основну точку входу та навігацію у програмі.
- ExhibitionsWindow: Клас, що представляє вікно для перегляду та керування виставками. Надає можливість переглядати, додавати, редагувати та видаляти виставки.
- VisitorsWindow: Клас, що представляє вікно для перегляду та керування відвідувачами. Забезпечує функціонал для перегляду, додавання, редагування та видалення відвідувачів.
- AddExhibitionWindow: Клас, що представляє вікно для додавання нової виставки. Дозволяє користувачеві ввести необхідну інформацію про виставку та зберегти ці дані у базі даних.
- AddVisitorWindow: Клас, що представляє вікно для додавання нового відвідувача. Він надає користувачеві можливість ввести необхідну інформацію про відвідувача та зберегти ці дані у базі даних.
- ChangeExhibitionWindow: Клас, що представляє вікно для редагування інформації про виставку. Він дозволяє користувачеві змінити різні атрибути виставки, такі як назва, дата, місце проведення та інші, та зберегти зміни у базі даних.
- ChangeVisitorWindow: Клас, що представляє вікно для редагування інформації про відвідувача. Він дозволяє користувачеві змінити різні атрибути відвідувача, такі як ім'я, прізвище, контактні дані тощо, та зберегти зміни у базі даних.
- CalendarDateTimeWindow: Клас, що представляє вікно календаря для вибору дати та часу. Що використовується при вводі даних про відвідувача.

- CalendarDateWindow: Клас, що представляє вікно календаря для вибору дати. Він надає користувачеві інтерфейс для вибору конкретної дати, яка використовується для вводу даних про виставки.

Пакет: Database Access Package

Опис: Містить класи та залежності для доступу до бази даних. Цей пакет відповідає за доступ до бази даних і взаємодію з нею з використанням підходу Code First.

Включені класи:

- ApplicationContext: Клас, що наслідується від DbContext, представляє контекст доступу до бази даних. Використовується для взаємодії з базою даних, включаючи створення, зчитування, оновлення та видалення даних. Він містить набір DbSet властивостей для доступу до таблиць бази даних.
- Visitor: клас-сутність, який відповідає табличці Visitors у базі даних.
- Exhibition: клас-сутність, який відповідає табличці Exhibitions у базі даних.

Включені залежності:

- Microsoft.EntityFrameworkCore.SqlServer: Пакет, який надає підтримку для використання SQL Server з Entity Framework Core. Вона містить необхідні класи та компоненти, які дозволяють підключатися до SQL Server та взаємодіяти з ним. Ця залежність додає можливості для роботи з базою даних SQL Server в контексті Entity Framework Core.
- Microsoft.EntityFrameworkCore.Tools: Пакет, який надає інструменти для розробки з Entity Framework Core, включаючи міграції та генерацію коду. Ця залежність дозволяє автоматично створювати базу даних, таблиці і відповідні схеми на основі визначеної моделі даних. Вона надає зручні інструменти для роботи з базою даних в контексті Entity Framework Core.

Взаємодія:

User UI використовує Database Access Package для доступу до бази даних і взаємодії з нею. Класи з пакету User UI можуть використовувати класи та

компоненти з пакету Database Access Package для створення, зчитування, оновлення, видалення даних та іншої взаємодії з базою даних.

Описаний на діаграмі пакетів інтерфейс з наявним списком вікон має повністю завершений дизайн вікон, що включає елементи керування, розміщення вмісту і оформлення. Крім того, налаштовані переходи між вікнами дозволяють користувачам зручно переходити між різними частинами програмного додатку.

