ТЕХНІЧНИЙ КОЛЕДЖ ТЕРНОПІЛЬСЬКОГО НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ ІМЕНІ ІВАНА ПУЛЮЯ

Циклова комісія програмних систем і комплексів

**КУРСОВА РОБОТА**

з дисципліни:

**«Об’єктно-орієнтоване програмування»**

на тему: **«Програма автоматизації кас залізничного вокзалу»**

Студента  3  курсу групи КН-321 спеціальності 112 «Комп’ютерні науки та інформаційні технології»

Сташків В.І.

(прізвище та ініціали)

Керівник: викладач   Слободян  Р.О.         
Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Члени комісії: \_\_\_\_\_\_\_\_\_\_\_ Г.Я. Марціяш

(підпис)

\_\_\_\_\_\_\_\_\_ Р.О. Слободян

(підпис)

м. Тернопіль – 2018

ЗМІСТ

[ВСТУП](#_Toc497455956) 5

[1 ТЕХНІЧНЕ ЗАВДАННЯ](#_Toc497455957)  7

[1.1 Найменування та область застосування 7](#_Toc497455958)

[1.2 Підстави для розробки 7](#_Toc497455959)

[1.3 Призначення розробки 7](#_Toc497455960)

[1.4 Вимоги до програми чи програмного виробу](#_Toc497455961)  8

[1.5 Вимоги до програмної документації](#_Toc497455962)  12

[1.6 Техніко-економічні показники 12](#_Toc497455963)

[1.7 Стадії та етапи розробки 12](#_Toc497455964)

[1.8 Порядок контролю та прийому 14](#_Toc497455965)

[2 РОЗРОБКА ТЕХНІЧНОГО ТА РОБОЧОГО ПРОЕКТУ 15](#_Toc497455966)

[2.1 Розробка загальної структури і варіантів використання програми 15](#_Toc497455967)

[2.2 Розробка системи класів 17](#_Toc497455968)

[2.3 Розробка методів 19](#_Toc497455969)

[2.4 Проектування і опис інтерфейсу користувача 21](#_Toc497455970)

[2.5 Опис файлової структури програми 25](#_Toc497455971)

[3 ТЕСТУВАННЯ ПРОГРАМИ І РЕЗУЛЬТАТИ ЇЇ ВИКОНАННЯ 27](#_Toc497455972)

[ВИСНОВКИ 32](#_Toc497455973)

[ПЕРЕЛІК ПОСИЛАНЬ 33](#_Toc497455974)

[Додаток А Лістинг файлу «main.cpp» 34](#_Toc497455975)

[Додаток Б Лістинг файлу «mainwindow.h» 35](#_Toc497455976)

[Додаток В Лістинг файлу «mainwindow.cpp» 37](#_Toc497455977)

[Додаток Г Лістинг файлу «add\_avtobus.h» 40](#_Toc497455978)

[Додаток Д Лістинг файлу «add\_avtobus.cpp» 41](#_Toc497455976)

[Додаток Е Лістинг файлу «add\_bilet.h» 43](#_Toc497455976)

[Додаток Ж Лістинг файлу «add\_bilet.cpp» 44](#_Toc497455977)

[Додаток И Лістинг файлу «add\_despetcher.h» 46](#_Toc497455978)

[Додаток К Лістинг файлу «add\_despetcher.cpp» 47](#_Toc497455975)

[Додаток Л Лістинг файлу «add\_marshrut.h» 49](#_Toc497455976)

[Додаток М Лістинг файлу «add\_marshrut.cpp» 50](#_Toc497455977)

[Додаток Н Лістинг файлу «database.h» 53](#_Toc497455978)

[Додаток П Лістинг файлу «database.cpp» 54](#_Toc497455975)

[Додаток Р Лістинг файлу «vodiy.h» 55](#_Toc497455976)

[Додаток С Лістинг файлу «vodiy.cpp» 56](#_Toc497455977)

[Додаток Т Лістинг файлу «kvitok.h» 58](#_Toc497455975)

[Додаток У Лістинг файлу «kvitok.cpp» 59](#_Toc497455976)

[Додаток Ф DVD-диск із програмним продуктом 60](#_Toc497455977)

ВСТУП

Протягом останніх декількох десятиліть комп'ютерні технології розвивались дуже швидко. Мови програмування також потерпіли значної еволюції. Поява більш потужних комп'ютерів дала життя більш об'ємним і складним програмам, які, водночас, висвітлювали нові проблеми в області керування програмами, а також їх подальшому супроводу.

Об'єктно-орієнтоване програмування сягає своїм корінням до створення мови програмування в 1960-тих роках, одночасно з посиленням дискусій про [кризу програмного забезпечення](https://uk.wikipedia.org/w/index.php?title=%D0%9A%D1%80%D0%B8%D0%B7%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F&action=edit&redlink=1). Разом із тим, як ускладнювалось апаратне та програмне забезпечення, було дуже важко зберегти якість програм. Об'єктно-орієнтоване програмування частково розв'язує цю проблему шляхом наголошення на модульності програми.

На відміну від традиційних поглядів, коли програму розглядали як набір [підпрограм](https://uk.wikipedia.org/wiki/%D0%9F%D1%96%D0%B4%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B0), або як перелік [інструкцій](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%86%D1%96%D1%8F) комп'ютеру, ООП програми можна вважати сукупністю об'єктів. Відповідно до парадигми об'єктно-орієнтованого програмування, кожен об'єкт здатний отримувати [повідомлення](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BC%D1%96%D0%BD_%D0%BF%D0%BE%D0%B2%D1%96%D0%B4%D0%BE%D0%BC%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F%D0%BC%D0%B8), обробляти дані, та надсилати повідомлення іншим об'єктам. Кожен об'єкт — своєрідний незалежний автомат з окремим призначенням та відповідальністю.

Сучасна економіка призвела до перегляду об’ємів витрат на комп’ютерну техніку та на спеціалізоване програмне забезпечення. Набагато краще і вигідніше стало розробляти прості програмні засоби, орієнтовані на розв’язання вузько спеціалізованих завдань. Тому, перед навчальними закладами постала мета підготувати спеціалістів, які спроможні створювати гнучкі програмні засоби, не витрачаючи на їх розробку та впровадження багато фінансів та часу.

Як приклад, в даній курсовій роботі представлено розробку спеціалізованого програмного рішення мовою С++ із використанням бази даних та принципів об’єктно-орієнтованого програмування. Ціль його розробки – автоматизація кас залізничного вокзалу, що значно полегшує їхню роботу.

Об'єктно-орієнтоване програмування - це парадигма програмування, заснована на поданні програми у вигляді сукупності взаємодіючих об'єктів, кожен з яких є екземпляром певного класу, а класи є членами певної ієрархії наслідування. Спочатку розробляється клас, а на його основі при виконанні програми створюються конкретні об'єкти (екземпляри класів). На основі класів можна створювати нові, які розширюють базовий клас і таким чином створюється ієрархія класів. Це дозволяє швидко конструювати достатньо функціональний програмний засіб.

# ТЕХНІЧНЕ ЗАВДАННЯ

## Найменування та область застосування

Найменування програми – «Програма автоматизації кас залізничного вокзалу».

Стисла назва – «CashdeskRS».

Область застосування програми – обслуговування пасажирів за допомогою автоматизованої програми для роботи з касами залізничного вокзалу.

## Підстави для розробки

Підставами для проведення даної розробки являється індивідуальне завдання на курсову роботу з дисципліни «Об’єктно-орієнтоване програмування».

Найменування теми курсової роботи – «Програма автоматизації кас залізничного вокзалу».

Замовник – керівник курсової роботи, викладач Слободян Руслан Олесійович.

Виконавець – студент групи КН-321 Сташків Віктор Ігорович.

## Призначення розробки

Підвищення ефективності і полегшення роботи кас залізничного вокзалу чи іншої організації, яка виконує таку ж діяльність, являється експлуатаційним призначенням даної розробки.

Відповідно функціональне призначення – спрощення і автоматизація роботи кас залізничного вокзалу.

## Вимоги до програми чи програмного виробу

### Вимоги до функціональних характеристик

Програма автоматизації кас залізничного вокзалу повинна забезпечити оптимізацію і спрощення ведення таких форм обліку:

−- Білети. Тут містяться місця пасажирів, їх вартість, час купівлі, П.І.П. пасажира і паспорт пасажира. Таким чином, це таблиця (див.табл. 1.1) з 5 стовпців і певної кількості рядків.

Таблиця 1.1 – Білети

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Місце | Вартість | Час купівлі | П.І.П. | Паспорт |
| 2 | 90 | 2018-12-20 11:21:00 | Іванов І.Н. | ВО 494888 |
| 4 | 220 | 2018-12-20 11:29:00 | Петров П.І. | ВС 203999 |
| 6 | 40 | 2018-12-20 12:07:00 | Сідоров С.А. | ВС 393939 |
| 8 | 120 | 2018-12-20 12:28:00 | Сидоренко С.М. | ВО 009888 |
| 10 | 290 | 2018-12-20 13:21:00 | Іващенко І.В. | ВО 838388 |
| 12 | 78 | 2018-12-20 13:34:00 | Юхименко С.В. | ВО 039399 |

- Маршрути. Тут містяться номера маршруту, їх пункт призначення, область, район, відстань, час відправлення і час прибуття. Таким чином, це таблиця (див.табл. 1.2) з 7 стовпців і певної кількості рядків.

