**Форма № 17**

Міністерство освіти і науки України

**Національний університет «Запорізька політехніка»**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Кафедра програмних засобів\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(найменування кафедри)

**КУРСОВИЙ ПРОЄКТ**

**(РОБОТА)**

з дисципліни «Бази даних»

(назва дисципліни)

на тему: «Єдиний реєстр документів громадян України»

Студентів(ки) 2 курсу КНТ-229 групи

спеціальності 122 Комп’ютерні науки

освітня програма (спеціалізація) комп’ютерні науки

\_\_\_\_\_\_\_\_\_\_\_\_ Колбаса Д. О.\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Керівник к.т.н., доц. Каплієнко Т.І.

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

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

Кількість балів: \_\_\_\_\_\_\_\_\_\_Оцінка: ECTS \_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   Каплієнко Т. І.

(підпис) (прізвище та ініціали)

2021 рік

**Форма № 25**

Міністерство освіти і науки України

**Національний університет «Запорізька політехніка»**

( повне найменування закладу вищої освіти)

Інститут, факультет ІІРЕ, ФКНТ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра програмних засобів\_\_\_\_\_\_\_**\_\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**\_\_**\_\_\_

Ступінь вищої освіти      бакалавр**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Спеціальність 122 Комп’ютерні науки**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(код і найменування)

Освітня програма (спеціалізація) Комп’ютерні науки**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(назва освітньої програми (спеціалізації)

**ЗАТВЕРДЖУЮ**

**Завідувач кафедри ПЗ, д.т.н, проф.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_С.О. Субботін**

“\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_року

З А В Д А Н Н Я

**НА КУРСОВИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)**

 Колбаси Д. О.

(прізвище, ім’я, по батькові)

1. Тема проекту (роботи) Єдиний реєстр документів громадян України

керівник проекту (роботи)\_ к.т.н., доц. Каплієнко Т.І. ,

( прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від

2. Строк подання студентом проекту (роботи)\_25 травня 2021 року\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Вихідні дані до проекту (роботи) розробити програмне забезпечення, що реалізує єдиний держаний реєстр документів громадян України з використанням баз даних.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області; 2. Концептуальне проектування; 3. Логічне проектування; 4. Прийняті рішення під час реалізації програмної системи; 5. Керівництво оператора; 6. Додатки.     
5. Консультанти розділів проекту (роботи)

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посада  консультанта | Підпис, дата | |
| завдання видав | прийняв  виконане завдання |
| 1-5 Основна частина | Каплієнко Т.І., доцент |  |  |
|  |  |  |  |
|  |  |  |  |

6. Дата видачі завдання\_\_12 лютого 2021 р.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів курсового  проекту (роботи) | Строк виконання етапів проекту  ( роботи ) | Примітка |
| 1. | Аналіз індивідуального завдання. | 1 тиждень |  |
| 2. | Аналіз програмних засобів, що будуть використовуватись в роботі. | 2 тиждень |  |
| 3. | Аналіз структур даних, що необхідно використати в курсової роботі. | 3 тиждень |  |
| 4. | Затвердження завдання | 4 тиждень |  |
| 5. | Вивчення можливостей програмної реалізації структур даних та інтерфейсу користувача. | 5-9 тиждень |  |
| 6. | Аналіз вимог до апаратних засобів | 9 тиждень |  |
| 7. | Розробка програмного забезпечення | 9-13 тиждень |  |
| 8. | Проміжний контроль | 10 тиждень | Розділи 1-2 ПЗ |
| 9. | Оформлення, відповідних пунктів пояснювальної записки. | 10-14 тиждень | Розділи 1-5 ПЗ |
| 10. | Захист курсової роботи. | 15 тиждень |  |
|  |  |  |  |
|  |  |  |  |

**Студент (ка) \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_** Колбаси Д. О.**\_\_\_\_**

( підпис ) (прізвище та ініціали)

**Керівник проєкту (роботи) \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_**Каплієнко Т. І.**\_\_\_\_**

( підпис ) (прізвище та ініціали)