Вся необхідна функціональність реалізована для ефективної роботи з даними. Це включає відображення таблиць бази даних з виставками та відвідувачами,

```
InitializeComponent();
using (ApplicationContext dbContext = new ApplicationContext())
{
    var result = dbContext.Visitors.Select(x => new
    {
        Id = x.Id,
        FirstName = x.FirstName,
        LastName = x.LastName,
        Email = x.Email,
        TimeOfVisit = x.TimeOfVisit.ToString("dd.MM.yyyy HH:mm"),
        ExhibitionName = x.Exhibition.Name
    }).ToList();

    VisitorsList.ItemsSource = result;
```

Рисунок 5.2 – Код для отримання списку даних з бази даних відвідувачів, та передача його до табличці інтерфейсу з ім’ям “VisitorsList”

А також створене додавання та зміна даних використовуючи спеціальну форму. Крім того, надається можливість легкого видалення елементів – все що для цього треба, це клікнути по рядку таблички, який ви хочете видалити, та натиснути кнопку видалити.

Існує можливість виконання пошуку за певними параметрами, що дозволяє швидко знаходити необхідну інформацію.

Для зручності користувачів надається можливість експорту даних з таблиць, як відфільтрованих за пошуком, так і усієї таблиці, в формат Excel, що дозволяє подальшу обробку або аналіз інформації у зручній формі.

5.4 Тестування додатку

Для тестування додатку з обліку відвідувачів та виставок було обрано основні параметри, які треба протестувати:

1) функціональність: Перевірка, чи виконує додаток всі функції, які передбачені в специфікації.

Таблиця 5.1 – Тестування додатку

Вікно яке тестувалось	**Протестований функціонал**	**Очікуваний результат**	**Результат тесту**
Вікно відображення виставок	Відображення даних	У вікні відображаються таблиця створених виставок	Успіх
Вікно відображення виставок	Видалення виставки з бази даних	Після введення даних про нову виставку у відповідні поля і збереження – створена виставка була додана у таблицю	Успіх
Вікно відображення виставок	Фільтр даних про виставки за пошуком	Після вводу у поле пошуку значення і натискання пошуку, в таблицю вивелись тільки виставки, що мають у своїх колонках відповідний текст	Успіх
Вікно відображення виставок	Експорт даних про виставки	Після натискання кнопки «Експортувати в Excel», був створений Excel файл із даними з таблиці виставок	Успіх
Вікно додавання виставок	Внесення даних про нову виставку	Після введення даних про нову виставку у відповідні поля і збереження – створена виставка була додана у таблицю	Успіх

Продовження таблиці 5.1

Вікно яке тестувалось	**Протестований функціонал**	**Очікуваний результат**	**Результат тесту**
Вікно редагування виставок	Зміна даних про додану у базу виставку	Після зміни даних про існуючу в базі виставку, використовуючи відповідні поля вводу, і збереження – дані про існуючу виставку були редаговані	Успіх

Ті самі перевірки функціональності були виконані з вікнами взаємодії з таблицею відвідувачів - всі перевірки успішні.

2) взаємодія з базою даних: Перевіка, чи зберігаються та відображаються дані правильно в базі даних. Удостовірення, що додаток взаємодіє з базою даних коректно, виконуючи операції додавання, оновлення та видалення даних.

Для цього тестування було використане SQL Server Management Studio інтегроване середовище розробки та управління базами даних SQL Server:

а) для перевірки даних у самій базі даних, спочатку треба підключитися до локального сервера:

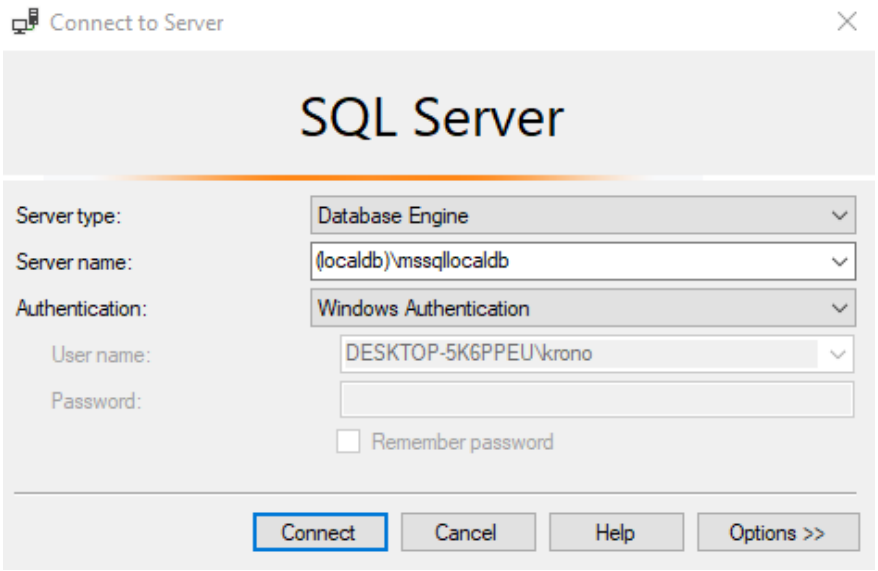
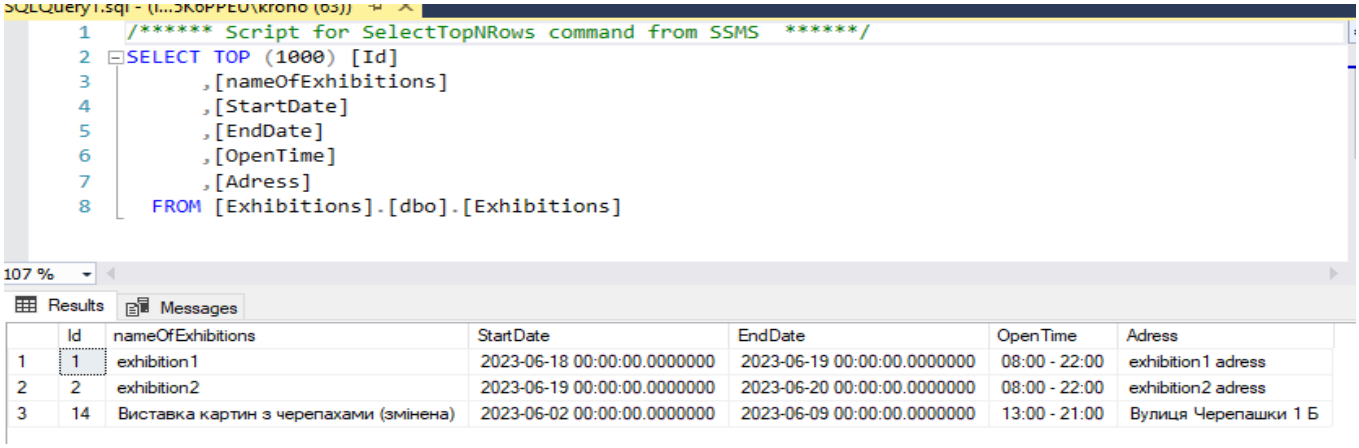


Рисунок 5.8 – Інтерфейс підключення до локального сервера

b) наступний крок це знайти в деревовидній структурі об’єктів бази даних ім’я використовуваної бази даних, в ній знайти таблиці бази даних, натиснути на них правою кнопкою миші та обрати команду “Select Top 1000 Rows”. Зважаючи на те, що кількість рядків у нашій таблиці менша за 1000, будуть виведені всі рядки



	Id	nameOfExhibitions	StartDate	EndDate	OpenTime	Adress
1	1	exhibition1	2023-06-18 00:00:00.0000000	2023-06-19 00:00:00.0000000	08:00 - 22:00	exhibition1 adress
2	2	exhibition2	2023-06-19 00:00:00.0000000	2023-06-20 00:00:00.0000000	08:00 - 22:00	exhibition2 adress
3	14	Виставка картин з черепахами (змінена)	2023-06-02 00:00:00.0000000	2023-06-09 00:00:00.0000000	13:00 - 21:00	Вулиця Черепашки 1 Б

Рисунок 5.9 – Виведення рядків бази даних відвідувачі в SSMS

Як ми бачимо, змінені в додатку дані, відображаються в базі даних, тому тест успішно пройдено.