Таблиця 1.2 – Маршрути

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Номер маршруту | Пункт призначення | Область | Район | Відстань | Час відправлення | Час прибуття |
| 1 | Львів | Львівська | Золоч. | 61 | 2018-12-20 16:15:00 | 2018-12-20 17:17:00 |
| 2 | Київ | Київська | Обол. | 369 | 2018-12-20 12:20:00 | 2018-12-20 18:20:00 |
| 3 | Луцьк | Волинська | Луцьк. | 42 | 2018-12-21 15:20:00 | 2018-12-21 17:10:00 |
| 4 | Івано-Франківськ | Івано-Франків. | Івано-Фран. | 257 | 2018-12-21 17:10:00 | 2018-12-21 22:00:00 |
| 5 | Воловець | Закарпатс. | Волов. | 234 | 2018-12-22 13:10:00 | 2018-12-22 15:10:00 |
| 6 | Стрий | Львівськ. | Стри. | 74 | 2018-12-22 15:10:00 | 2018-12-22 18:10:00 |

- Автобуси. Тут містяться моделі автобусів, їх номерні знаки і кількість мість в автобусі. Таким чином, це таблиця (див.табл. 1.3) з 3 стовпців і певної кількості рядків.

Таблиця 1.3 – Автобуси

|  |  |  |
| --- | --- | --- |
| Модель | Номерний знак | Кількість мість |
| БАЗ | ВС 1299 ВО | 29 |
| МБ-СПРИНТЕР | ВС 7777 ВВ | 18 |
| Богдан А301 | ВС 3938 ВА | 27 |
| МЕРСЕДЕС 18 | РЕ 2998 АО | 18 |
| БАЗ АО79 | ВС 3998 ВВ | 21 |
| ЗАЗ IVAN | ВС 7784 ВВ | 24 |

- Водії. Тут містяться П.І.П. водіїв, їх дата народження і паспорт водія. Таким чином, це таблиця (див.табл. 1.4) з 3 стовпців і певної кількості рядків.

Таблиця 1.4 – Водії

|  |  |  |
| --- | --- | --- |
| П.І.П. | Дата народження | Паспорт |
| Петренко В.І. | 1980-01-01 | ВО 000000 |
| Баліцький Т.Й. | 1980-02-02 | ВО 111111 |
| Бандас А.Н. | 1980-03-03 | ВО 222222 |
| Черній Г.М. | 1980-04-04 | ВО 333333 |
| Терекуш В.Є. | 1980-05-05 | ВО 444444 |
| Вернецький Н.П. | 1980-06-06 | ВО 555555 |

- Диспетчери. Тут містяться П.І.П. диспетчерів, їх дата народження і адреса проживання диспетчера. Таким чином, це таблиця (див.табл. 1.5) з 3 стовпців і певної кількості рядків.

Таблиця 1.5 – Диспетчери

|  |  |  |
| --- | --- | --- |
| П.І.П. | Дата народження | Адреса |
| Петрова І.В. | 1990-01-01 | м.Тернопіль |
| Соколова С.С. | 1990-02-02 | м.Львів |
| Микитенко В.В. | 1990-03-03 | м.Монастириськ |
| Кашуба І.В. | 1990-04-04 | м.Стрий |
| Богородець А.В. | 1990-05-05 | м.Київ |

### Вимоги до надійності

Надійне функціонування програми повинне забезпечуватись наявністю блоку перевірки правильності введених даних. Це збереже від помилок, що виникають при виконанні рутинної роботи користувачем.

Також необхідно забезпечити достатню стійкість програми до некваліфікованих дій користувача, що досягається супроводом будь-яких операцій в програмі чіткими і зрозумілими повідомленнями.

### Умови до експлуатації

Розроблений програмний продукт не потребуватиме яких-небудь видів обслуговування, для його використання достатньо буде навиків користувача ПК.

### Вимоги до функціональних характеристик

В склад технічних засобів повинні входити: персональний комп’ютер із такими мінімальними характеристиками:

− Процесор Pentium III;

− ОЗУ: 128Mb, або вище;

− обсяг дискової пам’яті – 80 ГБ.

− монітор;

− клавіатура;

− мишка.

### Вимоги до інформаційної і програмної сумісності

Вихідні коди програми повинні бути реалізовані мовою С++. В якості середовища розробки програми повинне використовуватись середовище Qt Creator 4.8.0.

Системні програмні засоби і утиліти, які буде використовувати програма, повинні забезпечуватись операційною системою сімейства Windows версії не нижче Windows 7, Linux або MacOS оскільки середовище Qt є кросс-платформне.

Вимоги до захисту інформації і програми не пред’являються.

## Вимоги до програмної документації

По закінченню розробки програмного забезпечення потрібно підготувати таку документацію:

- інструкція інсталяції програми;

- загальні відомості про можливості програми;

- інструкція з експлуатації.

## Техніко-економічні показники

Розрахунок економічної ефективності і вартості розробки програмного продукту не проводиться.

Приблизне число використань розробленої програми в рік – 2500-4500 раз.

## Стадії та етапи розробки

В загальному випадку, життєвий цикл визначається моделлю й описується у формі методології (методу). Модель або парадигма життєвого циклу визначає загальну організацію і, як правило, основні його фази та принципи переходу між ними. Методологія (метод) визначає комплекс робіт, їх детальний зміст і рольову відповідальність спеціалістів на всіх етапах вибраної моделі.

Життєвий цикл програмного забезпечення супроводжується розробленням, обігом та використанням програмної документації.

Програмна документація — сукупність документів, що містять відомості, необхідні для розробки, виготовлення, супроводу та експлуатації програм. Програмна документація є одним з видів технічної документації.

Комплекс державних стандартів, що встановлюють взаємопов'язані правила розробки, оформлення та обігу програм і програмної документації називається «Єдина система програмної документації» .

При розробці даного програмного забезпечення буде використовуватись спіральна модель життєвого циклу.

Розробка ітераціями відображає об'єктивно існуючий спіральний цикл створення системи. Неповне завершення робіт на кожному етапі дозволяє переходити на наступний етап, не чекаючи повного завершення роботи на поточному. При ітеративному способі розробки відсутню роботу можна буде виконати на наступній ітерації. Головне ж завдання - щонайшвидше показати користувачам системи працездатний продукт, тим самим активізуючи процес уточнення і доповнення вимог.

Виходячи з можливості внесення змін, як в процес, так і в проміжний продукт було створено спіральну модель ЖЦ .

Внесення змін орієнтоване на задоволення потреби користувачів одразу, як тільки буде встановлено, що створені артефакти або елементи документації не відповідають дійсному стану розробки.

Дана модель ЖЦ допускає аналіз продукту на витку розробки, його перевірку, оцінку правильності та прийняття рішення про перехід на наступний виток або повернення на попередній виток для доопрацювання на ньому проміжного продукту.

Відмінність цієї моделі від каскадної полягає в можливості багато разів повертатися до процесу формулювання вимог і до повторної розробки версії системи з будь-якого процесу моделі.

Для програмного продукту така модель не дуже підходить з декількох причин. По-перше, висловлення вимог замовником носить суб'єктивний характер, вимоги можуть багаторазово уточнюватися протягом розробки ПС і навіть після завершення та випробовування, і часом може з'ясуватися, що замовник «хотів зовсім інше». По-друге, змінюються обставини та умови використання системи, тому загальновизнаним законом програмної інженерії є закон еволюції, який сформулюємо так: кожна діюча ПС з часом потребує внесення змін або виводиться з експлуатації.

При необхідності внесення змін до системи на кожному витку з метою отримання нової версії системи обов'язково вносяться зміни в заздалегідь зафіксовані вимоги, після чого повертаються на попередній виток спіралі для продовження реалізації нової версії системи з урахуванням усіх змін.

## Порядок контролю та прийому

Прийом розробленого програмного забезпечення повинен відбуватися на об’єкті Замовника в терміни, які зазначені в індивідуальному завданні.

Для прийому роботи Виконавець повинен представити:

− діючу програму, яка повністю відповідає даному технічному завданню;

− вихідний програмний код, записаний разом із програмою на оптичний носій інформації.

Прийом програмного забезпечення повинен відбуватися перед комісією з двох чоловік (один з яких – Замовник) у такій послідовності:

− доповідь Виконавця про виконану роботу;

− демонстрація Виконавцем роботи програми;

− контрольні випробовування роботи програми;

− відповіді на запитання і зауваження комісії.

# РОЗРОБКА ТЕХНІЧНОГО ТА РОБОЧОГО ПРОЕКТУ

## Розробка загальної структури і варіантів використання програми

Під час проходження етапу аналізу спочатку було визначено такі питання:

* Хто буде діючими суб'єктами?
* Хто вводитиме інформацію?
* Хто запрошуватиме?
* Чи буде хто-небудь ще взаємодіяти з програмою?
* Чи буде сама програма взаємодіяти з іншими програмами?

В даному випадку з програмою CashdeskRS має працювати тільки один користувач – диспетчер залізничного вокзалу. Таким чином, один і той самий користувач вводить інформацію, і переглядає її у різних видах.

Наступне, що потрібно було з'ясувати, це групу дій, які ініціюватиме діючий суб'єкт. Виходячи із технічного завдання, діючий суб'єкт – диспетчер буде виконувати такі дії:

– почати роботу з програмою;

– білети;

– маршрути;

– автобуси;

– водії;

– диспетчери;

– оформити квиток;

– видалити квиток;

– перегляд бази.

Таким чином, діаграму варіантів використання, яка отримується внаслідок наведеного переліку дій, можна подати так, як це показано на рисунку 2.1.