# ЗМІСТ

[ЗМІСТ 4](#_Toc74515843)

[РЕФЕРАТ 7](#_Toc74515844)

[ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 8](#_Toc74515845)

[ВСТУП 9](#_Toc74515846)

[1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 10](#_Toc74515847)

[1.1 Аналіз бюрократії як процесу, що потребує цифровізації та автоматизації 10](#_Toc74515848)

[1.2 Огляд існуючих аналогів сервісів надання електронних документів. 10](#_Toc74515849)

[2 КОНЦЕПТУАЛЬНЕ ПРОЕКТУВАННЯ 12](#_Toc74515850)

[2.1 Склад схеми даних 12](#_Toc74515851)

[2.2 Опис схеми даних 12](#_Toc74515852)

[2.3 Опис зв’язків схеми даних 13](#_Toc74515853)

[3 ЛОГІЧНЕ ПРОЕКТУВАННЯ 16](#_Toc74515854)

[3.1 Вибір інструментів для реалізації програмної системи 16](#_Toc74515855)

[3.2 Проектування схеми даних 16](#_Toc74515856)

[3.3 Робота над запитами 25](#_Toc74515857)

[3.4 Робота над звітами 31](#_Toc74515858)

[4 ПРИЙНЯТІ РІШЕННЯ ПІД ЧАС РЕАЛІЗАЦІЇ СИСТЕМИ 36](#_Toc74515859)

[5 керівництво ОПЕРАТОР 43](#_Toc74515860)

[5.1 Оформлення документів 44](#_Toc74515861)

[5.2 Пошук в базі даних 50](#_Toc74515862)

[5.3 Перегляд власних документів 53](#_Toc74515863)

[ВИСНОВКИ 61](#_Toc74515864)

[ПЕРЕЛІК ПОСИЛАНЬ 62](#_Toc74515865)

[ДОДАТОК А КОД ПРОГРАМИ 63](#_Toc74515866)

[А.1 Клас Program 63](#_Toc74515867)

[А.2 Клас Database 63](#_Toc74515868)

[А.3 Клас DBUtils 75](#_Toc74515869)

[А.4 Клас Users 75](#_Toc74515870)

[А.5 Клас UserException 77](#_Toc74515871)

[А.6 Клас MainForm 78](#_Toc74515872)

[А.7 Клас CreateForm 79](#_Toc74515873)

[А.8 Клас SearchForm 85](#_Toc74515874)

[А.9 Клас PersonalForm 89](#_Toc74515875)

[А.10 Клас ReportPassportForm 91](#_Toc74515876)

[А.11 Клас ReportBirthCertificateForm 92](#_Toc74515877)

[А.12 Клас ReportTaxpeyerCertificateForm 93](#_Toc74515878)

[А.13 Клас ReportReferenceFromUSRForm 93](#_Toc74515879)

[А.14 Клас ReportMarriageCertificateForm 94](#_Toc74515880)

[А.15 Клас ReportTransportCertificateForm 95](#_Toc74515881)

[А.16 Клас ReportPropertyOwnershipForm 96](#_Toc74515882)

[ДОДАТОК Б КОД ВБУДОВАНИХ ПРОЦЕДУР ТА ФУНКЦІЙ 98](#_Toc74515883)

[Б.1 Процедура sign\_up\_user 98](#_Toc74515884)

[Б.2 Процедура sign\_in\_user 98](#_Toc74515885)

[Б.3 Процедура record\_birth 98](#_Toc74515886)

[Б.4 Процедура record\_reference\_from\_usr 99](#_Toc74515887)

[Б.5 Процедура record\_taxpayer\_certificate 99](#_Toc74515888)

[Б.6 Процедура record\_passport 100](#_Toc74515889)

[Б.7 Процедура record\_marriage 100](#_Toc74515890)

[Б.8 Процедура record\_transport\_registration\_certificate 101](#_Toc74515891)

[Б.9 Процедура record\_property\_right\_certificate 101](#_Toc74515892)