Ті самі кроки були зроблені для перевірки таблиці відвідувачів – так само успішно.

3) перехід між вікнами: Перевірка, чи працюють переходи між різними вікнами додатку.

Перевірка всіх переходів між вікнами була успішно пройдена. Підтвердження цієї перевірки також подається у презентації

ВИСНОВКИ

1.Проведено аналіз предметної галузі. Було виявлено основні потреби користувачів, це включало оцінку вимог і викликів, з якими стикаються користувачі в цій галузі.

2.Досліджено та проаналізовано ринок аналогів програм для автоматизації обліку відвідувачів

3. Сформовано необхідний для користувачів функціонал та створено модель прецедентів із зображенням різного використання продукту, використовуючи за основу дані виявлені в ході аналізу предметної галузі та ринку програм аналогів для обліку відвідувачів виставки.

4. Застосовано моделювання діяльності програми за допомогою діаграми діяльностей, що дозволило візуалізувати послідовність роботи програми. Діаграма діяльностей відображала основні процеси та взаємодію різних компонентів програми.

5. Розроблена програма, що включає в себе базу даних, інтерфейс користувача та бізнес-логіку, що сприяло полегшенню обліку відвідувачів. Для реалізації цього були використані такі сучасні інструменти та технології, як C#, .NET, Entity Framework Core, WPF та Microsoft SQL Server Management Studio. Ці технології дозволили створити ефективну систему для збереження даних та забезпечення зручного інтерфейсу користувачу без необхідності використання серверів та залежності від Інтернету.

6. Проведено тестування розробленої програми з метою перевірки її функціональності та відповідності вимогам. Тестування включало в себе різні сценарії взаємодії користувача з програмою, а також перевірку правильності обліку відвідувачів та відображення даних в інтерфейсі користувача. Результати тестування підтвердили працездатність та точність розробленої програми

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Microsoft Learn [Електронний ресурс] : [Web-сайт] – Режим доступу: https://learn.microsoft.com/en-us/
2. UML - Basic Notations [Електронний ресурс] : [Web-сайт] – Режим доступу: https://www.tutorialspoint.com/uml/uml_basic_notations.htm
3. ITVDN [Електронний ресурс] : [Web-сайт] – Режим доступу: https://itvdn.com/ua
4. Metanit [Електронний ресурс] : [Web-сайт] – Режим доступу: https://metanit.com
5. Excel spreadsheet library for .NET Framework/Core [Електронний ресурс] : [Web-сайт] – Режим доступу: https://www.epplussoftware.com
6. McConnell Steve. Code Complete: A Practical Handbook of Software Construction, Second Edition. 2004
7. ECMA-334 5th edition C# Language Specification. 2017
8. ECMA-335 6th edition Common Language Infrastructure (CLI). 2012
9. Hejlsberg Anders. C# Programming Language 4th Edition. 2012
10. Knuth Donald. The art of computer programming Volume 2. 2001
11. Fowler Martin. UML Distilled: A Brief Guide to the Standard Object Modeling Language (2nd Edition). 1999

Додаток А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО- НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка програмного забезпеченнядля обліку відвідувачів виставки мовою C# з використаннямSQL

Виконавстудент 4 курсу
групи ПД-42
Михайленко ВадимДмитрович
Керівникроботи
ГаманюкІгор Михайлович, ст. викладачкафедриІПЗ

Київ – 2023

Мета, об’єкт та предмет роботи

- ***Мета роботи*** – спрощення процесу обліку відвідувачів виставки.
- ***Об'єкт дослідження*** – процес обліку відвідувачів виставки.
- ***Предмет дослідження*** – застосунок з обліку відвідувачів виставки.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Аналіз предметної галузі
2. Дослідження програмних забезпечень для обліку відвідувачі виставок
3. Формування вимог до програмного забезпечення
4. Розробка архітектури застосунку
5. Розробка користувацького інтерфейсу
6. Налаштування та запуск системи