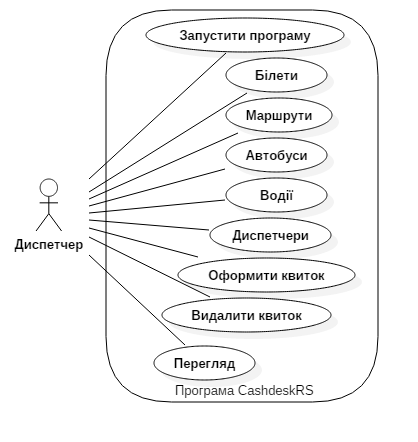


Рисунок 2.1 – Діаграма варіантів використання для програми CashdeskRS

**Запустити програму**. Ця дія, здавалося б, є дуже очевидною для того, щоб про неї зовсім не згадувати, але все ж таки... Коли запускається програма, на екран повинно виводитися меню, з якого користувач може вибрати потрібну дію. Це може називатися екраном інтерфейсу користувача.

**Білети**. На екрані має відобразитися форма, у якій програма добавляє, обновляє або видаляє дані пов’язані із білетами та 3 кнопки:

* Добавити;
* Видалити рядок із таким ID;
* Обновити дані із таким ID.

**Маршрути**. На екрані має відобразитися форма, у якій програма добавляє, обновляє або видаляє дані пов’язані із маршрутами та 3 кнопки:

* Добавити;
* Видалити рядок із таким ID;
* Обновити дані із таким ID.

**Автобуси**. На екрані має відобразитися форма, у якій програма добавляє, обновляє або видаляє дані пов’язані із автобусами та 3 кнопки:

* Добавити;
* Видалити рядок із таким ID;
* Обновити дані із таким ID.

**Водії**. На екрані має відобразитися форма, у якій програма добавляє, обновляє або видаляє дані пов’язані із водіями та 3 кнопки:

* Добавити;
* Видалити рядок із таким ID;
* Обновити дані із таким ID.

**Диспетчери**. На екрані має відобразитися форма, у якій програма добавляє, обновляє або видаляє дані пов’язані із диспетчерами та 3 кнопки:

* Добавити;
* Видалити рядок із таким ID;
* Обновити дані із таким ID.

**Оформити квиток.** На екрані має відобразитися форма, у якій програма просить користувача ввести: Номер перевезення, ID Маршруту, ID Білета, ID Водія, ID Диспетчера. Ця інформація повинна заноситися в таблицю.

**Видалити квиток**. Видаляє із бази квиток за вказаним номером.

**Перегляд**. На екрані мають відобразитися таблиці з відповідною інформацією.

## Розробка системи класів

Наступний етап розробки програмного забезпечення – етап проектування. Даний етап поділяється на логічне та фізичне проектування. Логічне полягає у розробці структури класів, коли визначаються поля для збереження складових об'єктів, алгоритми методів, що описують поведінку об'єктів. Насамперед необхідно передбачити класи, з яких буде складатися програма. Для цього спочатку проаналізуємо перелік іменників з опису варіантів використання, потім внесемо деякі їх уточнення, визначимо їх атрибути, перейдемо від дієслів до повідомлень, і, на завершення, розробимо діаграми класів і послідовностей.

**Аналіз переліку іменників з опису варіантів використання**. Розглянемо перелік усіх іменників, які візьмемо з опису варіантів використання:

1. База даних.
2. Головне вікно.
3. Вікно оформлення білету.
4. Вікно білети.
5. Вікно маршрути.
6. Вікно автобуси.
7. Вікно водії.
8. Вікно диспетчери.

**Уточнення переліку іменників.** З різних причин багато іменників не зможуть стати класами. Давайте проведемо відбір тільки тих іменників, які можуть претендувати на те, щоб бути класом.

Одним із таких претендентів може бути клас “База даних”, який буде містити підключення до сервера на якому буде зберігатись вся інформація в таблицях: таблиця білети, таблиця маршрути, таблиця автобуси, таблиця водії, таблиця диспетчери, таблиця перевезення. Наступні 6 класів це класи вікон, які містять інформацію про усі функції вікон і їх можливості. Отже, складемо перелік класів, які було тільки, що уточнено:

1. База даних.
2. Головне вікно.
3. Вікно оформлення білету.
4. Вікно білети.
5. Вікно маршрути.
6. Вікно автобуси.
7. Вікно водії.
8. Вікно диспетчери.

**Визначення** **атрибутів**. Багато іменників, яким відмовлено в реєстрації як кандидатами у класи, будуть потенційними кандидатами в атрибути (компонентні дані) класів.

**Перехід від дієслів до повідомлень.** Тепер подивимося, що нам дають варіанти використання для з'ясування того, якими повідомленнями обмінюватимуться класи. Оскільки повідомлення – це, по суті справи, є викликом методу в об'єкті, то визначення повідомлень зводиться до визначення методів класу, що приймає те або інше повідомлення. Як і у випадку з іменниками, далеко не кожне дієслово стає кандидатом у повідомлення.

## 2.3. Розробка методів

Етап починається із створення основного інтерфейсу проекту, бібліотеки класів, описуються класи, розробляються методи, уточнюються вимоги до програми та вносяться певні зміни. Одержаний прототип продукту тестується й налагоджується. В програмі використовуються класи та методи Qt Framework[3], які суттєво полегшили роботу з організацією роботи усіх складових програми.

Як уже було зазначено вище, багато атрибутів для кожного з класів є похідними з тих іменників, які самі не стали класами. Інші атрибути можуть бути виведені з асоціацій в діаграмі класів.

**Основні методи.** Такими методами є методи роботи з базою даних, інформацією яку хоче отримати користувач. Наприклад метод connect створює з’єднання з базою даних через яке інший метод такий як get може працювати з базою даних беручи з відти інформацію, яку потрібно побачити та додати іншу інформацію.

Визначивши усі необхідні атрибути і методи класів, а також допоміжні класи, можна з допомогою програми StarUML побудувати повну діаграму класів. Розроблена діаграма класів зображена на рисунку 2.2.

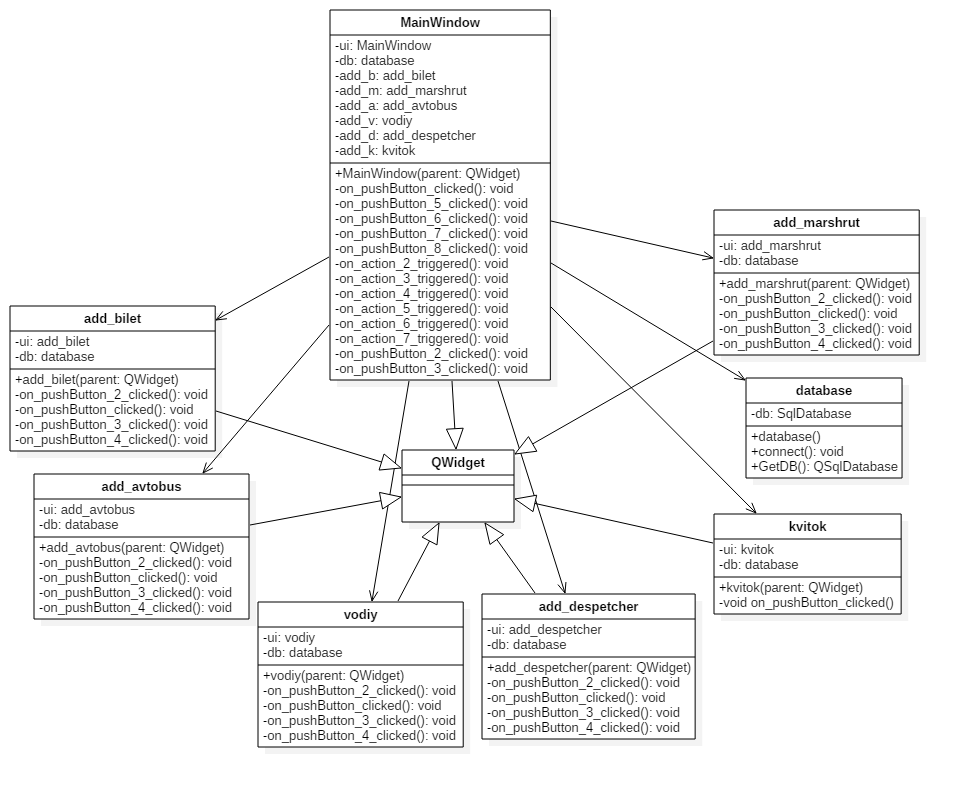


Рисунок 2.2 – UML-діаграма класів програми CashdeskRS

Маючи діаграми варіантів використання, детальні їх описи, діаграми класів, а також попередні плани щодо створення коду програми, можна приступити до написання коду програми.

Написання коду програми починається із написання заголовного (\*.h) файлу у якому необхідно визначити тільки інтерфейсні частини класів, але не подробиці їх реалізації. Оголошення в заголовному файлі – це загальнодоступна частина класів. Тіла функцій, які є розташовані в \*.срр-файлах, називаються реалізацією і користувачам (стороннім програмістам) недоступні.

Написання заголовного файлу – це проміжний крок між етапами плануванням і звичайним кодуванням методів.

Більшість оголошень появляються безпосередньо з класів, створених за допомогою взятих з описів варіантів використання іменників, і відображаються на діаграмі класів.

Написання коду програми розглянутих методів має починатися тільки на даному етапі розроблення ПЗ і ні кроком раніше, тому що тільки зараз відомі ім'я кожного класу чи методу, їх призначення і навіть, можливо, передбачити аргументи, що їм передаються.

## Проектування і опис інтерфейсу користувача

Інтерфейс користувача (ІК) - це сукупність засобів, за допомогою яких користувач взаємодіє з різними пристроями (з комп’ютером або побутовою технікою) або іншим складним інструментарієм (системою). Інтерфейс користувача - це такий різновид інтерфейсів, в якому з одного боку - людина, з іншого - машина (пристрій, програмне забезпечення). За визначенням Національного банку стандартизованих науково-технічних термінів, інтерфейс користувача - це комплекс апаратних і програмних засобів, що забезпечує взаємодію користувача з комп'ютером .