[Б.10 Процедура record\_death 101](#_Toc74515893)

[Б.11 Процедура archivate\_reference\_from\_usr 102](#_Toc74515894)

[Б.12 Процедура archivate\_passport 102](#_Toc74515895)

[Б.13 Процедура archivate\_marriage 103](#_Toc74515896)

[Б.14 Процедура archivate\_transport\_registration\_certificate 103](#_Toc74515897)

[Б.15 Процедура archivate\_property\_right\_certificate 103](#_Toc74515898)

[Б.16 Процедура archivate\_taxpayer\_certificate 104](#_Toc74515899)

[Б.17 Процедура archivate\_birth\_certificate 104](#_Toc74515900)

[Б.18 Процедура find\_car\_by\_registration\_id 104](#_Toc74515901)

[Б.19 Процедура find\_property\_by\_owner\_passport\_id 104](#_Toc74515902)

[Б.20 Процедура find\_children\_by\_dead\_parent 105](#_Toc74515903)

[Б.21 Процедура find\_passport\_data 105](#_Toc74515904)

[Б.22 Процедура get\_passport 105](#_Toc74515905)

[Б.23 Процедура get\_taxpayer\_certificate 106](#_Toc74515906)

[Б.24 Процедура get\_reference\_from\_usr 106](#_Toc74515907)

[Б.25 Процедура get\_birth\_certificate 106](#_Toc74515908)

[Б.26 Процедура get\_marriage\_certificate 107](#_Toc74515909)

[Б.27 Процедура get\_transport\_certificate 107](#_Toc74515910)

[Б.28 Процедура get\_property\_ownership\_certificate 108](#_Toc74515911)

[Б.29 Функція is\_username\_exists 108](#_Toc74515912)

[Б.30 Функція is\_user\_passport\_exists 108](#_Toc74515913)

[Б.31 Функція is\_passport\_exists 109](#_Toc74515914)

[Б.32 Функція is\_birth\_certificate\_exists 109](#_Toc74515915)

[Б.33 Функція is\_password\_correct 109](#_Toc74515916)

[Б.34 Функція is\_passport\_issued 109](#_Toc74515917)

[Б.35 Функція is\_reference\_from\_usr\_exists 110](#_Toc74515918)

[Б.36 Функція is\_reference\_from\_usr\_already\_archivated 110](#_Toc74515919)

[Б.37 Функція is\_passport\_has\_reference\_from\_usr 110](#_Toc74515920)

[Б.38 Функція is\_taxpayer\_certificate\_exists 111](#_Toc74515921)

[Б.39 Функція is\_passport\_has\_taxpayer\_id 111](#_Toc74515922)

[Б.40 Функція is\_birth\_certificate\_exists 111](#_Toc74515923)

[Б.41 Функція is\_passport\_has\_birth\_certificate 111](#_Toc74515924)

[Б.42 Функція is\_marriage\_certificate\_exists 112](#_Toc74515925)

[Б.43 Функція count\_of\_property\_certificates 112](#_Toc74515926)

[Б.44 Функція count\_of\_transport\_certificates 112](#_Toc74515927)

# **РЕФЕРАТ**

Мета даного курсового проекту є – навчитись розробляти програмне забезпечення, що використовує бази даних.

Проведено аналіз предметної області, досліджено аналогічне програмне забезпечення, а також існуючі методи та програмні засоби для вирішення завдання.

Для реалізації програмного продукту використовувалася мова програмування С#, платформа .Net та середа розробки Visual Studio 2019.

Розробка бази даних, складалася зі створення схеми даних та вбудованих процедур та функцій. Розробка застосунку, що використовує базу даних складалася зі створення графічного інтерфейсу, та класу взаємодії з базою даних.

Здійснено опис прийнятих рішень, наведено опис полів та методів реалізованих класів, а також таблиць, процедур та функцій бази даних.