АНАЛІЗ АНАЛОГІВ

	Excel	**Eventbrite**	**Cvent**	**Розроблена програма**
українська локалізація	+	-	-	+
Автоматизоване налаштування зберігання даних	-	+	+	+
Необмежена кількість подій	+	Лише у про версії	+	+
Автоматизована обробка даних	-	+	+	+
Автоматизована інтеграція даних	-	+	+	+
Модель реалізації	On-premises software	Software as a Service	Software as a Service	On-premises software

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги:

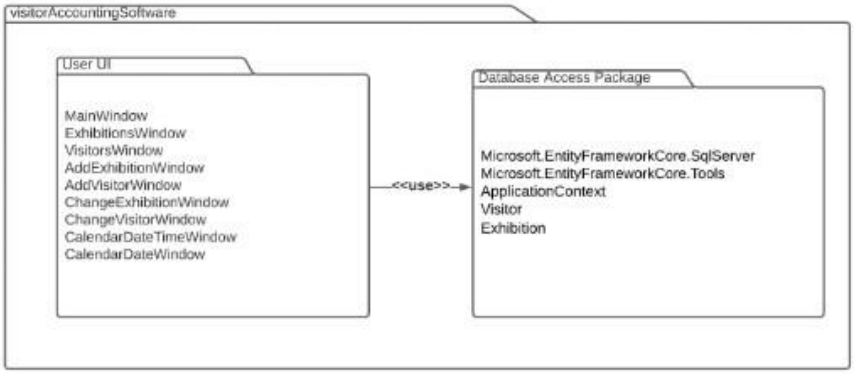
1. можливість додавати, редагувати та видаляти виставки та відвідувачів
2. автоматизоване налаштування зберігання даних
3. автоматизована обробка даних
4. автоматизована інтеграція даних
5. можливість пошуку відвідувачів та виставок
6. можливість експортування даних у форматі Excel

Не функціональні вимоги:

1. легкість використання та інтуїтивний інтерфейс
2. українська локалізація

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

АРХІТЕКТУРА ЗАСТОСУНКУ



ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ

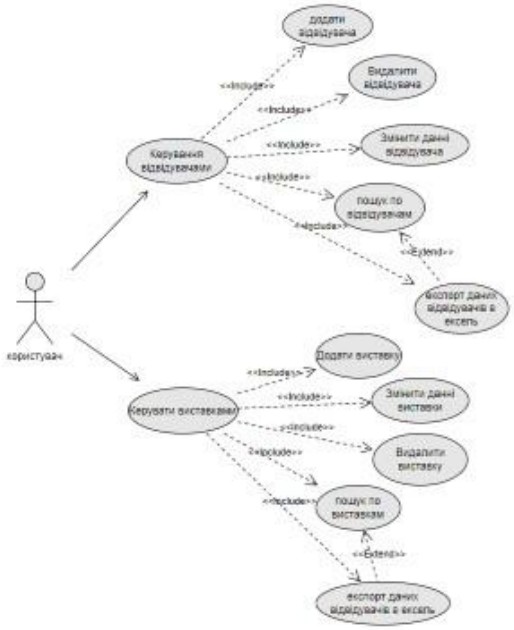
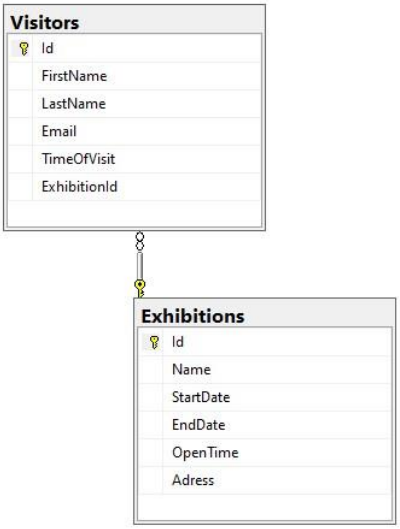
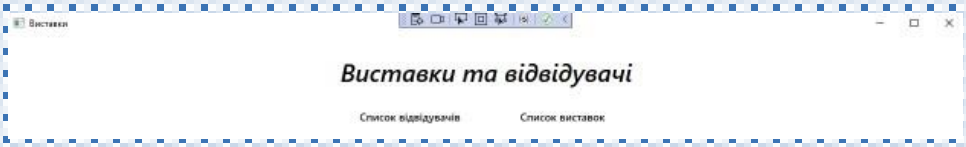


