МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ФАХОВИЙ КОЛЕДЖ РАКЕТНО-КОСМІЧНОГО МАШИНОБУДУВАННЯ

ДНІПРОВСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ім. О. ГОНЧАРА

Циклова комісія програмної інженерії

**КУРСОВИЙ ПРОЕКТ**

з навчальної дисципліни

«ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ»

на тему: «Програмний додаток ведення обліку продажу путівок «Тури по Україні»

(вказати тему курсового проекту)

Студента IVкурсу  ПЗ-20-1  групи

галузь знань 12 «Інформаційні технології»

спеціальності 121 «Інженерія програмного забезпечення»

    Немога О.Г.

(прізвище та ініціали студента)

Керівник   
викладач Гапоненко Н.В.

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_ Оцінка ECTS: \_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_ Любохинець В.М.

(підпис) (прізвище та ініціали) .

\_\_\_\_\_\_\_\_\_ Ланська С.С. .

(підпис) (прізвище та ініціали) .

\_\_\_\_\_\_\_\_\_ Гапоненко Н.В. .

(підпис) (прізвище та ініціали) .

м. Дніпро - 2021 рік

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ФАХОВИЙ КОЛЕДЖ РАКЕТНО-КОСМІЧНОГО МАШИНОБУДУВАННЯ

ДНІПРОВСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ім. О. ГОНЧАРА

Цикловакомісія програмної інженерії

|  |  |
| --- | --- |
|  | **ЗАТВЕРДЖУЮ**  Голова комісії ПІ  \_\_\_\_\_\_\_\_\_\_\_\_ С.С.Ланська  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2021 р. |

**ЗАВДАННЯ**

**на виконання курсового проекту**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| з дисципліни | | | Об’єктно-орієнтоване програмування | | | | | | | |
| студенту | Немозi Олегу Геннадiйовичу | | | | | | | | | |
|  | (прізвище, ім'я та по батькові) | | | | | | | | | |
| Відділення | | Комп'ютерної та програмної інженерії | | | | | | | | |
| Спеціальність | | | | 121 Інженерія програмного забезпечення | | | | | | |
| Курс | IV | | | | | |  | Група (шифр) | | ПЗ-20-1 |
| 1 Тема проекту | | | | | «Програмний додаток ведення обліку продажу путівок | | | | | |
| «Тури по Україні» | | | | | | | | | | |
|  | | | | | | | | | | |
| 2 Початкові дані | | | | | | Туристичні об’єкти(назва, опис зображення), Тури(назва, | | | | |
| опис, зображення, дата проведення, тривалість, тип, кількість місць, | | | | | | | | | | |
| вартість), Путівки(номер, назва туру, дата туру, ПІБ клієнта, кількість місць, | | | | | | | | | | |
| сума до сплати, статус оплати, дата бронювання), Клієнти(ПІБ, номер | | | | | | | | | | |
| телефону, електронна пошта) | | | | | | | | | | |
|  | | | | | | | | | | |
| Розглянуто і ухвалено на засіданні циклової комісії | | | | | | | | | програмної інженерії | |
| Протокол № 2 від 21.09.2021 р. | | | | | | | | | | |

Керівник КП \_\_\_\_\_\_\_\_\_\_\_\_\_ Н.В.Гапоненко

(підпис) (ініціали та прізвище)

Завдання до виконання

одержав студент \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (ініціали та прізвище)

Дата видачі 21 вересня 2021 р.

Термін виконання 22 листопада 2021 р.

ЗМІСТ

[ВСТУП 4](#_Toc88790802)

[1 ПОСТАНОВКА ЗАДАЧІ 5](#_Toc88790803)

[2 ХАРАКТЕРИСТИКА ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ ПРОЕКТУ 8](#_Toc88790804)

[2.1 Опис середовища програмування 8](#_Toc88790805)

[2.2 Опис мови програмування 11](#_Toc88790806)

[2.3 Опис СКБД 13](#_Toc88790807)

[2.4 Опис основних принципів ООП 14](#_Toc88790808)

[2.5 Опис подібних програмних продуктів 16](#_Toc88790809)

[3 ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ 18](#_Toc88790810)

[3.1 Опис бази даних 18](#_Toc88790811)

[3.2 Проектування користувацького інтерфейсу програми 22](#_Toc88790812)

[3.3 Контроль вхідних даних програми 32](#_Toc88790813)

[4 ІНСТРУКЦІЯ З КОРИСТУВАННЯ ПРОГРАМНОЮ СИСТЕМОЮ 39](#_Toc88790814)

[ВИСНОВКИ 56](#_Toc88790815)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 57](#_Toc88790816)

[Додаток А 58](#_Toc88790817)

[Додаток Б 61](#_Toc88790818)

[Додаток В 65](#_Toc88790819)

[Додаток Г 141](#_Toc88790820)

ВСТУП

В сучасному світі інформаційних технологій та інтернет-комунікацій туристична галузь розквітає і набуває все більшого попиту. Україна, зі своєю розмаїтою природою, багатою історією та культурними скарбами, не залишається осторонь цього тренду. Із зростанням інтересу до туризму в Україні, з'являється необхідність у зручному та ефективному способі ведення обліку продажу туристичних путівок.

У цьому контексті, створення програмного додатку для ведення обліку продажу путівок «Тури по Україні» стає актуальним завданням. Такий додаток допоможе не тільки туристичним агентствам та туроператорам оптимізувати свою роботу, але й надасть можливість туристам знаходити та придбавати путівки в онлайн-режимі, забезпечуючи їм легкий доступ до різноманітних туристичних пропозицій по всій Україні.

У даному курсовому проєкті ми розглянемо основні аспекти розробки та впровадження такого програмного додатку, включаючи аналіз потреб користувачів, проектування функціоналу, розробку інтерфейсу, забезпечення безпеки даних та інші ключові аспекти. Такий додаток може допомогти зробити туризм більш доступним та привабливим для українців та іноземних туристів, сприяючи розвитку галузі та зростанню туристичного потенціалу України.

У наш час, коли цифрові рішення вже завоювали велику частину нашого повсякденного життя, створення програмного додатку для обліку продажу туристичних путівок є важливим кроком вперед у розвитку туристичної галузі України, і цей курсовий проєкт присвячений дослідженню та реалізації цієї ідеї.

1 ПОСТАНОВКА ЗАДАЧІ

Розробка «Програмний додаток ведення обліку продажу путівок «Тури по Україні». Програмна система передбачає:

* організація каталогу турів;
* організація бронювання та оплати путівок;
* організація ведення обліку клієнтів;
* організація ведення обліку туристичних об’єктів;
* ведення аналітики турів та туристичних об’єктів;
* формування путівок для друку;
* надання усіх необхідних функцій для зручного перегляду і пошуку інформації.

Детальний опис функціоналу:

1. Організація каталогу турів: надання можливості додавання, редагування, видалення та перегляд турів.
2. Організація бронювання та оплати путівок: надання можливості користувачу бронювання путівок в певній кількості та на певну дату, реалізувати можливість фіксації оплати білетів клієнтом, а також, в разі не сплати в зазначений термін, відмінити бронь.
3. Організація ведення обліку клієнтів: надання можливості додавання, редагування, видалення та перегляду інформації про клієнтів.
4. Організація ведення обліку туристичних об’єктів: надання можливості додавання, редагування, видалення та перегляду інформації по туристичних об’єктах.
5. Ведення аналітики турів та туристичних об’єктів: програма повинна надати можливість перегляду затребуваності турів, статистику бронювання білетів для кожного туру, а також статистику по туристичним об’єктам, що входять до певних турів.
6. Формування путівок для друку: програма повинна мати можливість друку білетів, для подальшої їх видачі клієнтам.
7. Надання усіх необхідних функцій для зручного перегляду і пошуку інформації: програма повинна надати засоби пошуку та фільтрації інформації всюди, де надано доступ для перегляду інформації.

Початкові дані містять:

1. Туристичні об’єкти:
   1. назва;
   2. опис;
   3. зображення.
2. Тури:
   1. назва;
   2. опис;
   3. зображення;
   4. дата проведення;
   5. тривалість;
   6. тип;
   7. кількість місць;
   8. вартість.
3. Путівки:
   1. номер;
   2. назва туру;
   3. дата туру;
   4. ПІБ клієнта;
   5. кількість місць;
   6. сума до сплати;
   7. статус оплати;
   8. дата бронювання.
4. Клієнти:
   1. ПІБ;
   2. номер телефону;
   3. електронна пошта.