ІК часто розуміють лише як зовнішній вигляд програмного забезпечення (ПЗ), але таке розуміння є надто вузьким, оскільки саме за допомогою інтерфейсу користувач сприймає програму в цілому та використовує її функціональність. ІК забезпечує підтримку прийняття рішень у визначеній предметній галузі та визначає порядок використання ПЗ і документації до нього. В дійсності, ІК об'єднує усі елементи і компоненти ПЗ, які здатні впливати на взаємодію користувача з програмним забезпеченням . До таких елементів належать: набір задач, які користувач розв’язує за допомогою ПЗ; використовувана програмним забезпеченням метафора (наприклад, "робочий стіл" у операційній системі Windows); елементи управління ПЗ; навігація між блоками ПЗ; візуальний (і не тільки) дизайн вікон та екранних форм програми та інші складові .

При створенні ПЗ використовуватиметься середовище Qt Designer[2]. Основні класи, які використовуватимуться для роботи з вікнами: QMainWindow, QWidget, QMessagebox,QSqlTableModel.

Також при оформлені та створенні вікна будуть використовуватись такі елементи Qt Designer:

* LineEdit – поле для вводу;
* Label – поле для показу тексту;
* TableView – інструмент для виводу таблиці на екран;
* PushButton – кнопки;
* DateEdit – для спрощення вводу дати;
* DateTimeEdit – для спрощення вводу дати і часу;
* Layot – інтрумент для створення шарів на формі (для полегшення роботи з елементами форми);
* GroupBox – впорядковує елементи по групам.

Переглянути як ці елементи працюють можна в 3 розділі даної записки.

При оформлені та створенні вікна “Білети” використовуватись такі елементи Qt Designer: LineEdit, Layot, GroupBox, Label, PushButton, LineEdit, DateTimeEdit, які зображені на рисунку 2.3.

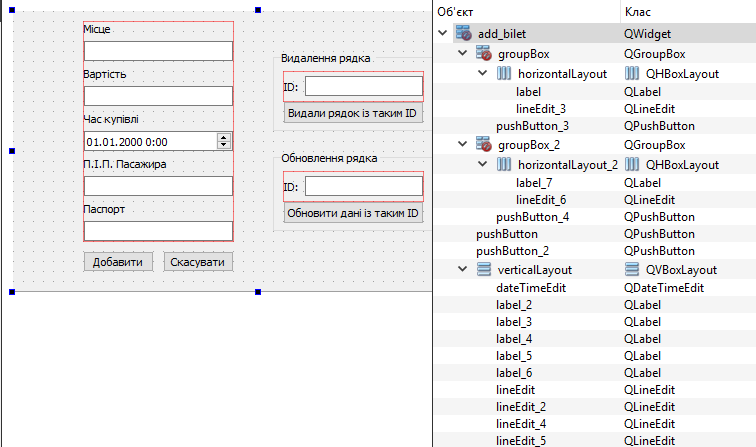


Рисунок 2.3 – Вікно “Білети”

При оформлені та створенні вікна “Маршрути” використовуватись такі елементи Qt Designer: LineEdit, Layot, GroupBox, Label, PushButton, LineEdit, DateTimeEdit, які зображені на рисунку 2.4.

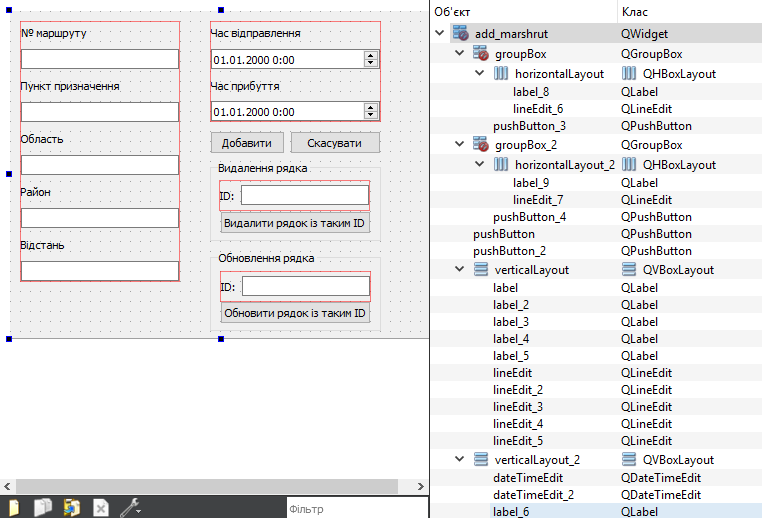


Рисунок 2.4 – Вікно “Маршрути”

При оформлені та створенні вікна “Автобуси” використовуватись такі елементи Qt Designer: LineEdit, Layot, GroupBox, Label, PushButton, LineEdit, які зображені на рисунку 2.5.

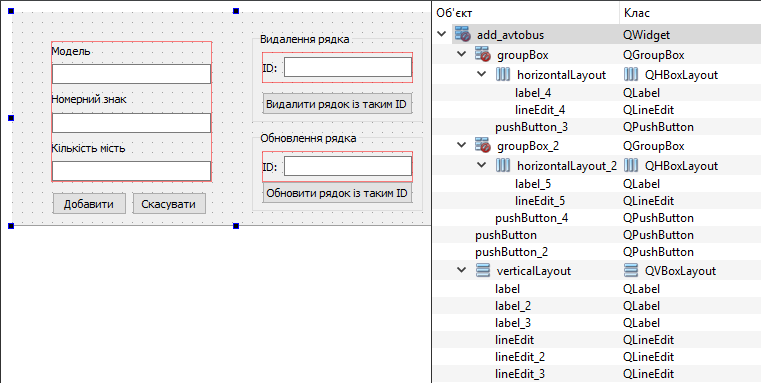


Рисунок 2.5 – Вікно “Автобуси”

При оформлені та створенні вікна “Водії” використовуватись такі елементи Qt Designer: LineEdit, Layot, GroupBox, Label, PushButton, LineEdit, DateEdit, які зображені на рисунку 2.6.

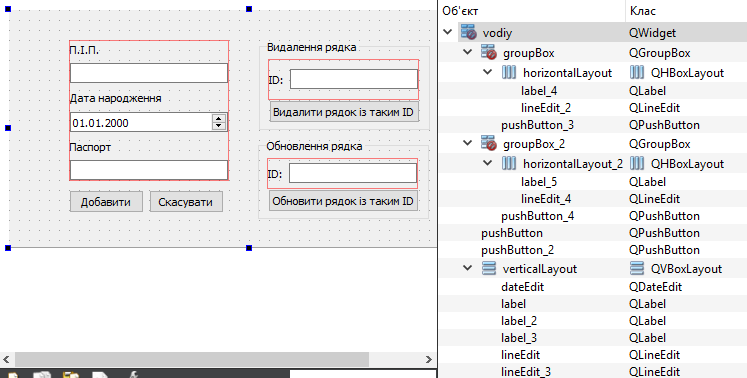


Рисунок 2.6 – Вікно “Водії”

При оформлені та створенні вікна “Диспетчери” використовуватись такі елементи Qt Designer: LineEdit, Layot, GroupBox, Label, PushButton, LineEdit, DateEdit, які зображені на рисунку 2.7.

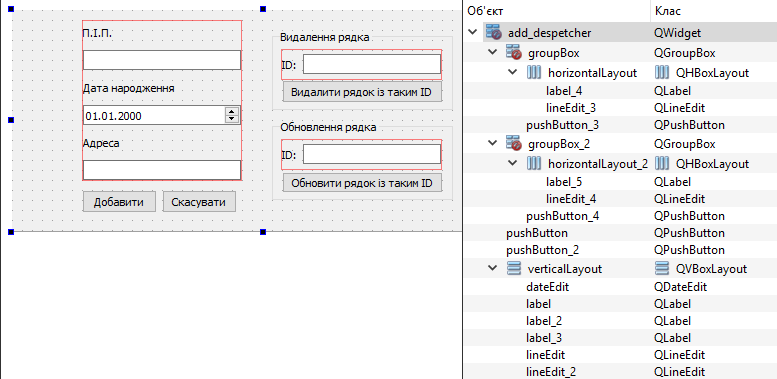


Рисунок 2.7 – Вікно “Диспетчери”

При оформлені та створенні вікна “Оформити квиток” використовуватись такі елементи Qt Designer: LineEdit, Layot, GroupBox, Label, PushButton, LineEdit, які зображені на рисунку 2.8.