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ЄДР – Єдиний Державний Реєстр

БД – база даних

КПП – Код Платника Податків

РМП – реєстрація місця проживання

СПН – свідоцтво про народження

СКБД – система керування базами даних

# ВСТУП

У сучасному світі дуже важливе значення в житті кожного громадянина мають документи, що посвідчують його персону, права на власність майна та сімейний стан. Кожна людина має досить багато документів, що зберігаються у неструктурованому вигляді, що викликає проблеми, коли треба їх знайти і зробити це швидко. Державні органи, що займаються оформлення та виданням таких документів мають у своєму підпорядкуванні дуже великі об’єми персональних даних, але отримання цих даних громадянином, найчастіше являє собою досить не швидку та не приємну процедуру.

В сучасному країнах світу, де існує тенденція цифровізації, з’являється надія на те, що більшість в цих проблем є тимчасовими і дуже скоро взагалі зникнуть.

В своєму курсовому проекті я виклав своє бачення системи, що об’єднує досить невелику кількість документів, що мають значення під час комунікації громадянина та державної структури.

Ця система дозволяє створювати записи підчас оформлення документів, проводити процес визначення особистості, що буває дуже важливо для таких державних органів, як поліція, прокуратура та СБУ. І нарешті система дає змогу громадянам мати цифрові копії їх документів.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

В даному розділі викладено аналіз предметної області, а саме взаємний зв’язок різних документів посвідчення особи громадянина та права власності. А також розглянутий існуючий аналог програмного забезпечення «Дія» проекту «Держава в смартфоні».

## 1.1 Аналіз бюрократії як процесу, що потребує цифровізації та автоматизації

Комунікація між державою як соціальною структурою та окремим громадянином відбувається за допомоги документів, свідоцтв та витягів з різного роду реєстрів. Це зручний інструмент, що дозволяє державі проводити гнучку соціальну політику, надавати пільги окремим категоріям громадян, доводити свої права на власність майна або частку майна.

Всі документи мають зв’язок один з один, наприклад, при отримані паспорту у перший раз необхідно мати КПП (Код Платника Податків), свідоцтво про реєстрацію місця проживання та свідоцтво про народження. Для того щоб отримати КПП потрібен або паспорт або свідоцтво про народження.

Однак іноді буває дуже не зручно коли один з документів губиться, коли він потрібен, або взагалі псується. Рішенням цієї проблеми може стати пред’явлення електронної копії.

## 1.2 Огляд існуючих аналогів сервісів надання електронних документів.

1.2.1 Застосунок «Дія»

Застосунок «Дія» був розроблений в рамках державного проекту «Держава в смартфоні». Застосунок дозволяє мати доступ до таких документів як паспорт, КПП, свідоцтво про народження дитини та навіть студентський квиток за допомоги лише смартфона. Вхід до особистого кабінету реалізовується за допомоги BankId або Приват24. Серед послуг, що надає цей сервіс є також заміна посвідчення водія, оплата штрафів за порушення ПДР та реєстрація місця проживання[1].

1.2.2 Програма eGovernment

Програма eGovernment розроблена в рамках естонського проекту Electronic Government. Суть проекту перенести якомога більше процедур взаємодія громадянина з державним апаратом в електронну форму. Оформлення довідок, подання заяв, реєстрація народження дитини, тощо. Ідея також полягає у знижені рівня корумпованості державного апарату, оскільки електронні сервіси не беруть хабарів за прискорення процедур оформлення документів. Також це дозволяє сформувати прозору систему контролю за всім процесом[2].

# 2 КОНЦЕПТУАЛЬНЕ ПРОЕКТУВАННЯ

## 2.1 Склад схеми даних

Під час розроблення БД була побудована схема даних, що включала такі документи:

* Паспорт;
* Свідоцтво про народження;
* Довідка про КПП;
* Довідка про реєстрацію місця проживання;
* Свідоцтво про шлюб;
* Свідоцтво про реєстрацію транспорту;
* Свідоцтво про право на власність нерухомого майна.

Були розроблені фондові та архівні таблиці. Фондові таблиці зберігають інформацію, що належить живим громадянам, архівні – мертвим.