Мінімальні вимоги до конфігурації ПК для користування цією програмою:

* центральний процесор Pentium IV або відповідний AMD;
* оперативна пам’ять 512 МБ;
* об’єм пам’яті жорсткого диску 256мб для клієнта і 5гб для сервера;

операційні системи Windows 7, 8, 10 або Linux.

2 ХАРАКТЕРИСТИКА ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ ПРОЕКТУ

2.1 Опис середовища програмування

Програма курсового проекту написана в середовищі Qt Creator Community Edition.

Qt Creator — кросплатформенний інструментарій розробки програмного забезпечення (ПЗ) мовою програмування C++. Дозволяє запускати написане за його допомогою ПЗ на більшості сучасних операційних систем (ОС), просто компілюючи текст програми для кожної операційної системи без зміни серцевого коду. Містить всі основні класи, які можуть бути потрібні для розробки прикладного програмного забезпечення, починаючи з елементів графічного інтерфейсу й закінчуючи класами для роботи з мережею, базами даних, OpenGL, SVG і XML. Бібліотека дозволяє керувати потоками, працювати з мережею та забезпечує кросплатформенний доступ до файлів.

Qt Creator має вбудований редактор форм, що дає змогу власноруч прописувати стиль форми можливостями мови CSS. Підтримує CMake, що дає змогу писати кросплатформенні додатки. З відомих додатків Qt використовує Google Earth, завдяки зручності використання цієї IDE .

Була обрана за можливість розробляти кросплатформенні проекти, легке перенесення проекту з однієї ОС у іншу, можливість написання мобільних додатків мовою С++, широкий вибір інструментів для розробки програмного забезпечення та зручний інтерфейс.

2.2 Опис мови програмування

Програма курсового проекту написана мовою програмування С++. Текст програми знаходиться у додатку Д.

C++, яка спочатку проектувалася, як розширення для С, а також на Java та C#, які запозичили у С синтаксис.

Мова програмування C++ найбільш поширена серед розробників програмного забезпечення. Вона є дуже зручною у розробці прикладних програм; драйверів пристроїв; розробка ОС; відео ігор. Реалізацією мови С++ займаються одночасно декілька проектів як безкоштовних, так і комерційних, а саме: GNU, Microsoft і Embarcadero (Borland).

C++ －високорівнева мова програмування загального призначення, поєднує властивості як високорівневих, так і низькорівневих мов програмування. Здатна до підтримки об'єктно-орієнтованого, узагальненого та процедурного програмування. Від свого попередника, мову програмування С, С++ відрізняється тим, що найбільшу увагу при розробці цієї мови було приділено підтримці об'єктно-орієнтованого та узагальненого програмування. Назва «мова програмування C++» походить від мови програмування C, в якому унарний оператор ++ позначає інкремент змінної.

На початку 1980-х років мову програмування С++ розробив співробітник фірми Bell Laboratories － Бьерн Страуструп. Створення мови С++ не планувалось, для власних потреб Бьерн розробив ряд удосконалень мови програмування С. Ранні версії мови С++, називались «Cі з класами» та почали з'являтися з 1980 року. Страуструп додав до мови програмування С можливість роботи з класами та об'єктами, поклавши фундамент нової, заснованої на синтаксисі С, мови програмування. Синтаксис C++ був заснований на синтаксисі C, так як Бьерн Страуструп прагнув зберегти сумісність з мовою C.

Нововведеннями С++ порівняно з С є:

– простий і практичний інтерфейс;

– підтримка об'єктно-орієнтованого програмування через класи;

– підтримка узагальненого програмування через шаблони;

– доповнення до стандартної бібліотеки;

– додаткові типи даних;

– обробка винятків;

– простори імен;

– вбудовані функції;

– перевантаження імен функцій;

– посилання і оператори управління вільно розподіленою пам'яттю.

Переваги С++:

1 Швидкість роботи програм на С++ майже не змінилась порівняно із С, хоча розробники отримали нові можливості і засоби.

2 Мова С++ є кросплатформенною.

3 Можливість роботи на низькому рівні з пам'яттю, адресами, портами.

4 Можливість створення узагальнених алгоритмів для різних типів даних, їхня спеціалізація і обчислення на етапі компіляції з використанням шаблонів.

5 Підтримуються різні стилі та технології програмування, включаючи традиційне директивне програмування, ООП, узагальнене програмування, мета програмування (шаблони, макроси).

Недоліки С++:

* погана підтримка модульності. Підключення інтерфейсу зовнішнього модуля через процесорну вставку заголовного файлу (#include) серйозно уповільнює компіляцію при підключенні великої кількості модулів. Для усунення цього недоліку багато компіляторів реалізують механізм компіляції заголовних файлів;
* недостача інформації про типи даних під час компіляції;
* мова C++ є складною для вивчення і для компіляції;
* деякі перетворення типів не інтуїтивні. Зокрема, операція над беззнаковим і знаковим числами видає беззнаковий результат.

2.3 Опис СУБД MySQL

MySQL — це система управління базами даних.

База даних являє собою структуровану сукупність даних. Ці дані можуть бути будь-якими — від простого списку майбутніх покупок до переліку експонатів картинної галереї або величезної кількості інформації в корпоративній мережі. Для запису, вибірки й обробки даних, що зберігаються в комп'ютерній базі даних, необхідна система управління базою даних, якою і є ПО MySQL. Оскільки комп'ютери чудово справляються з обробкою великих обсягів даних, управління базами даних відіграє центральну роль в обчисленнях. Реалізовано таке управління може бути по-різному - як у вигляді окремих утиліт, так і у вигляді коду, що входить до складу інших додатків.

MySQL — це система управління реляційними базами даних.

В реляційній базі даних дані зберігаються не всі скопом, а в окремих таблицях, завдяки чому досягається виграш в швидкості і гнучкості. Таблиці зв'язуються між собою за допомогою відносин, завдяки чому забезпечується можливість об'єднувати при виконанні запиту дані з декількох таблиць. SQL як частина системи MySQL можна охарактеризувати як мову структурованих запитів й одну з найвикористовуваних мов для доступу до баз даних.

Програмне забезпечення MySQL — це ПЗ з відкритим кодом.

ПЗ з відкритим кодом означає, що застосовувати і модифікувати його може будь-хто. Таке ПЗ можна отримувати по Internet і використовувати безкоштовно. При цьому кожен користувач може вивчити вихідний код і змінити його відповідно до своїх потреб. Використання програмного забезпечення MySQL регламентується ліцензією GPL (GNU General Public License), в якій зазначено, що можна і чого не можна робити з цим програмним забезпеченням в різних ситуаціях. Якщо робота у рамках GPL вас не влаштовує або планується вбудовування MySQL-коду в комерційний додаток, є можливість купити комерційну ліцензовану версію у компанії MySQL AB. See section 1.6.3 ліцензії на ПЗ MySQL.

MySQL є дуже швидким, надійним і легким у використанні. MySQL має також ряд зручних можливостей, розроблених в тісному контакті з користувачами. Спочатку сервер MySQL розроблявся для управління великими базами даних з метою забезпечити більш високу швидкість роботи в порівнянні з існуючими на той момент аналогами. І ось вже протягом декількох років даний сервер успішно використовується в умовах промислової експлуатації з високими вимогами. Незважаючи на те що MySQL постійно вдосконалюється, він вже сьогодні забезпечує широкий спектр корисних функцій. Завдяки своїй доступності, швидкості і безпеки MySQL дуже добре підходить для доступу до баз даних по Internet.

Технічні можливості СУБД MySQL.

MySQL є системою клієнт-сервер, яка містить багатопоточний SQL-сервер, що забезпечує підтримку різних обчислювальних машин баз даних, а також кілька різних клієнтських програм і бібліотек, засоби адміністрування і широкий спектр програмних інтерфейсів (API). Також поставляється у вигляді багатопоточної бібліотеки, яку можна підключити до користувача додатком і отримати компактний, швидкий і легкий в управлінні продукт.

2.4 Опис основних принципів ООП

Об'єктно-орієнтоване програмування або ООП (object — oriented programming) — методологія програмування, заснована на представленні програми у вигляді сукупності об'єктів кожен з яких являється реалізацією визначеного типу, що використовує механізм пересилки повідомлень і класи організовані в ієрархію наслідування.