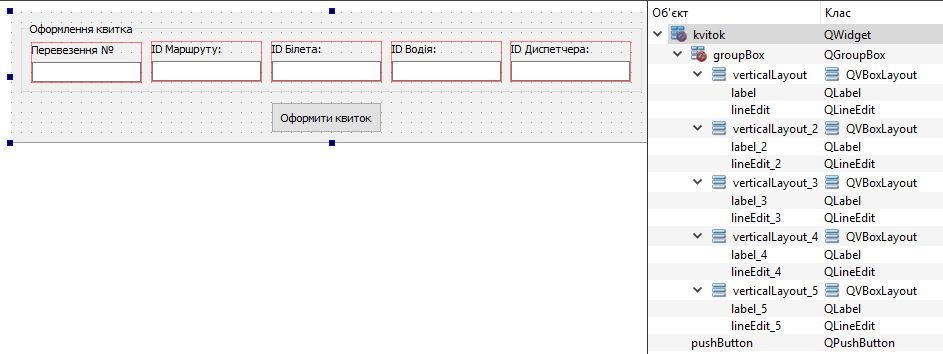


Рисунок 2.8 – Вікно “Оформити квиток”

## Опис файлової структури програми

Програма складається із таких файлів:

* «main.cpp» -лістинг даного файла знаходиться в додатку А. В цьому файлі міститься головна функція програми ;
* «mainwindow.h» - лістинг даного файла знаходиться в додатку Б;
* «mainwindow.cpp» - лістинг даного файла знаходиться в додатку В;
* «add\_avtobus.h» - лістинг даного файла знаходиться в додатку Г;
* «add\_avtobus.cpp» - лістинг даного файла знаходиться в додатку Д;
* «add\_bilet.h» - лістинг даного файла знаходиться в додатку Е;
* «add\_bilet.cpp» - лістинг даного файла знаходиться в додатку Ж;
* «add\_despetcher.h» - лістинг даного файла знаходиться в додатку И;
* «add\_despetcher.cpp» - лістинг даного файла знаходиться в додатку К;
* «add\_marshrut.h» - лістинг даного файла знаходиться в додатку Л;
* «add\_marshrut.cpp» - лістинг даного файла знаходиться в додатку М:
* «database.h» - лістинг даного файла знаходиться в додатку Н;
* «database.cpp» - лістинг даного файла знаходиться в додатку П;
* «vodiy.h» - лістинг даного файла знаходиться в додатку Р;
* «vodiy.cpp» - лістинг даного файла знаходиться в додатку С;
* «kvitok.h» - лістинг даного файла знаходиться в додатку Т;
* «kvitok.cpp» - лістинг даного файла знаходиться в додатку У;
* DVD-диск із програмним продуктом знаходиться в додатку Ф.

База даних складається із шести таблиць, які зображено на ER-діаграмі рисунку 2.9. В якості системи управління бази даних було використано SQLite — полегшена [реляційна система керування базами даних](https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BA%D0%B5%D1%80%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85).

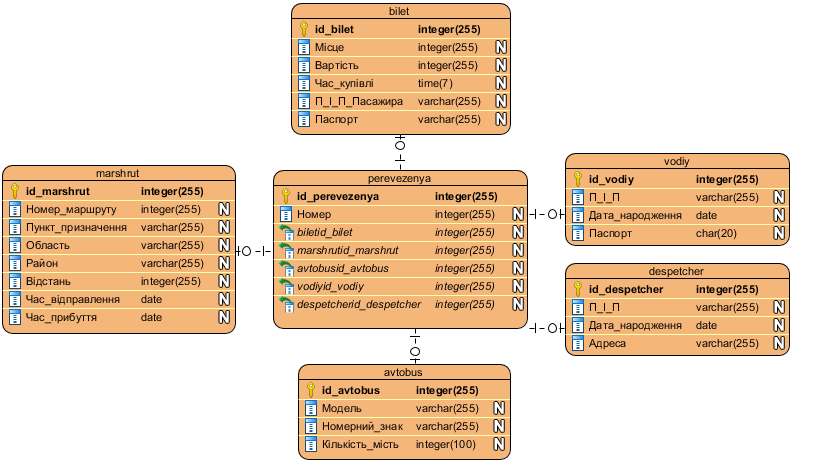


Рисунок 2.8 – ER-діаграма бази даних

# ТЕСТУВАННЯ ПРОГРАМИ І РЕЗУЛЬТАТИ ЇЇ ВИКОНАННЯ

В ході тестування програми було виявлено і виправлено незначні помилки. Запустивши програму на екрані монітора відображається головне вікно представлене на рисунку 3.1.

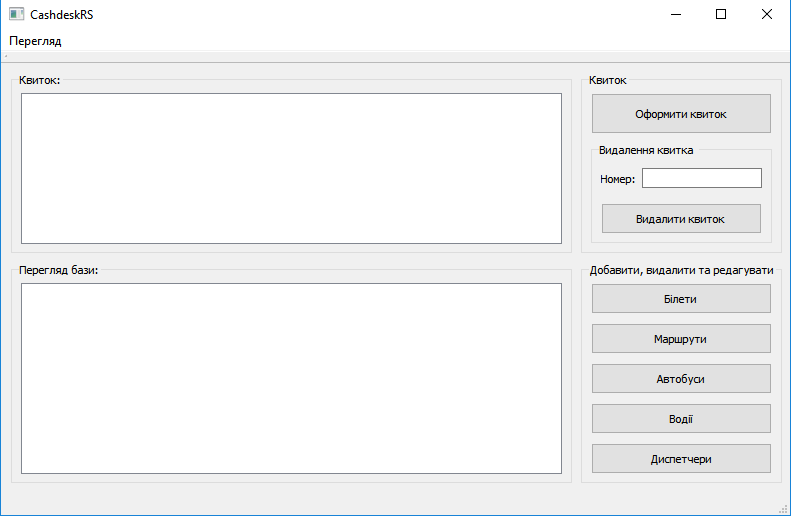


Рисунок 3.1 – Головне вікно програми “CashdeskRS”

В головному меню ми бачимо 7 основних кнопок і одну позицію:

− Оформити квиток;

− Видалити квиток;

− Білети;

− Маршрути;

− Автобуси;

− Водії;

− Диспетчери;

− Перегляд.

Нажавши на кнопку «оформити квиток» ми зможемо перейти в вікно, яке зображене на рисунку 3.2.

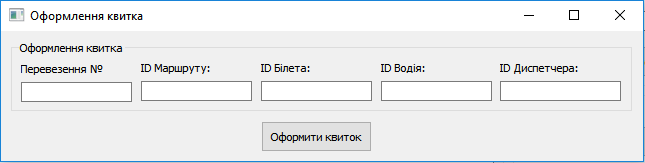


Рисунок 3.2 – Оформлення квитка

Нажавши на кнопку «видалити квиток» ми зможемо видалити квиток по вказаному “номеру квитка”.

Нажавши на кнопку «білети» ми зможемо перейти в вікно, яке зображене на рисунку 3.3.

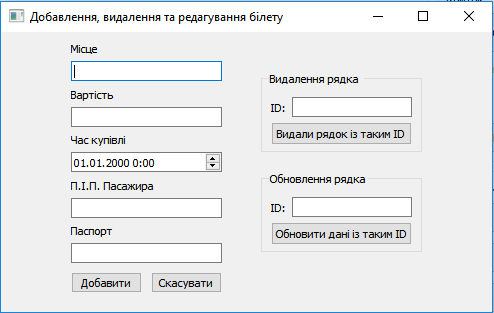


Рисунок 3.3 – Добавлення, видалення та редагування квитка

Нажавши на кнопку «маршрути» ми зможемо перейти в вікно, яке зображене на рисунку 3.4.

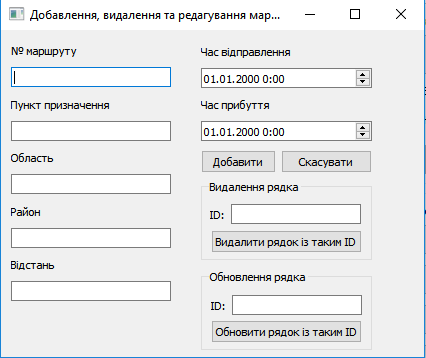


Рисунок 3.4 – Добавлення, видалення та редагування маршруту

Нажавши на кнопку «автобуси» ми зможемо перейти в вікно, яке зображене на рисунку 3.5.

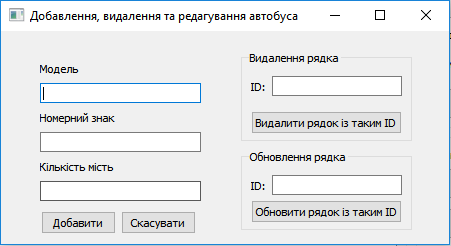


Рисунок 3.5 – Добавлення, видалення та редагування автобуса

Нажавши на кнопку «водії» ми зможемо перейти в вікно, яке зображене на рисунку 3.6.

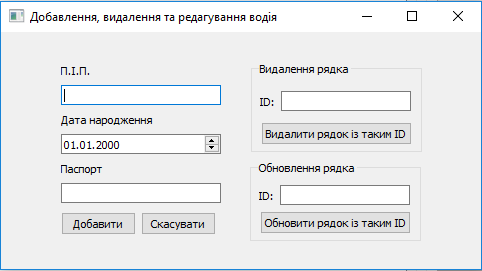


Рисунок 3.6 – Добавлення, видалення та редагування водія

Нажавши на кнопку «диспетчери» ми зможемо перейти в вікно, яке зображене на рисунку 3.7.

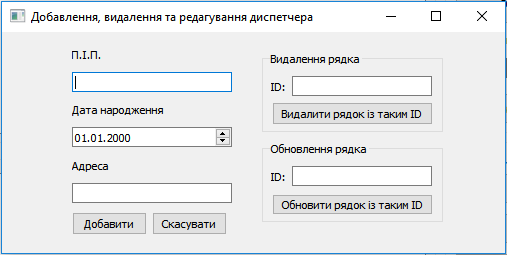


Рисунок 3.7 – Добавлення, видалення та редагування диспетчера

Нажавши на позицію «перегляд» ми зможемо переглядати інформацію про білети, маршрути, автобуси, водіїв, диспетчерів, квитки, яка зображена на рисунку 3.8.