Архівована інформація про паспорт, реєстрацію місця проживання, свідоцтво про народження та шлюб може бути використана для генеалогічних досліджень.

Архівована інформація про власність рухомого та нерухомого майна може бути використана в процедурі розділу майна між нащадками після смерті людини. Таким чином дуже зручно виявити чим володів громадянин до смерті.

Інформація про користувачів застосунку буде зберігатися в окремій таблиці (логін, пароль, рівень доступу та паспорт).

## 2.2 Опис схеми даних

Всі документи, що посвідчують особистість (паспорт, довідка про КПП, довідка про РМП (реєстрацію місця проживання), навіть СПН (свідоцтво про народженння)) мають свої унікальні номери. Паспорт вміщує в себе інформацію про всі вище перераховані відомості. (КПП, номер запису РМП, та відомості, про місце народження, що зберігаються в самому свідоцтві, але доступ до них краще отримувати маючи зовнішній ключ у якості номеру СПН).

Свідоцтво про шлюб має ПІБ чоловіка та дружини, але для створення зв’язків у схемі даних, на мою думку, краще використовувати номери паспортів, які при формуванні звіту будуть заміщуватися на ПІБ. За цим же принципом краще зробити зв’язки між свідоцтвом про реєстрацію транспорту і паспортом, та свідоцтвом про реєстрацію нерухомого майна і паспортом.

Користувачі застосунку поділяються на 3 групи за рівнем доступу до БД:

* Користувачі, яким доступний лише перегляд лише своїх документів;
* Користувачі, яким доступний лише перегляд всіх документів;
* Користувачі, яким доступне створення (оформлення) та пошук документів.

До першої групи користувачів відносяться люди, які зареєструвалися в додатку. Під час реєстрації від них вимагається номер паспорту, який має бути присутній в реєстрі, а також на який не створено облікових записів. Вони можуть переглядати всі фондові дані, які пов’язані з певним номером паспорту.

До другої групи відносяться користувачі державних органів таких як поліція, СБУ, та інші, що мають потребу переглядати відомості громадян в межах розслідувань.

До третьої групи відносяться користувачі державних органів, що займаються оформленням документів.

Останні дві групи не мають шляху реєстрування, паспорти працівників цих відомств не прив’язуються до облікових записів. А самі облікові записи мають бути створені власноруч головним оператором бази даних, шляхом додавання відомостей про логін та пароль, а також рівень дозволу для облікового запису.

## 2.3 Опис зв’язків схеми даних

Концептуальну схему даних наведено на рисунку 2.1.

На рисунку 2.1 видно дві групи зв’язаних таблиць. Група звехру – таблиці фондових даних, група знизу – таблиці архівних даних.