Центральний елемент ООП — абстракція. Дані за допомогою абстракції перетворяться в об'єкти, а послідовність обробки цих даних перетворюється на набір повідомлень, що передаються між цими об'єктами. Кожен з об'єктів має свою власну унікальну поведінку. До об'єктів можна звертатися як до окремих сутностей, які реагують на повідомлення, що наказують їм виконати якісь дії.

ООП характеризується наступними принципами (по Алану Кею) :

1. Усе являється об'єктом;
2. Обчислення здійснюються шляхом взаємодії (обміну даними) між об'єктами, при якому один об'єкт вимагає, щоб інший об'єкт виконав деяку дію; об'єкти взаємодіють, посилаючи і отримуючи повідомлення; повідомлення - це запит на виконання дії, доповнений набором аргументів, які можуть знадобитися при виконанні дії;
3. Кожен об'єкт має незалежну пам'ять яка складається з інших об'єктів
4. Кожен об'єкт є представником класуякий виражає загальні властивості об'єктів даного типу
5. У класі задається функціональність (поведінка об'єкту); тим самим усе об'єкти, які являються екземплярами одного класу, можуть виконувати одні і ті ж дії;
6. Класи організовані в єдину деревовидну структуру із загальним коренем, звану ієрархією спадкоємства; пам'ять і поведінка, пов'язана з екземплярами певного класу, автоматично доступні будь-якому класу, розташованому нижче в ієрархічному дереві.

Абстрагування (abstraction) — метод рішення задачі, при якому об'єкти різного роду об'єднуються загальним поняттям (концепцією), а потім згруповані суті розглядаються як елементи єдиної категорії. Абстрагування дозволяє відокремити логічний сенс фрагмента програми від проблеми його реалізації, розділивши зовнішній опис (інтерфейс) об'єкту і його внутрішню організацію (реалізацію).

Інкапсуляція (encapsulation) — техніка, при якій несуттєва з точки зору інтерфейсу об'єкту інформація ховається усередині нього.

Спадкоємство (inheritance) — властивість об'єктів, за допомогою якої екземпляри класу дістають доступ до даних і методів класів-предків без їх повторного визначення. Спадкоємство дозволяє різним типам даних спільно використовувати один і той же код, призводячи до зменшення його розміру і підвищення функціональності.

Поліморфізм (polymorphism) —- властивість, що дозволяє використовувати один і той же інтерфейс для різних дій; поліморфній змінній, наприклад, може відповідати декілька різних методів.

Поліморфізм перекроює загальний код, що реалізовує деякий інтерфейс, так, щоб задовольнити конкретним особливостям окремих типів даних. Клас (class) - безліч об'єктів, пов'язаних спільністю структури і поведінка; абстрактний опис даних і поведінки (методів) для сукупності схожих об'єктів, представники якої називаються екземплярами класу.

Об'єкт (object) — конкретна реалізація класу, що володіє характеристиками стану, поведінки і індивідуальності, синонім екземпляра.

2.5 Опис подібних програмних продуктів

Tourwriter

Tourwriter - це програмне рішення, призначене для туроператорів та компаній, що спеціалізуються на створенні та продажу туристичних путівок та пакетних турів. Ця система допомагає у керуванні всіма аспектами процесу створення, управління та продажу турів, що дозволяє підприємствам оптимізувати свою роботу та покращити обслуговування клієнтів.

Основні функції і можливості Tourwriter включають:

1. Створення турів: Система дозволяє туроператорам створювати різні типи турів, включаючи індивідуальні, групові, культурні, природні та багато інших. Вона надає інструменти для розробки унікальних турів, враховуючи різноманітні вимоги клієнтів.
2. Управління інвентарем: Tourwriter допомагає веденню обліку різноманітних послуг, включаючи готелі, транспорт, екскурсії, квитки та інше. Ви можете додавати нові послуги, визначати ціни, доступність і інші параметри.
3. Планування маршруту: Туроператори можуть розробляти та оптимізувати маршрути для своїх турів. Система дозволяє враховувати різні фактори, такі як час подорожі, відстань між визначеними точками, перебування та активності.
4. Бронювання та календар управління: Tourwriter включає функцію бронювання послуг та керування календарем, що дозволяє вам слідкувати за бронюваннями та управляти доступністю послуг.
5. Фінансовий облік: Система допомагає веденню обліку витрат і прибутків, а також генерації фінансових звітів для керівництва і бухгалтерії.
6. Звіти та аналітика: Tourwriter надає засоби для аналізу роботи туроператора, включаючи звіти про продажі, прибутковість турів та інші показники.
7. Інтеграція: Система може інтегруватися з іншими програмними продуктами та послугами, такими як бронювання авіаквитків та готелів.

Tourwriter - це потужний інструмент для туроператорів, який допомагає автоматизувати багато аспектів бізнесу та підвищити ефективність управління турами. Ця система може бути корисною для туроператорів різних масштабів, які спеціалізуються на різних видках туризму.

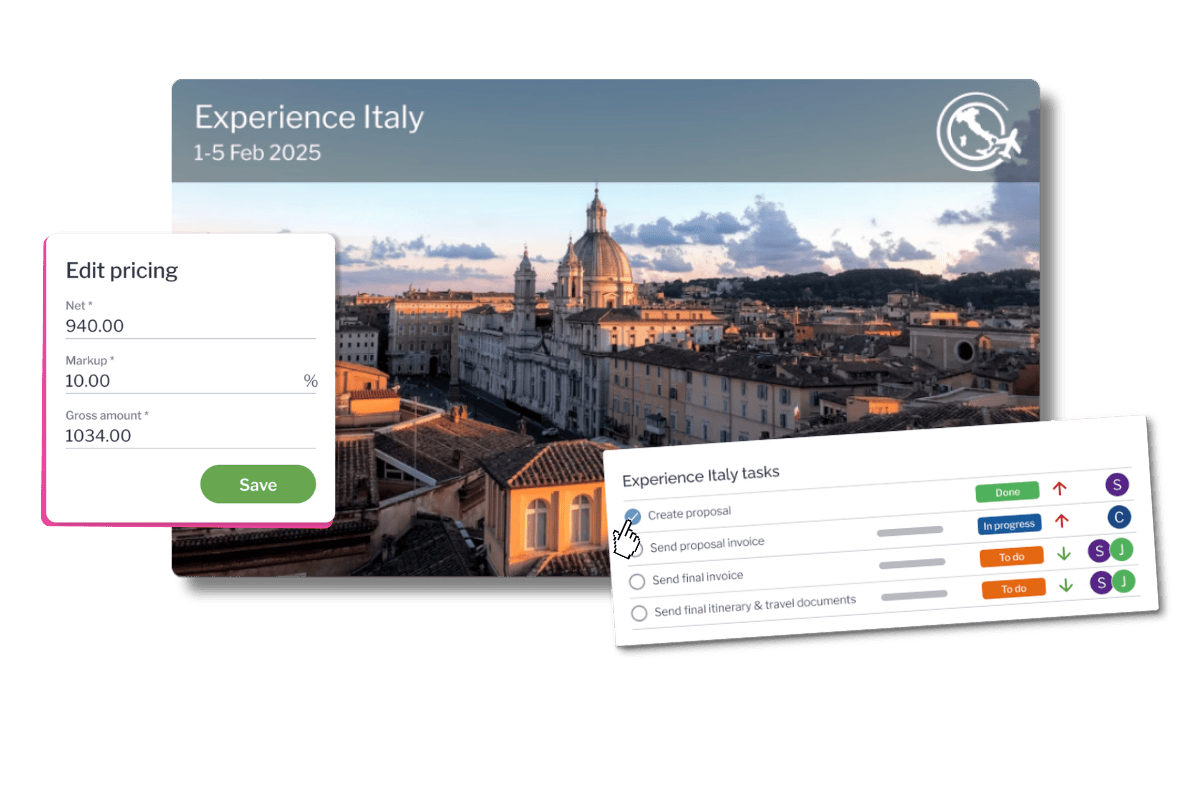


Рисунок 2.1 – Зображення з сайту розробника

TourCMS

TourCMS - це онлайн-система для управління туристичним бізнесом, яка призначена для туроператорів, агентств та екскурсійних компаній. Ця система допомагає в управлінні всіма аспектами туроператорського бізнесу, починаючи від створення турів та бронювання послуг і закінчуючи звітністю та фінансовим обліком. Основні функції і можливості TourCMS включають:

1. Система управління турами: Туроператори можуть створювати та налаштовувати різні тури, включаючи групові та індивідуальні, культурні, природні, активні та інші типи турів. Вони можуть додавати різноманітні послуги, такі як готелі, транспорт, екскурсії тощо
2. Бронювання та онлайн-продажі: TourCMS надає інструменти для приймання бронювань та продажів турів через веб-сайт та інші онлайн-канали. Клієнти можуть вибирати та бронювати тури, вибираючи дати, кількість учасників і інші параметри.
3. Календар управління: Система допомагає слідкувати за доступністю турів та послуг, забезпечуючи користувачам актуальну інформацію про те, що доступно для бронювання.
4. Управління клієнтами: TourCMS містить інструменти для ведення бази клієнтів, історії замовлень та іншої інформації про клієнтів.
5. Фінансовий облік та звіти: Система надає функції для ведення обліку витрат та прибутків, а також генерації фінансових звітів для аналізу роботи туроператора.
6. Сповіщення і повідомлення: TourCMS може надсилати сповіщення і підтвердження бронювань клієнтам і персоналу.
7. Інтеграція і зовнішні API: Система може інтегруватися з іншими програмними продуктами та послугами, а також має відкритий API для розширення функціональності.