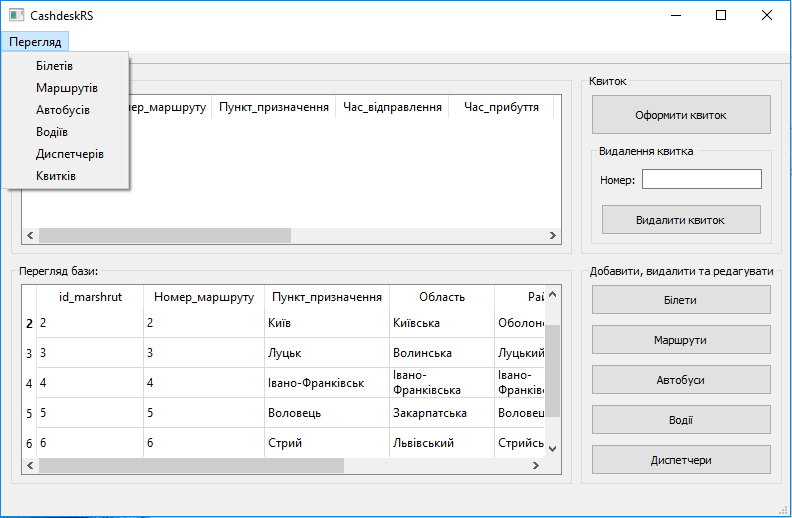


Рисунок 3.8 – Перегляд

ВИСНОВКИ

В даній курсовій роботі було описано етапи розробки програми CashdeskRS, яка призначена для автоматизації кас залізничного вокзалу. Розроблена програма відповідає усім поставленим вимогам технічного завдання. Під час виконання курсової роботи було закріплено і поглиблено знання, одержанні за час вивчення предмету «Об’єктно-орієнтоване програмування», більш детально вивчено мову програмування С++. Для розробки даного програмного забезпечення використовувались принципи і концепції об’єктно-орієнтованого програмування. Використане середовище програмування Qt , його особливі можливості і переваги, що дозволило швидко конструювати програмний засіб. Також під час розробки використано стандартні бібліотеки Qt, що дозволило в значній мірі пришвидшити розробку даної програми.

ПЕРЕЛІК ПОСИЛАНЬ

1. Грицюк Ю.І. Об’єктно-орієнтоване програмування мовою С++ [Текст]: навчальний посібник / Ю.І. Грицюк, Т.Є. Рак. – Львів: Вид-во Львівського ДУ БЖД, 2011. – 404 с.
2. Вайсфельд М. Объектно-ориентированное мышление [Текст] / М. Вайсфельд. – СПб.: Питер, 2014. — 304 с.: ил. — ISBN 978-5-496-00793-1.
3. Build with Qt [Електронний ресурс]. – Електорон. дан. – Режим доступу: https://www1.qt.io/built-with-qt/ – Дата доступу: 20.11.2018.

Додаток А  
Лістинг файлу «main.cpp»

#include "mainwindow.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}

Додаток Б  
Лістинг файлу «mainwindow.h»

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include "add\_bilet.h"

#include "add\_marshrut.h"

#include "add\_avtobus.h"

#include "vodiy.h"

#include "add\_despetcher.h"

#include "kvitok.h"

#include "database.h"

#include <QString>

namespace **Ui** {

class **MainWindow**;

}

class **MainWindow** : public QMainWindow

{

Q\_OBJECT

public:

explicit **MainWindow**(QWidget \*parent = nullptr);

~***MainWindow***();

private slots:

void **on\_pushButton\_clicked**();

void **on\_pushButton\_5\_clicked**();

void **on\_pushButton\_6\_clicked**();

void **on\_pushButton\_7\_clicked**();

void **on\_pushButton\_8\_clicked**();

void **on\_action\_2\_triggered**();

void **on\_action\_3\_triggered**();

void **on\_action\_4\_triggered**();

void **on\_action\_5\_triggered**();

void **on\_action\_6\_triggered**();

void **on\_action\_7\_triggered**();

void **on\_pushButton\_2\_clicked**();

void **on\_pushButton\_3\_clicked**();

private:

Ui::MainWindow \*ui;

database db;

add\_bilet \*add\_b;

add\_marshrut \*add\_m;

add\_avtobus \*add\_a;

vodiy \*add\_v;

add\_despetcher \*add\_d;

kvitok \*add\_k;

};

#endif // MAINWINDOW\_H

Додаток В  
Лістинг файлу «mainwindow.cpp»

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

db.connect();

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::on\_pushButton\_clicked()

{

add\_b= new add\_bilet();

add\_b->show();

}

void MainWindow::on\_pushButton\_5\_clicked()

{

add\_m= new add\_marshrut();

add\_m->show();

}

void MainWindow::on\_pushButton\_6\_clicked()

{

add\_a= new add\_avtobus();

add\_a->show();

}

void MainWindow::on\_pushButton\_7\_clicked()

{

add\_v= new vodiy();

add\_v->show();

}

void MainWindow::on\_pushButton\_8\_clicked()

{

add\_d= new add\_despetcher();

add\_d->show();

}

void MainWindow::on\_action\_2\_triggered()

{

QSqlQuery query = QSqlQuery(db.getDB());

QSqlQueryModel \*model1= new QSqlQueryModel();

query.prepare("select \* from bilet");

query.exec();

model1->setQuery(query);

ui->tableView\_2->setModel(model1);

}

void MainWindow::on\_action\_3\_triggered()

{

QSqlQuery query = QSqlQuery(db.getDB());

QSqlQueryModel \*model2= new QSqlQueryModel();

query.prepare("select \* from marshrut");

query.exec();

model2->setQuery(query);

ui->tableView\_2->setModel(model2);

}

void MainWindow::on\_action\_4\_triggered()

{

QSqlQuery query = QSqlQuery(db.getDB());

QSqlQueryModel \*model3= new QSqlQueryModel();

query.prepare("select \* from avtobus");

query.exec();

model3->setQuery(query);

ui->tableView\_2->setModel(model3);

}

void MainWindow::on\_action\_5\_triggered()

{

QSqlQuery query = QSqlQuery(db.getDB());

QSqlQueryModel \*model3= new QSqlQueryModel();

query.prepare("select \* from vodiy");

query.exec();

model3->setQuery(query);

ui->tableView\_2->setModel(model3);

}

void MainWindow::on\_action\_6\_triggered()

{

QSqlQuery query = QSqlQuery(db.getDB());

QSqlQueryModel \*model4= new QSqlQueryModel();

query.prepare("select \* from despetcher");

query.exec();

model4->setQuery(query);

ui->tableView\_2->setModel(model4);

}

void MainWindow::on\_action\_7\_triggered()

{

QSqlQuery query = QSqlQuery(db.getDB());

QSqlQueryModel \*model5= new QSqlQueryModel();

query.prepare("select perevezenya.Номер, marshrut.Номер\_маршруту, marshrut.Пункт\_призначення, marshrut.Час\_відправлення, marshrut.Час\_прибуття, bilet.Місце, bilet.П\_І\_П\_Пасажира, bilet.Вартість, vodiy.П\_І\_П, despetcher.id\_despetcher from perevezenya, marshrut, bilet, vodiy, despetcher where (perevezenya.id\_marshrut=marshrut.id\_marshrut) and (perevezenya.id\_bilet=bilet.id\_bilet) and (perevezenya.id\_vodiy=vodiy.id\_vodiy) and (perevezenya.id\_despetcher=despetcher.id\_despetcher) ");

query.exec();

model5->setQuery(query);

ui->tableView->setModel(model5);

}

void MainWindow::on\_pushButton\_2\_clicked()

{

add\_k= new kvitok();

add\_k->show();

}

void MainWindow::on\_pushButton\_3\_clicked()

{

QSqlQuery query = QSqlQuery(db.getDB());

query.prepare("delete from perevezenya "

"where Номер=(?)");

QString id\_perevezenya = ui->lineEdit->text();

query.addBindValue(id\_perevezenya);

if(!query.exec())

{

qDebug() << query.lastError();

QMessageBox::critical(this,"Помилка!","Введені дані неправильні!");

} else {QMessageBox::information(this,"Успіх!","Квиток видалено!");}

}

Додаток Г  
Лістинг файлу «add\_avtobus.h»

#ifndef ADD\_AVTOBUS\_H

#define ADD\_AVTOBUS\_H

#include <QWidget>

#include <QString>

#include <QMainWindow>

#include "database.h"

#include <QMessageBox>

namespace Ui {

class add\_avtobus;

}

class add\_avtobus : public QWidget

{

Q\_OBJECT

public:

explicit add\_avtobus(QWidget \*parent = nullptr);

~add\_avtobus();

private slots:

void on\_pushButton\_2\_clicked();

void on\_pushButton\_clicked();

void on\_pushButton\_3\_clicked();

void on\_pushButton\_4\_clicked();

private:

Ui::add\_avtobus \*ui;

database db;

};

#endif // ADD\_AVTOBUS\_H

Додаток Д  
Лістинг файлу «add\_avtobus.cpp»

#include "add\_avtobus.h"

#include "ui\_add\_avtobus.h"

add\_avtobus::add\_avtobus(QWidget \*parent) :

QWidget(parent),

ui(new Ui::add\_avtobus)

{

ui->setupUi(this);

}

add\_avtobus::~add\_avtobus()

{

delete ui;

}

void add\_avtobus::on\_pushButton\_2\_clicked()

{

close();

}

void add\_avtobus::on\_pushButton\_clicked()

{

QSqlQuery query = QSqlQuery(db.getDB());

query.prepare("INSERT INTO avtobus (Модель,Номерний\_знак,Кількість\_мість)"

"VALUES (?, ?, ?)");

QString model = ui->lineEdit->text();

QString nomer\_znak = ui->lineEdit\_2->text();

QString kilkist\_mist = ui->lineEdit\_3->text();

query.addBindValue(model);

query.addBindValue(nomer\_znak);

query.addBindValue(kilkist\_mist);

if(!query.exec())

{

qDebug() << query.lastError();

QMessageBox::critical(this,"Помилка!","Дані введено невірно!");

} else {QMessageBox::information(this,"Успіх!","Дані успішно добавлені!"); close();}

}

void add\_avtobus::on\_pushButton\_3\_clicked()

{

QSqlQuery query = QSqlQuery(db.getDB());

query.prepare("delete from avtobus "

"where id\_avtobus=(?)");

QString id\_avtobus = ui->lineEdit\_4->text();

query.addBindValue(id\_avtobus);

if(!query.exec())

{

qDebug() << query.lastError();

QMessageBox::critical(this,"Помилка!","Введені дані неправильні!");

} else {QMessageBox::information(this,"Успіх!","Рядок видалено!");}

}

void add\_avtobus::on\_pushButton\_4\_clicked()

{

QSqlQuery query = QSqlQuery(db.getDB());

query.prepare("update avtobus set Модель=(?), Номерний\_знак=(?), Кількість\_мість=(?)"