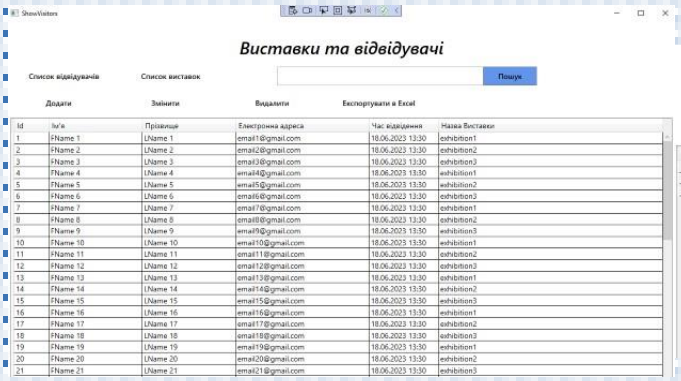
СХЕМА БАЗИ ДАНИХ



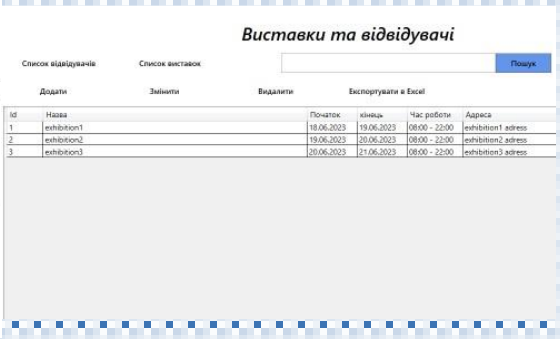
ЕКРАННІ ФОРМИ



Головна сторінка застосунку

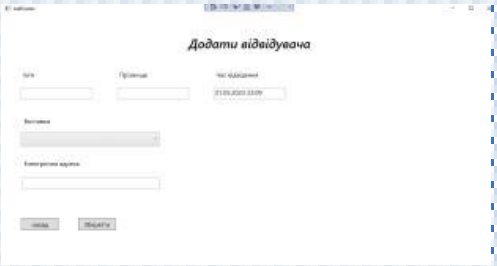


Перегляд відвідувачів



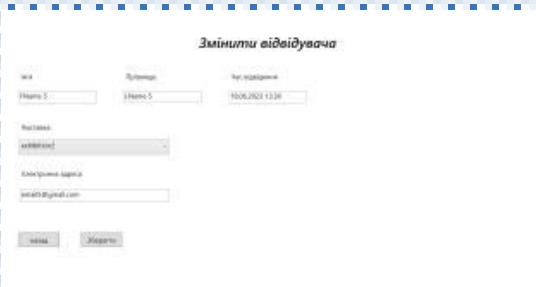
Перегляд виставок

ЕКРАННІ ФОРМИ

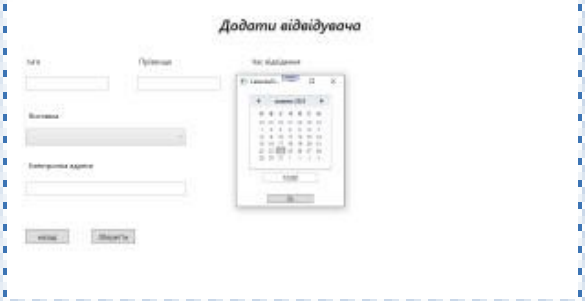


Додати відвідувача

Додати виставку

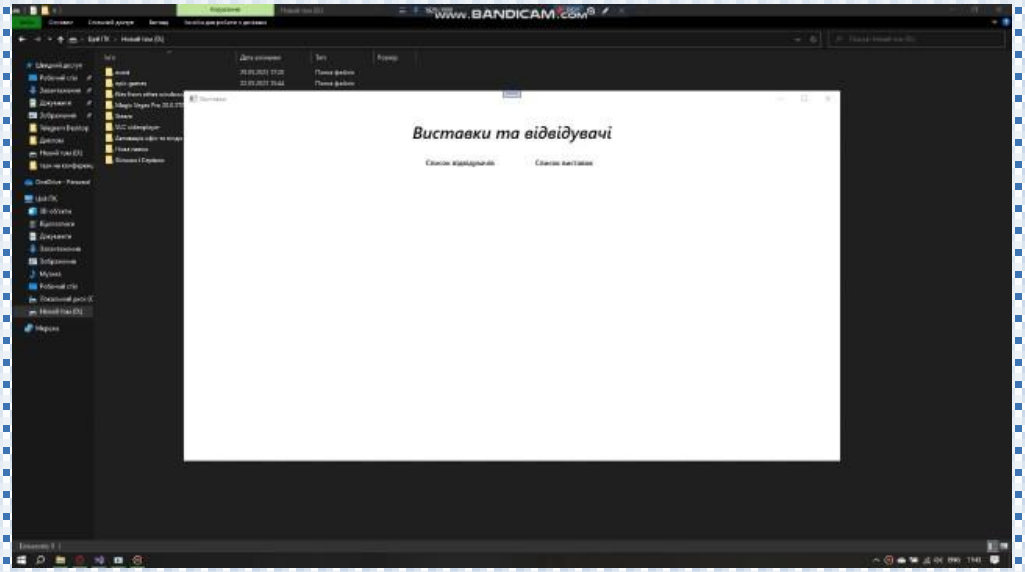


Змінити дані про відвідувача



Модальне вікно встановлення дати

Презентація взаємодії з програмою



АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Михайленко В. Д. Аналіз можливостей технології entity framework core для розробки програмного забезпечення для обліку відвідувачів виставки. Застосування програмного забезпечення в інфокомунікаційних технологіях : матеріали наук. -тех. конф., м. Київ, 20 квітня 2023 р. / Державний університет телекомунікацій, кафедра інженерії програмного забезпечення. Київ , 2023, с.12

2. Михайленко В. Д. Розробка програмного забезпечення для обліку відвідувачів виставки з використанням uml діаграми прецедентів . Застосування програмного забезпечення в інфокомунікаційних технологіях: матеріали наук. -тех. конф., м. Київ , 20 квітня 2023 р. / Державний університет телекомунікацій, кафедра інженерії програмного забезпечення. Київ, 2023, с.15

ВИСНОВКИ

1.Проведено аналіз предметної галузі. Було виявлено основні потреби користувачів, це включало оцінку вимог і викликів, з якими стикаються користувачів цій галузі.

2.Досліджено та проаналізовано ринок аналогів програм для автоматизації обліку відвідувачів

3. Сформованонеобхідний для користувачів функціонал та створено модель прецедентів із зображенням різного використання продукту,використовуючиза основу дані виявлені в ході аналізу предметної галузіта ринку програманалогів для обліку відвідувачіввиставки.

4. Розробленапрограма, що включає в себе розробкубази даних, інтерфейсу користувачата бізнес-логіку, що сприяло полегшенню обліку відвідувачів Для реалізації цього були використані такі сучасні інструменти та технології, як C#, .NET, Entity Framework Core, WPF та Microsoft SQL Server. Ці технології дозволили створити ефективну систему для збереження даних та забезпечення зручного інтерфейсу користувачубез необхідності використання серверівта залежності від Інтернету.