TourCMS дозволяє туроператорам керувати всіма аспектами свого бізнесу в одній системі, спрощуючи процеси бронювання, продажу і управління клієнтами. Ця система може бути особливо корисною для малих і середніх туроператорів, які шукають потужний інструмент для автоматизації свого бізнесу.

Изображение выглядит как текст, мультимедиа, снимок экрана, Устройство связи

Автоматически созданное описание

Рисунок 2.2 – Зображення з сайту розробника

Проте розроблювана програма для ведення обліку продажу путівок "Тури по Україні" має свої унікальні переваги, які відзначають її перед описаними системами, такими як TourCMS і Tourwriter. Наприклад орієнтованість на регіонального користувача, що дозволить розробити програму актуальнішою для користувачів в Україні, використання української мови в інтерфейсі програми, можливість надання додаткових функцій з урахуванням особлиивостей українського туризму. Спираючись на це та на відсутність українських аналогів цих програм, вважаю програму актуальною а її розробку доцільною.

3 ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ

3.1 Опис бази даних

Зважаючи на необхідність зберігати великі об’єми інформації, під час проектування доцільно розділити базу даних на декілька сутностей (таблиць).

База даних складається з 7-и таблиць: clients – клієнти, tours – тури, tourist\_sites – туристичні об’єкти, vauchers – путівки, tourist\_sites\_in\_tour – проміжна таблиця для збереження туристичних об’єктів у турі, images\_in\_tourist\_sites – зображення для туристичних об’єктів,  
images\_in\_tours – зображення для турів. ER-діаграма бази даних представлена на рисунку 3.1, перелік таблиць, їх поля та призначення полів наведено у таблицях 3.1 — 3.6.

Таблиця 3.1 – Таблиця clients

|  |  |
| --- | --- |
| Поле | Призначення |
| id | Ідентифікатор |
| pib | ПІБ |
| phone\_number | Номер телефону |
| email | Електронна пошта |

Таблиця 3.2 – Таблиця tours

|  |  |
| --- | --- |
| Поле | Призначення |
| id | Ідентифікатор |
| title | Назва |
| description | Опис |
| date\_event | Дата та час проведення |
| duration | Тривалість |
| type | Тип подорожі |
| number\_seats | Кількість місць |
| cost | Вартість |

Таблиця 3.3 – Таблиця tourist\_sites

|  |  |
| --- | --- |
| Поле | Призначення |
| id | Ідентифікатор |
| title | Назва |
| image | Зображення |
| description | Опис |

Таблиця 3.4 – Таблиця vauchers

|  |  |
| --- | --- |
| Поле | Призначення |
| id | Ідентифікатор |
| idTour | Ідентифікатор туру |
| idClient | Ідентифікатор клієнта |
| number\_seats | Кількість місць |
| sum | Сумма до сплати |
| payment\_status | Статус сплати |
| booking\_date | Дата бронювання |

Таблиця 3.5 – Таблиця tourist\_sites\_in\_tour

|  |  |
| --- | --- |
| Поле | Призначення |
| idTour | Ідентифікатор туру |
| idTouristSite | Ідентифікатор тустичного об’єкта |
| number | Порядковий номер у списку |

Таблиця 3.6 – Таблиця images\_in\_tours

|  |  |
| --- | --- |
| Поле | Призначення |
| id | Ідентифікатор |
| idTour | Ідентифікатор туру |
| image | Зображення |

Изображение выглядит как текст, диаграмма, снимок экрана, Параллельный

Автоматически созданное описание

Рисунок 3.1 – ER-діаграма бази даних

3.2 Проєктування користувацького інтерфейсу

Задачею графічного інтерфейсу програми є забезпечення комфортної взаємодії користувача з програмним додатком. Графічний інтерфейс складається з форм та елементів, які розташовуються на формах. Розробка інтерфейсу програми починається зі створення нової, стартової форми, шляхом внесення змін до властивостей автоматично згенерованої форми (centralWidget), а також шляхом додавання на форму нових необхідних елементів. На форму можуть бути додані різноманітні компоненти для взаємодії з текстом, числовими даними, базами даних тощо. Для перегляду і зміни значень властивостей форми і її компонентів використовується вікно Object Inspector, у якому знаходиться більшість даних про об’єкт, який є активним на даний момент (ім’я об’єкту, шрифт, ширина, висота тощо). Під час проектування дуже важливо цілеспрямовано обрати компоненти.

Інтерфейс програми складається з основної форми, яка має у собі усі необхідні елементи для взаємодії користувача із інформацією, та сторінок, які в залежності від дій користувача змінюють одна одну і мають необхідні компоненти для виконання бажаної дії користувачем.

Інтерфейс програми налічує декілька вікон:

* MainWindows — головне вікно програми, яке в залежності від дій користувача показує відповідні компоненти для виконання бажаної дії. Налічує у собі контролюючі компоненти StackedWidget для відтворення механізму зміни сторінок. Усі сторінки створені у схожими за розташуванням елементів керування, для зручного користування программою;
* AddEditClientsDialog — вікно, що реалізовує інтерфейс програми для додавання або редагування інформації про клієнтів;
* AddEditTouristSitesDialog — вікно, що реалізовує інтерфейс програми для додавання або редагування інформації про туристичні об’єкти;
* AddEditToursDialog — вікно, що реалізовує інтерфейс програми для додавання або редагування інформації про тури;
* AddEditVaushersDialog — вікно, що реалізовує інтерфейс програми для додавання або редагування інформації про білети.

На рисунку 3.2 зображено стартове вікно привітання, що пропонує користувачу почати роботи з вибору дій у меню зверху.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 3.2 – Стартова сторінка програми

На рисунку 3.3 зображено сторінку головного вікна для ведення даних клієнтів, вона має титульний напис, панель пошуку, представлення інформації про клієнтів та кнопки керування

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 3.3 – Сторінка перегляду клієнтів

На рисунку 3.4 зображено стоірнку ведення туристичних турів, вона має титуьний напис, панель пошуку, списочне представлення даних та кнопки керування.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 3.4 – Сторінка перегляду туристичних об’єктів

На рисунку 3.5 зображено сторінку ведення обліку турів. Вона має титульний напис, панель пошуку, табличне представлення даних про тури та кнопки керування.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 3.5 – Сторінка перегляду турів

На рисунку 3.6 зображено сторінку ведення обліку путівок, сторінка має титульний напис, панель пошуку, табличне представлення даних про путівки та кнопки керування.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 3.6 – Сторінка перегляду путівок

На рисунку 3.7 зображено вікно додавання або редагування запису про клієнта. Воно має титульний напис, поля для введення даних про клієнта та кнопки керування.

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 3.7 – Сторінка додавання клієнтів

На рисунку 3.8 зображено вікно додавання та редагування запису туру. Вікно має титульний напис, панель зображення, кнопку для завантаження цього зображення, інші компоненти для введення даних про тур та кнопки керування.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 3.8 – Сторінка перегляду туристичних об’єктів

На рисунку 3.9 зображено вікно додавання або редагування даних про тур. Вікно має титульний напис, сисочне представлення для відображення зображень у турі, списочне представлення туристичних місць у турі, інші компоненти для ведення інформації про тур та кнопки керування.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 3.9 – Вікно додавання та редагування турів

На рисунку 3.10 зображено вікно додавання або редагування путівки. Вікно має титульний напис, компоненти для ведення інформації про путівку та кнопки керування.