"where id\_avtobus=(?)");

QString model = ui->lineEdit->text();

QString nomer\_znak = ui->lineEdit\_2->text();

QString kilkist\_mist = ui->lineEdit\_3->text();

QString id\_avtobus = ui->lineEdit\_5->text();

query.addBindValue(model);

query.addBindValue(nomer\_znak);

query.addBindValue(kilkist\_mist);

query.addBindValue(id\_avtobus);

if(!query.exec())

{

qDebug() << query.lastError();

QMessageBox::critical(this,"Помилка!","Введені дані неправильні!");

} else {QMessageBox::information(this,"Успіх!","Рядок із заданим ID обновлений!");}

}

Додаток Е  
Лістинг файлу «add\_bilet.h»

#ifndef ADD\_BILET\_H

#define ADD\_BILET\_H

#include <QWidget>

#include <QString>

#include <QMainWindow>

#include "database.h"

#include <QMessageBox>

namespace **Ui** {

class **add\_bilet**;

}

class **add\_bilet** : public QWidget

{

Q\_OBJECT

public:

explicit **add\_bilet**(QWidget \*parent = nullptr);

~***add\_bilet***();

private slots:

void **on\_pushButton\_2\_clicked**();

void **on\_pushButton\_clicked**();

void **on\_pushButton\_3\_clicked**();

void **on\_pushButton\_4\_clicked**();

private:

Ui::add\_bilet \*ui;

database db;

};

#endif // ADD\_BILET\_H

Додаток Ж  
Лістинг файлу «add\_bilet.cpp»

#include "add\_bilet.h"

#include "ui\_add\_bilet.h"

add\_bilet::add\_bilet(QWidget \*parent) :

QWidget(parent),

ui(new Ui::add\_bilet)

{

ui->setupUi(this);

}

add\_bilet::~add\_bilet()

{

delete ui;

}

void add\_bilet::on\_pushButton\_2\_clicked()

{

close();

}

void add\_bilet::on\_pushButton\_clicked()

{

QSqlQuery query = QSqlQuery(db.getDB());

query.prepare("INSERT INTO Bilet (Місце,Вартість,Час\_купівлі,П\_І\_П\_Пасажира,Паспорт)"

"VALUES (?, ?, ?, ?, ?)");

QString misze = ui->lineEdit->text();

QString vartist = ui->lineEdit\_2->text();

QString chas\_kupivli = ui->dateTimeEdit->text();

QString p\_i\_p\_pasazir = ui->lineEdit\_4->text();

QString pasport = ui->lineEdit\_5->text();

query.addBindValue(misze);

query.addBindValue(vartist);

query.addBindValue(chas\_kupivli);

query.addBindValue(p\_i\_p\_pasazir);

query.addBindValue(pasport);

if(!query.exec())

{

qDebug() << query.lastError();

QMessageBox::critical(this,"Помилка!","Дані введено невірно!");

} else {QMessageBox::information(this,"Успіх!","Дані успішно добавлені!"); close();}

}

void add\_bilet::on\_pushButton\_3\_clicked()

{

QSqlQuery query = QSqlQuery(db.getDB());

query.prepare("delete from bilet "

"where id\_bilet=(?)");

QString id\_bilet = ui->lineEdit\_3->text();

query.addBindValue(id\_bilet);

if(!query.exec())

{

qDebug() << query.lastError();

QMessageBox::critical(this,"Помилка!","Введені дані неправильні!");

} else {QMessageBox::information(this,"Успіх!","Рядок видалено!");}

}

void add\_bilet::on\_pushButton\_4\_clicked()

{

QSqlQuery query = QSqlQuery(db.getDB());

query.prepare("update bilet set Місце=(?), Вартість=(?), Час\_купівлі=(?), П\_І\_П\_Пасажира=(?), Паспорт=(?) "

"where id\_bilet=(?)");

QString misze = ui->lineEdit->text();

QString vartist = ui->lineEdit\_2->text();

QString chas\_kupivli = ui->dateTimeEdit->text();

QString p\_i\_p\_pasazir = ui->lineEdit\_4->text();

QString pasport = ui->lineEdit\_5->text();

QString id\_bilet = ui->lineEdit\_6->text();

query.addBindValue(misze);

query.addBindValue(vartist);

query.addBindValue(chas\_kupivli);

query.addBindValue(p\_i\_p\_pasazir);

query.addBindValue(pasport);

query.addBindValue(id\_bilet);

if(!query.exec())

{

qDebug() << query.lastError();

QMessageBox::critical(this,"Помилка!","Введені дані неправильні!");

} else {QMessageBox::information(this,"Успіх!","Рядок із заданим ID обновлений!");}

}

Додаток И  
Лістинг файлу «add\_despetcher.h»

#ifndef ADD\_DESPETCHER\_H

#define ADD\_DESPETCHER\_H

#include <QWidget>

#include <QString>

#include <QMainWindow>

#include "database.h"

#include <QMessageBox>

namespace Ui {

class add\_despetcher;

}

class add\_despetcher : public QWidget

{

Q\_OBJECT

public:

explicit add\_despetcher(QWidget \*parent = nullptr);

~add\_despetcher();

private slots:

void on\_pushButton\_2\_clicked();

void on\_pushButton\_clicked();

void on\_pushButton\_3\_clicked();

void on\_pushButton\_4\_clicked();

private:

Ui::add\_despetcher \*ui;

database db;

};

#endif // ADD\_DESPETCHER\_H

Додаток К  
Лістинг файлу «add\_despetcher.cpp»

#include "add\_despetcher.h"

#include "ui\_add\_despetcher.h"

add\_despetcher::**add\_despetcher**(QWidget \*parent) :

QWidget(parent),

ui(new Ui::add\_despetcher)

{

ui->setupUi(this);

}

add\_despetcher::~***add\_despetcher***()

{

delete ui;

}

void add\_despetcher::**on\_pushButton\_2\_clicked**()

{

close();

}

void add\_despetcher::**on\_pushButton\_clicked**()

{

QSqlQuery query = QSqlQuery(db.getDB());

query.prepare("INSERT INTO despetcher (П\_І\_П,Дата\_народження,Адреса)"

"VALUES (?, ?, ?)");

QString p\_i\_p = ui->lineEdit->text();

QString data\_b = ui->lineEdit\_2->text();

QString adressa = ui->dateEdit->text();

query.addBindValue(p\_i\_p);

query.addBindValue(data\_b);

query.addBindValue(adressa);

if(!query.exec())

{

qDebug() << query.lastError();

QMessageBox::critical(this,"Помилка!","Дані введено невірно!");

} else {QMessageBox::information(this,"Успіх!","Дані успішно добавлені!"); close();}

}

void add\_despetcher::**on\_pushButton\_3\_clicked**()

{

QSqlQuery query = QSqlQuery(db.getDB());

query.prepare("delete from despetcher "

"where id\_despetcher=(?)");

QString id\_despetcher = ui->lineEdit\_3->text();

query.addBindValue(id\_despetcher);

if(!query.exec())

{

qDebug() << query.lastError();

QMessageBox::critical(this,"Помилка!","Введені дані неправильні!");

} else {QMessageBox::information(this,"Успіх!","Рядок видалено!");}

}

void add\_despetcher::**on\_pushButton\_4\_clicked**()

{

QSqlQuery query = QSqlQuery(db.getDB());

query.prepare("update despetcher set П\_І\_П=(?), Дата\_народження=(?), Адреса=(?)"

"where id\_despetcher=(?)");

QString p\_i\_p = ui->lineEdit->text();

QString data\_b = ui->lineEdit\_2->text();

QString adressa = ui->dateEdit->text();

QString id\_despetcher = ui->lineEdit\_4->text();

query.addBindValue(p\_i\_p);

query.addBindValue(data\_b);

query.addBindValue(adressa);

query.addBindValue(id\_despetcher);

if(!query.exec())

{

qDebug() << query.lastError();

QMessageBox::critical(this,"Помилка!","Введені дані неправильні!");

} else {QMessageBox::information(this,"Успіх!","Рядок із заданим ID обновлений!");}

}

Додаток Л  
Лістинг файлу «add\_marshrut.h»

#ifndef ADD\_MARSHRUT\_H

#define ADD\_MARSHRUT\_H

#include <QWidget>

#include <QString>

#include <QMainWindow>

#include "database.h"

#include <QMessageBox>

namespace Ui {

class add\_marshrut;

}

class add\_marshrut : public QWidget

{

Q\_OBJECT

public:

explicit add\_marshrut(QWidget \*parent = nullptr);

~add\_marshrut();

private slots:

void on\_pushButton\_2\_clicked();

void on\_pushButton\_clicked();

void on\_pushButton\_3\_clicked();

void on\_pushButton\_4\_clicked();

private:

Ui::add\_marshrut \*ui;

database db;

};

#endif // ADD\_MARSHRUT\_H

Додаток М  
Лістинг файлу «add\_marshrut.cpp»

#include "add\_marshrut.h"

#include "ui\_add\_marshrut.h"

add\_marshrut::**add\_marshrut**(QWidget \*parent) :