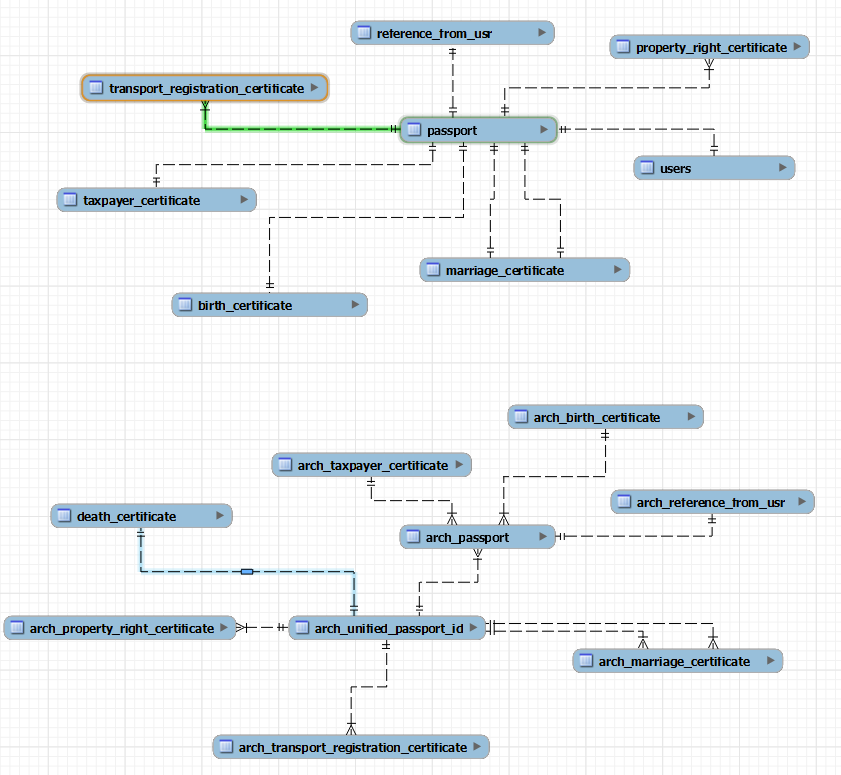


Рисунок 2.1 – Концептуальна схема даних

2.3.1 Опис зв’язків таблиць фондових даних

Номер паспорту – є ключовим ідентифікатором громадянина в державній системі, тож таблиця passport має центральне значення.

Свідоцтво про народження, КПП, поточне місце реєстрації та особистий кабінет у системі Єдиного Держаного Реєстру у кожного громадянина може бути лише однин, тому зв’язок між таблицями passport та birth\_certificate, passport та taxpayer\_certificate, passport та reference\_from\_usr, passport та users 1:1.

Свідоцтво про шлюб може мати лише 2 різних паспорти, тому між таблицями passport та marriage\_certificate існує 2 зв’язки 1:1.

Кожний громадянин може володіти багатьма, одним, або взагалі не володіти транспортним засобом, тому зв’язок між таблицями passport та transport\_registration\_certificate 1 : М.

Так само і з свідоцтвом на права власності на нерухоме майно зв’язок між passport та property\_right\_certificate 1 : М.

2.3.2 Опис зв’язків таблиць архівних даних

За своє життя людина отримує декілька паспортів. Вони можуть містити різні відомості про стать та ПІБ, але точно незмінним залишається номер паспорту. Тож всі документи, що посилаються на номер паспорту (свідоцтво про смерть, свідоцтво про володіння транспортом та свідоцтво про володіння нерухомим майном) зв’язані з таблицею arch\_unified\_passport\_id, що містить 1 колонку – номер паспорту, який є унікальним в межах цієї таблиці. Зв’язок таблиці arch\_unified\_passport\_id з таблицями arch\_transport\_registration\_certificate, arch\_property\_right\_certificate, arch\_marriage\_certificate та acrh\_passport 1 : М. Також ця таблиця має зв’язок зі свідоцтвом про смерть 1 : 1, лише тому, що в даному випадку неважливо з якою таблицею (arch\_unified\_passport\_id або arch\_passport) воно зв’язано.

Через те, що за життя людина може мати лише один КПП та лише одне свідоцтво про народження, та ще й ці документи мають відношення до паспорту не за його номером а за своїми номерами, то і в схемі даних arch\_birth\_certificate та arch\_taxpayer\_certificate відносяться до arch\_passport як 1 : М.

Місце проживання людина може змінювати необмежену кількість разів протягом життя тому зв’язок arch\_reference\_from\_usr з arch\_passport 1 : 1. Це найзручніший варіант.

# 3 ЛОГІЧНЕ ПРОЕКТУВАННЯ

## 3.1 Вибір інструментів для реалізації програмної системи

3.1.1 Вибір СКБД (Система Керування Базами Даних) для проектування БД

Для розробки БД в курсовому проекті я обрав СКБД MySQL. Головною перевагою ції СКБД є її поширеність та наявність великої кількості документації[3] та відеоуроків.

3.1.2 Вибір платформи та мови програмування

Для реалізації програмного застосунку була обрана платформа .Net та мова програмування C#.

С# - об’єктно-орієнтована мова програмування, що доволяє представити програму як набір об’єктів, кожен з яких має свої властивості, поведінку і своє призначення в програмі. Завдяки фреймворку .NetFramework розробка застосунків з графічним інтерфейсом дуже сильно спрощується в порівнянні з фреймворком Qt, що використовується з мовою С++[4].