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 3.10 – Сторінка додавання та редагування путівок

На рисунку 3.11 зображено вікно перегляду звітності. Воно має титуьний напис, панель, з сторінками для переходу по різним типам звітності та табличні представлення для звітності.

Изображение выглядит как снимок экрана, текст, программное обеспечение, компьютер

Автоматически созданное описание

Рисунок 3.11 – Сторінка звітності

4 ІНСТРУКЦІЯ З КОРИСТУВАННЯ ПРОГРАМНОЮ СИСТЕМОЮ

Робота з програмою починається у головному вікні, в якому відображено стартову сторінку із запитом, щодо вибору необхідної дії у меню зверху. Оскільки программа призначена для одного користувача, то авторизація не потребується. У меню зверху відображені усі доступні пункти меню, а при виборі конкретного, він буде підсвічуватись, демонструючи на якій сторінці знаходиться користувач. При відкритті программи та відображенні стартової сторінки, жоден з пунктів меню не підсвічується. Стартову сторінку зображено на рисунку 4.1.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 4.1 – Стартова сторінка головного вікна

На сторінці клієнтів, зображено таблицю з поточними даними про клієнтів, над нею розташоване поле пошуку, а під табличкою кнопки для взаємодії з даними. Сторінку перегляду даних клієнтів зображено на рисунку 4.2.

Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание

Рисунок 4.2 – Сторінка перегляду даних клієнтів

Для пошуку записів у таблиці, необхідно заповнити одне або декілька полей пошуку та натиснути відповідну кнопку.

Для редагування запису, необхідно обрати потрібний запис із таблиці та натиснути кнопку «Редагувати», після чого відкриється вікно редагування даних клієнта. Демонстрацію редагування зображено на рисунку 4.3.

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 4.3 – Вікно редагування даних клієнта

Для збереження змін необхідно натиснути кнопку «Зберегти», для скасування «Скасувати».

Для видалення запису про користувача слід обрати його із загальної таблички користувачів та натиснути кнопку «Видалити».

За додавання нових записів клієнтів відповідає однойменна кнопка на сторінці, при натисканні на яку вікриється вікно додавання клієнтів, робота з ним аналогічна роботі із вікном редагування, тому демонстрації непотребує.

Наступною йде сторінка роботи з туристичними об’єктами, яка зроблена відовідно до сторінки з клієнтами, де в центрі відображено усі записи про об’єкти, зверху поля для пошуку інфформації, а з низу кнопки керування. Сторінку зображено на рисунку 4.4.

Для додаваня нового запису необхідно натиснути кнопку «Додати» після чого відкриється вікно додавання туристичного об’єкту. Вікно заображено на рисунку 4.5.

Для туристичного об’єкта є можливість додати зображення, для цього необхідно натиснути кнопку завантажити та обрати необхідне зображення у системі, після чого воно з’явиться на формі.

Для збереження змін необхідно натиснути кнопку «Зберегти», а для скасування «Скасувати».

Для редагування записів необхідно обрати запис для редагування та натиснути відповідну кнопку, після чого відкриється вікно редагування запису. Оскілько воно розроблене відповідно до вікна додавання, то демонстрації не потребує.

Для видалення запису необхідно обрати запис для видалення та натиснути кнопку «Видалити».

Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание

Рисунок 4.4 – Сторінка роботи з туристичними об’єктами

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 4.5 – Вікно додавання запису туристичного об’єкта

Наступною йде сторінка обліку турів. Вона розроблена відповідно до попередніх сторінок та має поле пошуку, список записів та кнопки керування Сторінку зображено на рисунку 4.6.

Для додавання запису необхвдно натиснути відповідну кнопку після чого відкриється вікно додавання туру. Вікно зображено на рисунку 4.7.

При додаванні туру є можливість додати до нього зображення, для цього необхідно натиснути кнопку «Завантажити» та обрати необхідний файл у системі, після чого зображення з’явиться у загальному списку зображень. Для видалення непотрібного зображення необхідно обрати його у списку та натиснути кнопку «Видалити».

Для додавання туристичного об’єкту до туру, необхідно натиснути кнопку додати, та обрати потрібний тур зі списку. Для його видалення, необхідно обрати об’єкт зі списку та натиснути кнопку «Видалення».

По завершенню заповнення полів для збереження запису необхідно натиснути кнопку «Зберегти», для скасування додавання натиснути кнопку «Скасувати».

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок 4.6 – Сторінка роботи з турами

Для редагування запису необхідно обрати запис для редагування у таблиці та натиснути кнопку «Редагувати», після чого відкриється вікно редагування. Вікно редагування ідентичне вікну додавання тому демонстрацї не потребує.

Для видалення запису, необхідно обрати відповідний запис за списку та натиснути кнопку «Видалити».

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Рисунок 4.7 – Вікно додавання туру

Наступною йде сторінка обліку путівок. Розміщення компонентів зроблено відповідно, до попередніх сторінок, де зверху йде поле пошуку, наступним таблиця зі списком білетів, а нижче кнопки керування.

Для пошуку необхідно заповнити відповідні поля на натиснути кнопку «Пошук». Для відміни фільтра треба зкинути значення усіх полів пошуку та знову натиснути кнопку «Пошук». Сторінку перегляду путівок зображено на рисунку 4.8.

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок 4.8 – Сторінка перегляду путівок

Для редагування запису необхідно обрати запис для редагування та натиснути кнопку «Редагування». Вікно редагування послуги зображено на рисунку 4.9.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 4.9 – Вікно редагування путівок

Після редагування запису для збереження змін необхідно натиснути кнопку «Зберегти», для скасування змін необхідно натиснути кнопку «Скасувати».

Для додавання запису необхідно натиснути відповідну кнопку, після чого відкриється вікно додавання путівки. Вікно додавання розроблено відповідно довікна редагування тому демонстацію не потребує.

Для видалення запису, необхідно обрати відповідний запис та натиснути кнопку «Видалити».

Програма також має можливість ведення звітності, для цього на панелі меню необхідно обрати пункт «Звітність», після чого відкриється відповідна сторінка. Сторінку звітності зображено на рисунку 4.10.

Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание

Рисунок 4.10 – Сторінка звітності

Є можливість переглянути звіти за затребованістю турів, затребуваності туристичних об’єктів та переглянути розрахунок прибутків по турах. Навігація по звітності відбувається за допомогою натискання відповідних ярликів на вікні звітності.

Для завершення роботи програми необхідно натиснути кнопку «Вихід» у меню.

ВИСНОВКИ

У результаті виконання цього курсового проєкту з розробки програмного додатка для ведення обліку продажу путівок "Тури по Україні" можна визначити, що створення такого інструменту є важливим етапом в удосконаленні та автоматизації процесів у сфері туризму. Проект передбачає розробку функціональності, спрямованої на полегшення роботи туроператорів, покращення обслуговування клієнтів та підвищення ефективності бізнесу.

Введення програмного додатка має на меті вирішити проблеми, пов'язані з ручним веденням обліку, організацією інформації та оптимізацією робочих процесів туроператорів. Система надає можливість створювати та керувати різноманітними турами, вести облік туристичних об’єктів, здійснювати бронювання та вести фінансовий облік.

Програмний додаток "Тури по Україні" спрямований на підтримку українських туроператорів та сприяння розвитку внутрішнього туризму. Його використання дозволить підприємствам оптимізувати роботу, забезпечити більш точний та швидкий облік, а також підняти якість обслуговування клієнтів.

Усе це робить розроблений програмний продукт важливим інструментом для туроператорської галузі, сприяючи покращенню професійної діяльності та розвитку туристичного бізнесу в Україні.

В процесі розробки програмного забезпечення було закріплено навички роботи з інструментрієм QtFramework, MySQL та візуального оформлення багатовіконних додатків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Відео та лекції по виченню Qt Creatorehttps://resources.qt.io/resources-by-content-type-videos-demos-tutorials.