QWidget(parent),

ui(new Ui::add\_marshrut)

{

ui->setupUi(this);

}

add\_marshrut::~***add\_marshrut***()

{

delete ui;

}

void add\_marshrut::**on\_pushButton\_2\_clicked**()

{

close();

}

void add\_marshrut::**on\_pushButton\_clicked**()

{

QSqlQuery query = QSqlQuery(db.getDB());

query.prepare("INSERT INTO marshrut (Номер\_маршруту,Пункт\_призначення,Область,Район,Відстань,Час\_відправлення,Час\_прибуття)"

"VALUES (?, ?, ?, ?, ?, ?, ?)");

QString nomer\_marshrutu = ui->lineEdit->text();

QString punkt\_priznachenya = ui->lineEdit\_2->text();

QString oblast = ui->lineEdit\_2->text();

QString rayon = ui->lineEdit\_4->text();

QString vidstan = ui->lineEdit\_5->text();

QString chas\_vidpravlenya = ui->dateTimeEdit->text();

QString chas\_pributya = ui->dateTimeEdit\_2->text();

query.addBindValue(nomer\_marshrutu);

query.addBindValue(punkt\_priznachenya);

query.addBindValue(oblast);

query.addBindValue(rayon);

query.addBindValue(vidstan);

query.addBindValue(chas\_vidpravlenya);

query.addBindValue(chas\_pributya);

if(!query.exec())

{

qDebug() << query.lastError();

QMessageBox::critical(this,"Помилка!","Дані введено невірно!");

} else {QMessageBox::information(this,"Успіх!","Дані успішно добавлені!"); close();}

}

void add\_marshrut::**on\_pushButton\_3\_clicked**()

{

QSqlQuery query = QSqlQuery(db.getDB());

query.prepare("delete from marshrut "

"where id\_marshrut=(?)");

QString marshrut = ui->lineEdit\_6->text();

query.addBindValue(marshrut);

if(!query.exec())

{

qDebug() << query.lastError();

QMessageBox::critical(this,"Помилка!","Введені дані неправильні!");

} else {QMessageBox::information(this,"Успіх!","Рядок видалено!");}

}

void add\_marshrut::**on\_pushButton\_4\_clicked**()

{

QSqlQuery query = QSqlQuery(db.getDB());

query.prepare("update marshrut set Номер\_маршруту=(?), Пункт\_призначення=(?), Область=(?), Район=(?), Відстань=(?), Час\_відправлення=(?), Час\_прибуття=(?)"

"where id\_marshrut=(?)");

QString nomer\_marshrutu = ui->lineEdit->text();

QString punkt\_priznachenya = ui->lineEdit\_2->text();

QString oblast = ui->lineEdit\_2->text();

QString rayon = ui->lineEdit\_4->text();

QString vidstan = ui->lineEdit\_5->text();

QString chas\_vidpravlenya = ui->dateTimeEdit->text();

QString chas\_pributya = ui->dateTimeEdit\_2->text();

QString id\_marshrut = ui->lineEdit\_7->text();

query.addBindValue(nomer\_marshrutu);

query.addBindValue(punkt\_priznachenya);

query.addBindValue(oblast);

query.addBindValue(rayon);

query.addBindValue(vidstan);

query.addBindValue(chas\_vidpravlenya);

query.addBindValue(chas\_pributya);

query.addBindValue(id\_marshrut);

if(!query.exec())

{

qDebug() << query.lastError();

QMessageBox::critical(this,"Помилка!","Введені дані неправильні!");

} else {QMessageBox::information(this,"Успіх!","Рядок із заданим ID обновлений!");}

}

Додаток Н  
Лістинг файлу «database.h»

#ifndef DATABASE\_H

#define DATABASE\_H

#include <QWidget>

#include <QSqlQuery>

#include <QSqlDatabase>

#include <QtSql/QSqlDatabase>

#include <QtSql/QSqlTableModel>

#include <QtSql/QSqlError>

#include <QDebug>

class **database**

{

private:

QSqlDatabase db;

public:

**database**();

void **connect**();

QSqlDatabase **getDB**();

};

#endif // DATABASE\_H

Додаток П  
Лістинг файлу «database.cpp»

#include "database.h"

#include "add\_bilet.h"

database::**database**()

{

}

void database::**connect**()

{

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

db.setDatabaseName("C:/t/kasa.db");

if (!db.open()) {

qDebug() << db.lastError().text();

}

else {

qDebug() << "База даних підключена!";

}

}

QSqlDatabase database::**getDB**()

{

return db;

}

Додаток Р  
Лістинг файлу «vodiy.h»

#ifndef VODIY\_H

#define VODIY\_H

#include <QWidget>

#include <QString>

#include <QMainWindow>

#include "database.h"

#include <QMessageBox>

namespace Ui {

class vodiy;

}

class vodiy : public QWidget

{

Q\_OBJECT

public:

explicit vodiy(QWidget \*parent = nullptr);

~vodiy();

private slots:

void on\_pushButton\_2\_clicked();

void on\_pushButton\_clicked();

void on\_pushButton\_3\_clicked();

void on\_pushButton\_4\_clicked();

private:

Ui::vodiy \*ui;

database db;

};

#endif // VODIY\_H

Додаток С  
Лістинг файлу «vodiy.cpp»

#include "vodiy.h"

#include "ui\_vodiy.h"

vodiy::vodiy(QWidget \*parent) :

QWidget(parent),

ui(new Ui::vodiy)

{

ui->setupUi(this);

}

vodiy::~vodiy()

{

delete ui;

}

void vodiy::on\_pushButton\_2\_clicked()

{

close();

}

void vodiy::on\_pushButton\_clicked()

{

QSqlQuery query = QSqlQuery(db.getDB());

query.prepare("INSERT INTO vodiy (П\_І\_П,Дата\_народження,Паспорт)"

"VALUES (?, ?, ?)");

QString p\_i\_p = ui->lineEdit->text();

QString data\_b2 = ui->dateEdit->text();

QString pasport = ui->lineEdit\_3->text();

query.addBindValue(p\_i\_p);

query.addBindValue(data\_b2);

query.addBindValue(pasport);

if(!query.exec())

{

qDebug() << query.lastError();

QMessageBox::critical(this,"Помилка!","Дані введено невірно!");

} else {QMessageBox::information(this,"Успіх!","Дані успішно добавлені!"); close();}

}

void vodiy::on\_pushButton\_3\_clicked()

{

QSqlQuery query = QSqlQuery(db.getDB());

query.prepare("delete from vodiy "

"where id\_vodiy=(?)");

QString id\_vodiy = ui->lineEdit\_2->text();

query.addBindValue(id\_vodiy);

if(!query.exec())

{

qDebug() << query.lastError();

QMessageBox::critical(this,"Помилка!","Введені дані неправильні!");

} else {QMessageBox::information(this,"Успіх!","Рядок видалено!");}

}

void vodiy::on\_pushButton\_4\_clicked()

{

QSqlQuery query = QSqlQuery(db.getDB());

query.prepare("update vodiy set П\_І\_П=(?), Дата\_народження=(?), Паспорт=(?)"

"where id\_vodiy=(?)");

QString p\_i\_p = ui->lineEdit->text();

QString data\_b2 = ui->dateEdit->text();

QString pasport = ui->lineEdit\_3->text();

QString id\_vodiy = ui->lineEdit\_4->text();

query.addBindValue(p\_i\_p);

query.addBindValue(data\_b2);

query.addBindValue(pasport);

query.addBindValue(id\_vodiy);

if(!query.exec())

{

qDebug() << query.lastError();

QMessageBox::critical(this,"Помилка!","Введені дані неправильні!");

} else {QMessageBox::information(this,"Успіх!","Рядок із заданим ID обновлений!");}

}

Додаток Т  
Лістинг файлу «kvitok.h»

#ifndef KVITOK\_H

#define KVITOK\_H

#include <QWidget>

#include <QString>

#include <QMainWindow>

#include "database.h"

#include <QMessageBox>

namespace Ui {

class kvitok;

}

class kvitok : public QWidget

{

Q\_OBJECT

public:

explicit kvitok(QWidget \*parent = nullptr);

~kvitok();

private slots:

void on\_pushButton\_clicked();

private:

Ui::kvitok \*ui;

database db;

};

#endif // KVITOK\_H

Додаток У  
Лістинг файлу «kvitok.cpp»

#include "kvitok.h"

#include "ui\_kvitok.h"

kvitok::kvitok(QWidget \*parent) :

QWidget(parent),

ui(new Ui::kvitok)

{

ui->setupUi(this);

}

kvitok::~kvitok()

{

delete ui;

}

void kvitok::on\_pushButton\_clicked()

{

QSqlQuery query = QSqlQuery(db.getDB());

query.prepare("INSERT INTO perevezenya (Номер,id\_marshrut,id\_bilet,id\_vodiy,id\_despetcher)"

"VALUES (?, ?, ?, ?, ?)");

QString nomer = ui->lineEdit->text();

QString id\_marshrut = ui->lineEdit\_2->text();

QString id\_bilet = ui->lineEdit\_3->text();

QString id\_vodiy = ui->lineEdit\_4->text();

QString id\_despetcher = ui->lineEdit\_5->text();

query.addBindValue(nomer);

query.addBindValue(id\_marshrut);

query.addBindValue(id\_bilet);

query.addBindValue(id\_vodiy);

query.addBindValue(id\_despetcher);

if(!query.exec())

{

qDebug() << query.lastError();

QMessageBox::critical(this,"Помилка!","Дані введено невірно!");

} else {QMessageBox::information(this,"Успіх!","Квиток оформлено!"); close();}

}

Додаток Ф  
DVD-диск із програмним продуктом