Взаємодія з БД виконується за модулю MySQLConnector-.Net, що також спрощує розробку програмної системи.

Розробка звітів виконувалась за допомоги розширення Microsoft RDLC Report Designer, що створений на мові С#[5].

Таким чином можна зробити висновок, що C# та платформа .Net є, якщо не найкращим, то безумовно зручним вибором.

## 3.2 Проектування схеми даних

Схема фондових даних зображена на рисунку 3.1.

Схема архівних даних зображена на рисунку 3.2.

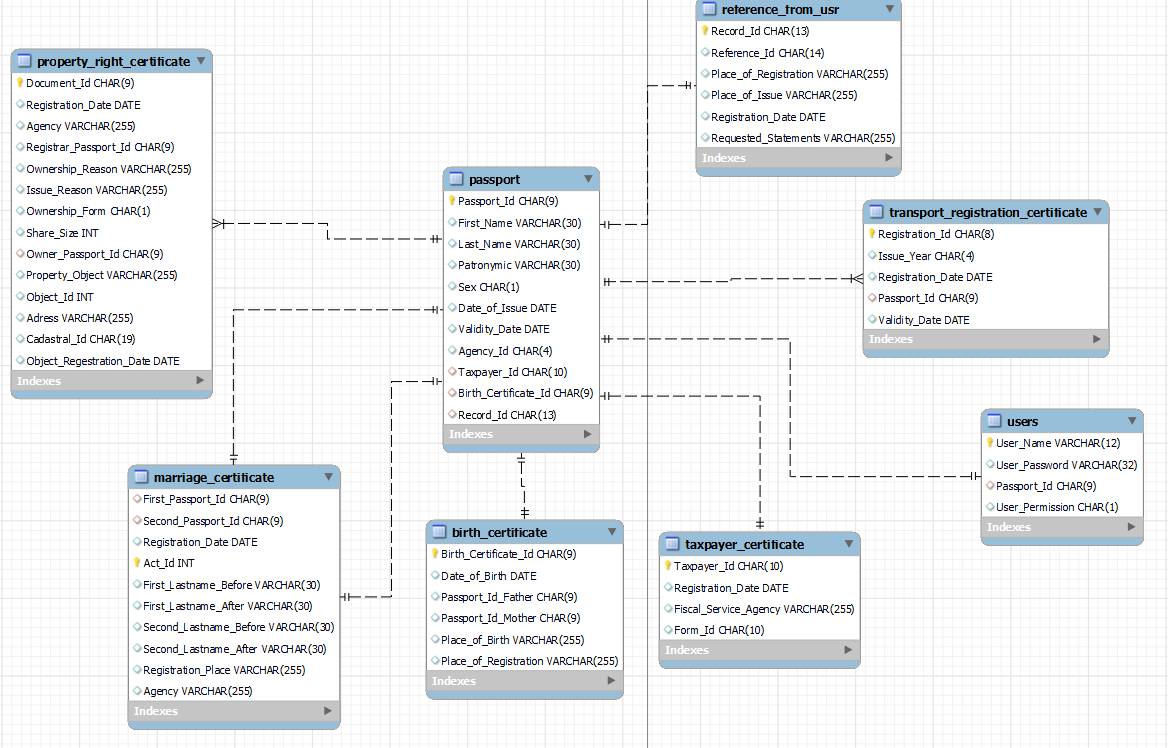


Рисунок 3.1 – Схема фондових даних

3.2.1 Таблиця passport

Зберігає данні про ПІБ, стать, дату видачі, дату дійсності, орган, що видав, та данні, що використовуються для зв’язку з іншими документами.

Як вже відзначалося у попередньому розділі, паспорт грає абсолютну центральну роль у системі даних, тобто всі документи будуть пов’язані з ним.

Ключові поля:

* Passport\_Id типу char(9) not null.
* Унікальні поля:
* Taxpayer\_Id типу char(10) default null;
* Birth\_Certificate\_Id типу char(9) default null;
* Record\_Id типу char(13) default null.