2 Електронний научно-практический журнал «Современная техника и технологии» [Електронний ресурс] // technology.snauka.ru. Режим доступу: http://technology.snauka.ru/2016/12/11465

3 М. Саммерфілд «Qt. Профессиональное программирование. Разработка кроссплатформенных приложений на С++», Символ-Плюс, 2011

4 Ж. Бланшет, М. Саммерфілд «QT 4: программирование GUI на С++», КУДИЦ-Пресс, 2008

5 М. Шлеє «Qt 5.3. Профессиональное программирование на C++», БХВ-Петербург, 2015

6 Офіційна документація — doc.qt.io

Додаток А

Лістинг програми

Лістинг А.1 – Клас роботи з базою даних

#ifndef DATABASEMODULE\_H

#define DATABASEMODULE\_H

#include <QObject>

#include <QSqlDatabase>

#include <QSqlTableModel>

#include <QSqlError>

#include <QSqlQuery>

#include "clientsmodel.h"

#include "imageintoursmodel.h"

#include "touristsitesintourmodel.h"

#include "touristsitesmodel.h"

#include "toursmodel.h"

#include "vauchersmodel.h"

class DatabaseModule : public QObject

{

Q\_OBJECT

QSqlDatabase db;

public:

DatabaseModule(QObject \*parent = nullptr);

bool connect();

void disconnect() { db.close(); }

QSqlError lastError() { return db.lastError(); }

void setDbName(QString name) { db.setDatabaseName(name); }

void setHostAddress(QString address) { db.setHostName(address); }

void setHostPort(int port) { db.setPort(port); }

void selectTables();

ClientsModel\* clientsModel;

ImageInToursModel\* imageInToursModel;

TouristSitesInTourModel\* touristSitesInTourModel;

TouristSitesModel\* touristSitesModel;

ToursModel\* toursModel;

VauchersModel\* vauchersModel;

};

#endif // DATABASEMODULE\_H

#include "databasemodule.h"

#include <QDebug>

DatabaseModule::DatabaseModule(QObject \*parent)

{

db = QSqlDatabase::addDatabase("QMARIADB");

db.setHostName("localhost");

db.setPort(3306);

db.setDatabaseName("tours\_of\_ukrain");

db.setUserName("dtours\_of\_ukrain\_user");

db.setPassword("tours\_of\_ukrain\_user");

}

bool DatabaseModule::connect()

{

if(db.open())

{

clientsModel = new ClientsModel(this);

clientsModel->setTable("clients", db);

imageInToursModel = new TouristSitesInTourModel(this);

imageInToursModel->setTable("managers", db);

touristSitesInTourModel = new TouristSitesInTourModel(this);

touristSitesInTourModel->setTable("types\_of\_services", db);

touristSitesModel = new TouristSitesModel(this);

touristSitesModel->setTable("services", db);

toursModel = new ToursModel(this);

toursModel->setTable("files", db);

vauchersModel = new VauchersModel(this);

vauchersModel->setTable("orders", db);

ordersServicesModel = new OrdersServicesModel(this);

ordersServicesModel->setTable("orders\_services", db);

statusesModel = new StatusesModel(this);

statusesModel->setTable("statuses", db);

return true;

}

else

return false;

}

void DatabaseModule::selectTables()

{

clientsModel->select();

managersModel->select();

typeOfServicesModel->select();

servicesModel->select();

filesModel->select();

ordersModel->select();

ordersServicesModel->select();

statusesModel->select();

}

Лістинг А.2 – Клас головного вікна

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QMessageBox>

#include "databasemodule.h"

#include "clientsfiltermodel.h"

#include "touristsitesfiltermodel.h"

#include "toursfiltermodel.h"

#include "vauchersfiltermodel.h"

#include "addeditclientsdialog.h"

#include "addedittouristsitesdialog.h"

#include "addedittoursdialog.h"

#include "addeditvauchersdialog.h"

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

DatabaseModule dbm;

ClientsFilterModel\* clientsFilterModel;

TouristSitesFilterModel\* touristSitesFilterModel;

ToursFilterModel\* toursFilterModel;

VauchersFilterModel\* vauchersFilterModel;

enum PAGES {

HELLO = 0,

CLIENTS,

TOURIST\_SITES,

TOURS,

VAUCHERS,

STATISTIC

};

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

private slots:

void on\_action\_clients\_triggered();

void on\_action\_tours\_objects\_triggered();

void on\_action\_tours\_triggered();

void on\_action\_vauchers\_triggered();

void on\_pushButton\_clients\_search\_clicked();

void on\_pushButton\_tourist\_sites\_search\_clicked();

void on\_pushButton\_tours\_search\_clicked();

void on\_pushButton\_vauchers\_search\_clicked();

void on\_pushButton\_vauchers\_add\_clicked();

void on\_pushButton\_vauchers\_edit\_clicked();

void on\_pushButton\_vauchers\_delete\_clicked();

void on\_pushButton\_clients\_add\_clicked();

void on\_pushButton\_clients\_edit\_clicked();

void on\_pushButton\_clients\_delete\_clicked();

void on\_pushButton\_tourist\_sites\_add\_clicked();

void on\_pushButton\_tourist\_sites\_edit\_clicked();

void on\_pushButton\_tourist\_sites\_delete\_clicked();

void on\_pushButton\_tours\_add\_clicked();

void on\_pushButton\_tours\_edit\_clicked();

void on\_pushButton\_tours\_delete\_clicked();

void on\_action\_stats\_triggered();

private:

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

if(!dbm.connect())

{

QMessageBox::critical(nullptr, "Критична помилка!", "Неможливо під'єднатися до бази даних.\nПрограма буде завершена.");

close();

}

ui->stackedWidget->setCurrentIndex(PAGES::HELLO);

dbm.selectTables();

clientsFilterModel = new ClientsFilterModel(this);

touristSitesFilterModel = new TouristSitesFilterModel(this);

toursFilterModel = new ToursFilterModel(this);

vauchersFilterModel = new VauchersFilterModel(this);

clientsFilterModel->setSourceModel(dbm.clientsModel);

touristSitesFilterModel->setSourceModel(dbm.touristSitesModel);

toursFilterModel->setSourceModel(dbm.toursModel);

vauchersFilterModel->setSourceModel(dbm.vauchersModel);

ui->tableView\_clients->setModel(clientsFilterModel);

ui->listView\_tourist\_sites->setModel(touristSitesFilterModel);

ui->tableView\_tours->setModel(toursFilterModel);

ui->tableView\_vauchers->setModel(vauchersFilterModel);

ui->tableView\_zap\_1->setModel(dbm.zap1);

ui->tableView\_zap\_2->setModel(dbm.zap2);

ui->tableView\_zap\_3->setModel(dbm.zap3);

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::on\_action\_clients\_triggered()

{

ui->stackedWidget->setCurrentIndex(CLIENTS);

ui->action\_tours->setChecked(false);

ui->action\_tours\_objects->setChecked(false);

ui->action\_vauchers->setChecked(false);

ui->action\_stats->setChecked(false);

}

void MainWindow::on\_action\_tours\_objects\_triggered()

{

ui->stackedWidget->setCurrentIndex(TOURIST\_SITES);

ui->action\_tours->setChecked(false);

ui->action\_clients->setChecked(false);

ui->action\_vauchers->setChecked(false);

ui->action\_stats->setChecked(false);

}

void MainWindow::on\_action\_tours\_triggered()

{

ui->stackedWidget->setCurrentIndex(TOURS);

ui->action\_clients->setChecked(false);

ui->action\_tours\_objects->setChecked(false);

ui->action\_vauchers->setChecked(false);

ui->action\_stats->setChecked(false);

}

void MainWindow::on\_action\_vauchers\_triggered()

{

ui->stackedWidget->setCurrentIndex(VAUCHERS);

ui->action\_tours->setChecked(false);

ui->action\_tours\_objects->setChecked(false);

ui->action\_clients->setChecked(false);

ui->action\_stats->setChecked(false);

}

void MainWindow::on\_pushButton\_clients\_search\_clicked()

{

if(ui->lineEdit\_pib\_clients\_search->text().size() == 0)

clientsFilterModel->setEnabledFilterParam(ClientsFilterModel::FilterParam::PIB, false);

else

{

clientsFilterModel->setEnabledFilterParam(ClientsFilterModel::FilterParam::PIB, true);

clientsFilterModel->setFilterParam(ClientsFilterModel::FilterParam::PIB, ui->lineEdit\_pib\_clients\_search->text());

}

}

void MainWindow::on\_pushButton\_tourist\_sites\_search\_clicked()

{

if(ui->lineEdit\_tourist\_sites\_title\_search->text().size() > 0)

{

touristSitesFilterModel->setEnabledFilterParam(TouristSitesFilterModel::FilterParam::TITLE, true);

touristSitesFilterModel->setFilterParam(TouristSitesFilterModel::FilterParam::TITLE, ui->lineEdit\_tourist\_sites\_title\_search->text());

}

else

touristSitesFilterModel->setEnabledFilterParam(TouristSitesFilterModel::FilterParam::TITLE, false);

}

void MainWindow::on\_pushButton\_tours\_search\_clicked()

{

if(ui->lineEdit\_tours\_title\_search->text().size() > 0)

{

toursFilterModel->setEnabledFilterParam(ToursFilterModel::FilterParam::TITLE, true);

toursFilterModel->setFilterParam(ToursFilterModel::FilterParam::TITLE, ui->lineEdit\_tours\_title\_search->text());

}

else

toursFilterModel->setEnabledFilterParam(ToursFilterModel::FilterParam::TITLE, false);

}

void MainWindow::on\_pushButton\_vauchers\_search\_clicked()

{

if(ui->lineEdit\_vauchers\_id\_search->text().size() > 0)

{

vauchersFilterModel->setEnabledFilterParam(VauchersFilterModel::FilterParam::ID, true);

vauchersFilterModel->setFilterParam(VauchersFilterModel::FilterParam::ID, ui->lineEdit\_vauchers\_id\_search->text());

}

else

vauchersFilterModel->setEnabledFilterParam(VauchersFilterModel::FilterParam::ID, false);

}

void MainWindow::on\_pushButton\_vauchers\_add\_clicked()

{

AddEditVauchersDialog dialog;

dialog.setDialog(dbm.clientsModel, dbm.toursModel);

dialog.exec();

if(!dialog.isSave())

return;

dbm.vauchersModel->appendRow(dialog.getData());

dbm.vauchersModel->saveChanges();

dbm.vauchersModel->select();

}

void MainWindow::on\_pushButton\_vauchers\_edit\_clicked()

{

if(ui->tableView\_vauchers->currentIndex().row() == -1)

return;

AddEditVauchersDialog dialog;

dialog.setDialog(dbm.vauchersModel->getDataByRow(vauchersFilterModel->mapToSource(ui->tableView\_vauchers->currentIndex()).row()), dbm.clientsModel, dbm.toursModel);

dialog.exec();

if(!dialog.isSave())

return;

dbm.vauchersModel->updateRow(vauchersFilterModel->mapToSource(ui->tableView\_vauchers->currentIndex()).row(), dialog.getData());

dbm.vauchersModel->saveChanges();

dbm.vauchersModel->select();

}

void MainWindow::on\_pushButton\_vauchers\_delete\_clicked()

{

if(ui->tableView\_vauchers->currentIndex().row() == -1)

return;

dbm.vauchersModel->removeRow(vauchersFilterModel->mapToSource(ui->tableView\_vauchers->currentIndex()).row());

dbm.vauchersModel->saveChanges();

dbm.vauchersModel->select();

}

void MainWindow::on\_pushButton\_clients\_add\_clicked()

{

AddEditClientsDialog dialog;

dialog.exec();

if(!dialog.isSave())

return;

dbm.clientsModel->appendRow(dialog.getData());

dbm.clientsModel->saveChanges();

dbm.clientsModel->select();

}

void MainWindow::on\_pushButton\_clients\_edit\_clicked()

{

if(ui->tableView\_clients->currentIndex().row() == -1)

return;

AddEditClientsDialog dialog;

dialog.setDialog(dbm.clientsModel->getDataByRow(clientsFilterModel->mapToSource(ui->tableView\_clients->currentIndex()).row()));

dialog.exec();

if(!dialog.isSave())

return;

dbm.clientsModel->updateRow(clientsFilterModel->mapToSource(ui->tableView\_clients->currentIndex()).row(), dialog.getData());

dbm.clientsModel->saveChanges();

dbm.clientsModel->select();

}

void MainWindow::on\_pushButton\_clients\_delete\_clicked()

{

if(ui->tableView\_clients->currentIndex().row() == -1)

return;

dbm.clientsModel->removeRow(clientsFilterModel->mapToSource(ui->tableView\_clients->currentIndex()).row());

dbm.clientsModel->saveChanges();

dbm.clientsModel->select();

}

void MainWindow::on\_pushButton\_tourist\_sites\_add\_clicked()

{

AddEditTouristSitesDialog dialog;

dialog.exec();

if(!dialog.isSave())

return;

dbm.touristSitesModel->appendRow(dialog.getData());

dbm.touristSitesModel->saveChanges();

dbm.touristSitesModel->select();

}

void MainWindow::on\_pushButton\_tourist\_sites\_edit\_clicked()

{

if(ui->listView\_tourist\_sites->currentIndex().row() == -1)

return;

AddEditTouristSitesDialog dialog;

dialog.setDialog(dbm.touristSitesModel->getDataByRow(touristSitesFilterModel->mapToSource(ui->listView\_tourist\_sites->currentIndex()).row()));

dialog.exec();

if(!dialog.isSave())

return;

dbm.touristSitesModel->updateRow(touristSitesFilterModel->mapToSource(ui->listView\_tourist\_sites->currentIndex()).row(), dialog.getData());

dbm.touristSitesModel->saveChanges();

dbm.touristSitesModel->select();

}

void MainWindow::on\_pushButton\_tourist\_sites\_delete\_clicked()

{

if(ui->listView\_tourist\_sites->currentIndex().row() == -1)

return;

dbm.touristSitesModel->removeRow(touristSitesFilterModel->mapToSource(ui->listView\_tourist\_sites->currentIndex()).row());

dbm.touristSitesModel->saveChanges();

dbm.touristSitesModel->select();

}

void MainWindow::on\_pushButton\_tours\_add\_clicked()

{

AddEditToursDialog dialog;

dialog.setDialog(dbm.touristSitesModel, dbm.touristSitesInTourModel, dbm.imageInToursModel);

dialog.exec();

dbm.toursModel->appendRow(dialog.getData());

QVariant id = dbm.toursModel->saveChanges();

dbm.toursModel->select();

QList <touristSitesInTourObject> listToursSites = dialog.getTouristSitesData();

for(int i = 0; i < listToursSites.size(); i++)

{

listToursSites[i][TouristSitesInTourModel::Column::ID\_TOUR] = id;

dbm.touristSitesInTourModel->appendRow(listToursSites[i]);

}

dbm.touristSitesInTourModel->saveChanges();

dbm.touristSitesInTourModel->select();

QList <imageInTourObject> listImages = dialog.getImagesData();

for(int i = 0; i < listImages.size(); i++)

{

listImages[i][ImageInToursModel::Column::ID\_TOUR] = id;

dbm.imageInToursModel->appendRow(listImages[i]);

}

dbm.imageInToursModel->saveChanges();

dbm.imageInToursModel->select();

}

void MainWindow::on\_pushButton\_tours\_edit\_clicked()

{

if(ui->tableView\_tours->currentIndex().row() == -1)

return;

AddEditToursDialog dialog;

dialog.setDialog(dbm.toursModel->getDataByRow(toursFilterModel->mapToSource(ui->tableView\_tours->currentIndex()).row()), dbm.touristSitesModel, dbm.touristSitesInTourModel, dbm.imageInToursModel);

dialog.exec();

QVariant id = dbm.toursModel->getDataByRow(toursFilterModel->mapToSource(ui->tableView\_tours->currentIndex()).row())[ToursModel::Column::ID];

dbm.toursModel->updateRow(toursFilterModel->mapToSource(ui->tableView\_tours->currentIndex()).row(), dialog.getData());

qDebug() << toursFilterModel->mapToSource(ui->tableView\_tours->currentIndex()).row();

dbm.toursModel->saveChanges();

dbm.toursModel->select();

dbm.touristSitesInTourModel->removeRow(id);

dbm.touristSitesInTourModel->saveChanges();

dbm.touristSitesInTourModel->select();

QList <touristSitesInTourObject> listToursSites = dialog.getTouristSitesData();

for(int i = 0; i < listToursSites.size(); i++)

{

listToursSites[i][TouristSitesInTourModel::Column::ID\_TOUR] = id;

dbm.touristSitesInTourModel->appendRow(listToursSites[i]);

}

dbm.touristSitesInTourModel->saveChanges();

dbm.touristSitesInTourModel->select();

dbm.imageInToursModel->removeRow(id);

dbm.imageInToursModel->saveChanges();

dbm.imageInToursModel->select();

QList <imageInTourObject> listImages = dialog.getImagesData();

for(int i = 0; i < listImages.size(); i++)

{

listImages[i][ImageInToursModel::Column::ID\_TOUR] = id;

dbm.imageInToursModel->appendRow(listImages[i]);

}

dbm.imageInToursModel->saveChanges();

dbm.imageInToursModel->select();

}

void MainWindow::on\_pushButton\_tours\_delete\_clicked()

{

if(ui->tableView\_tours->currentIndex().row() == -1)

return;

dbm.toursModel->removeRow(toursFilterModel->mapToSource(ui->tableView\_tours->currentIndex()).row());

dbm.toursModel->saveChanges();

dbm.toursModel->select();

}

void MainWindow::on\_action\_stats\_triggered()

{

ui->action\_tours->setChecked(false);

ui->action\_tours\_objects->setChecked(false);

ui->action\_clients->setChecked(false);

ui->action\_vauchers->setChecked(false);

dbm.query\_1.exec();

dbm.query\_2.exec();

dbm.query\_3.exec();

qDebug() << dbm.zap1->lastError();

ui->stackedWidget->setCurrentIndex(PAGES::STATISTIC);

}

Лістинг А.3 – Клас моделі клієнтів

#ifndef CLIENTSMODEL\_H

#define CLIENTSMODEL\_H

#include <QSqlDatabase>

#include <QSqlQuery>

#include <QSqlError>

#include <QAbstractTableModel>

#include <QObject>

#include <QDebug>

class ClientsModel : public QAbstractTableModel

{

Q\_OBJECT

public:

enum Column {

ID = 0,

PIB,

PHONE\_NUMBER,

EMAIL,

IS\_DELETE,

LAST,

STATE\_ROW

};

typedef QHash< ClientsModel::Column, QVariant > clientObject;

int rowCount( const QModelIndex& parent ) const;

int columnCount( const QModelIndex& parent ) const;

QVariant data( const QModelIndex& index, int role ) const;

bool setData( const QModelIndex& index, const QVariant& value, int role );

QVariant headerData( int section, Qt::Orientation orientation, int role ) const;

Qt::ItemFlags flags( const QModelIndex& index ) const;

void appendRow( const clientObject& newClient );

void updateRow( int row, const clientObject& client );

void removeRow( int row );

bool select();

bool saveChanges();

void setTable(QString t, QSqlDatabase \*database);

enum Role {

Display = 0x1,

Read

};

enum StatesRows {

NOT\_EDITED = 0x1,

ADDED,

EDITED,

DELETED

};

QVariant getDataById(int id, Column column);

clientObject getDataByRow(int row);

int getRowByData(QVariant data, Column column);

private:

typedef QList< clientObject > DataList;

DataList model;

QSqlDatabase \*db;

QSqlQuery query;

QString table;

public:

explicit ClientsModel(QObject \*parent = nullptr);

};

typedef QHash< ClientsModel::Column, QVariant > clientObject;

#endif // CLIENTSMODEL\_H

#include "clientsmodel.h"

ClientsModel::ClientsModel(QObject \*parent)

: QAbstractTableModel{parent}

{

}

int ClientsModel::rowCount( const QModelIndex& parent ) const

{

Q\_UNUSED( parent )

return model.count();

}

int ClientsModel::columnCount( const QModelIndex& parent ) const

{

Q\_UNUSED( parent )

return LAST;

}

QVariant ClientsModel::headerData( int section, Qt::Orientation orientation, int role ) const

{

if( role != Qt::DisplayRole ) {

return QVariant();

}

if( orientation == Qt::Vertical ) {

return section;

}

switch( section ) {

case ID:

return "ID";

case PIB:

return "ПІБ";

case PHONE\_NUMBER:

return "Номер телефону";

case EMAIL:

return "Пошта";

}

return QVariant();

}

QVariant ClientsModel::data( const QModelIndex& index, int role ) const {

if(!index.isValid())

return QVariant();

switch (role)

{

case Qt::DisplayRole:

{

return model[ index.row() ][ Column( index.column() ) ];

}

case ClientsModel::Role::Read:

{

return model[ index.row() ][ Column( index.column() ) ];

}

default:

{

return QVariant();

}

}

}

Qt::ItemFlags ClientsModel::flags( const QModelIndex& index ) const {

Qt::ItemFlags flags = QAbstractTableModel::flags( index );

return flags;

}

void ClientsModel::appendRow( const clientObject& newClient ) {

clientObject record;

record[ ID ] = newClient[ ID ];

record[ PIB ] = newClient[ PIB ];

record[ PHONE\_NUMBER ] = newClient[ PHONE\_NUMBER ];

record[ EMAIL ] = newClient[ EMAIL ];

record[ STATE\_ROW ] = (int)StatesRows::ADDED;

int row = model.count();

beginInsertRows( QModelIndex(), row, row );

model.append( record );

endInsertRows();

}

void ClientsModel::updateRow( int row, const clientObject& client )

{

beginResetModel();

model[ row ][ ID ] = client[ ID ];

model[ row ][ PIB ] = client[ PIB ];

model[ row ][ PHONE\_NUMBER ] = client[ PHONE\_NUMBER ];

model[ row ][ EMAIL ] = client[ EMAIL ];

model[ row ][ STATE\_ROW ] = (int)StatesRows::EDITED;

endResetModel();

}

void ClientsModel::removeRow(int row)

{

model[ row ][ STATE\_ROW ] = StatesRows::DELETED;

}

bool ClientsModel::select()

{

beginResetModel();

beginRemoveRows(createIndex(0, 0), 0, model.count());

while(model.count() != 0)

model.removeFirst();

endRemoveRows();

query.prepare(QString("SELECT \* FROM %1").arg(table));

query.exec();

if(query.next())

{

int row = model.count();

beginInsertRows( createIndex(0, 0), row, row+query.size()-1 );

clientObject record;

do

{

record[ ID ] = query.value( ID );

record[ PIB ] = query.value( PIB );

record[ PHONE\_NUMBER ] = query.value( PHONE\_NUMBER );

record[ EMAIL ] = query.value( EMAIL );

record[ IS\_DELETE ] = query.value( IS\_DELETE );

record[ STATE\_ROW ] = StatesRows::NOT\_EDITED;

model.append( record );

}while(query.next());

endInsertRows();

}

endResetModel();

return false;

}

bool ClientsModel::saveChanges()

{

for(int i = 0; i < model.size(); i++)

{

if(model[ i ][ STATE\_ROW ] != StatesRows::NOT\_EDITED)

{

qDebug() << "!= NOT\_EDIT";

if(model[ i ][ STATE\_ROW ] == StatesRows::ADDED)

{

qDebug() << "ADDED";

query.prepare(QString("INSERT INTO %1 (pib, phone\_number, email, isDelete) VALUES (:pib, :phone\_number, :email, 0)").arg(table));

query.bindValue(":pib", model[ i ][ PIB ]);

query.bindValue(":phone\_number", model[ i ][ PHONE\_NUMBER ]);

query.bindValue(":email", model[ i ][ EMAIL ]);

query.exec();

}

else if(model[ i ][ STATE\_ROW ] == StatesRows::EDITED)

{

qDebug() << "EDITED";

query.prepare(QString("UPDATE %1 SET pib = :pib, phone\_number = :phone\_number, email = :email WHERE id = :id").arg(table));

query.bindValue(":pib", model[ i ][ PIB ]);

query.bindValue(":phone\_number", model[ i ][ PHONE\_NUMBER ]);

query.bindValue(":email", model[ i ][ EMAIL ]);

query.bindValue(":id", model[ i ][ ID ]);

query.exec();

}

else if(model[ i ][ STATE\_ROW ] == StatesRows::DELETED)

{

qDebug() << "DELETED";

query.prepare(QString("UPDATE %1 SET isDelete = 1 WHERE id = :id").arg(table));

query.bindValue(":id", model[ i ][ ID ]);

query.exec();

}

}

}

qDebug() << query.lastError();

return true;

}

void ClientsModel::setTable(QString t, QSqlDatabase \*database)

{

table = t;

db = database;

}

QVariant ClientsModel::getDataById(int id, Column column)

{

for(int i = 0; i < model.size(); i++)

{

if(model[i][ID].toInt() == id)

return model[i][column];

}

return QVariant();

}

ClientsModel::clientObject ClientsModel::getDataByRow(int row)

{

return model[row];

}

int ClientsModel::getRowByData(QVariant data, Column column)

{

for(int i = 0; i < model.size(); i++)

if(model[i][column] == data)

return i;

}

bool ClientsModel::setData( const QModelIndex& index, const QVariant& value, int role ) {

if( !index.isValid() || role != Qt::EditRole || model.count() <= index.row() ) {

return false;

}

model[ index.row() ][ Column( index.column() ) ] = value;

emit dataChanged( index, index );

return true;

}