Унікальні поля є зовнішніми ключами, що пов’язують таблицю passport з таблицями taxpayer\_certificate (довідка про КПП), birth\_certificate (СПН), reference\_from\_usr (дослівно «витяг з ЄДР», але фактично це довідка про РМП).

3.2.2 Таблиця birth\_certificate

Зберігає дані про номер довідки, номери паспортів батька та матері, дату народження, місце народження, місце реєстрації (фактично орган, що оформив свідоцтво).

Ключові поля:

* Birth\_Certificate\_Id типу char(9) not null.

Унікальних полів не має. Зовнішніх ключів не має.

3.2.3 Таблиця taxpayer\_certificate

Зберігає дані про КПП, дату внесення до реєстру, орган фіскальної служби, що надав довідку та номер форми

Ключаві поля:

* Taxpayer\_Id типу char(9) not null.

Унікальних полів не має. Зовнішніх ключів не має.

3.2.4 Таблиця reference\_from\_usr

Зберігає дані про номер запису в реєстрі, номер довідки, місце реєстрації (адреса проживання), місце видачі (орган, що видав), дату реєстрації та дані що запрошуються (зазвичай місце проживання, але за потреби можна вказати інше).

Ключаві поля:

* Record\_Id типу char(13) not null.

Унікальних полів не має. Зовнішніх ключів не має.

3.2.5 Таблиця marriage\_certificate

Зберігає дані про номер свідоцтва (акту), номера паспортів чоловіка та дружини, їх прізвища до оформлення шлюбу та після, дату оформлення місце реєстрації та орган, що видав свідоцтво.

Ключові поля:

* Act\_Id типу int not null.

Унікальні поля:

* First\_Passport\_Id типу char(9) default null;
* Second\_Passport\_Id типу char(9) default null.

Унікальні полі є зовнішніми ключами до таблиці passport. Кожен громадянин може перебувати лише у одному шлюбі, тому номера паспортів унікальні.

3.2.6 Таблиця transport\_certificate

Зберігає дані про реєстраційний номер транспортного засобу, дату реєстрації, номер паспорту володаря, дату випуску транспорту та місце реєстрації.

Ключові поля:

* Registration\_Id типу char(8) not null.

Унікальних полів немає.

Зовнішній ключ – Passport\_Id для зв’язку з таблицею passport.

3.2.7 Таблиця property\_rigth\_certificate

Зберігає дані про номер свідоцтва, дату реєстрації, агентство, номер паспорту реєстратора, причину набуття власності, причину видачі (свідоцтва), форму власності (приватна, колективна), частку, номер паспорту власника, об’єкт (нерухомого майна), номер об’єкту, адресу об’єкту, кадастровий номер об’єкту, дату реєстрації об’єкту.

Ключові поля:

* Document\_Id типу char(9) not null.

Унікальних полів не має.

Зовнішній ключ – Owner\_Passport\_Id для зв’язку з таблицею passport.

3.2.8 Таблиця users

Зберігає дані про логін, пароль, паспорт користувача та рівень доступу.

Ключові поля:

* User\_Name типу varchar(12) not null.

Унікальні поля:

* Passport\_Id типу char(9) default null.

Унікальне поле є зовнішнім ключем, що зв’язується з таблицею passport.

Колонка Permission (рівень доступу) має приймати лише такі значення:

* S (self read) – рівень доступу звичайного користувача, що може переглядати лише свої і лише актуальні документи.
* A (all read) – рівень доступу, що дозволяє виконувати пошук за всіма документами (паспортні дані, зареєстрований транспорт, зареєстрована нерухомість, пошук нащадків за номером свідоцтва про смерть).
* C (create) – рівень доступу, що дозволяє створювати нові записи в реєстрі, тобто оформлювати документи.

Поле Passport\_Id повинно дорівнювати null якщо рівень доступу «А» або «С» і не повинно дорівнювати null якщо рівень доступу «S».

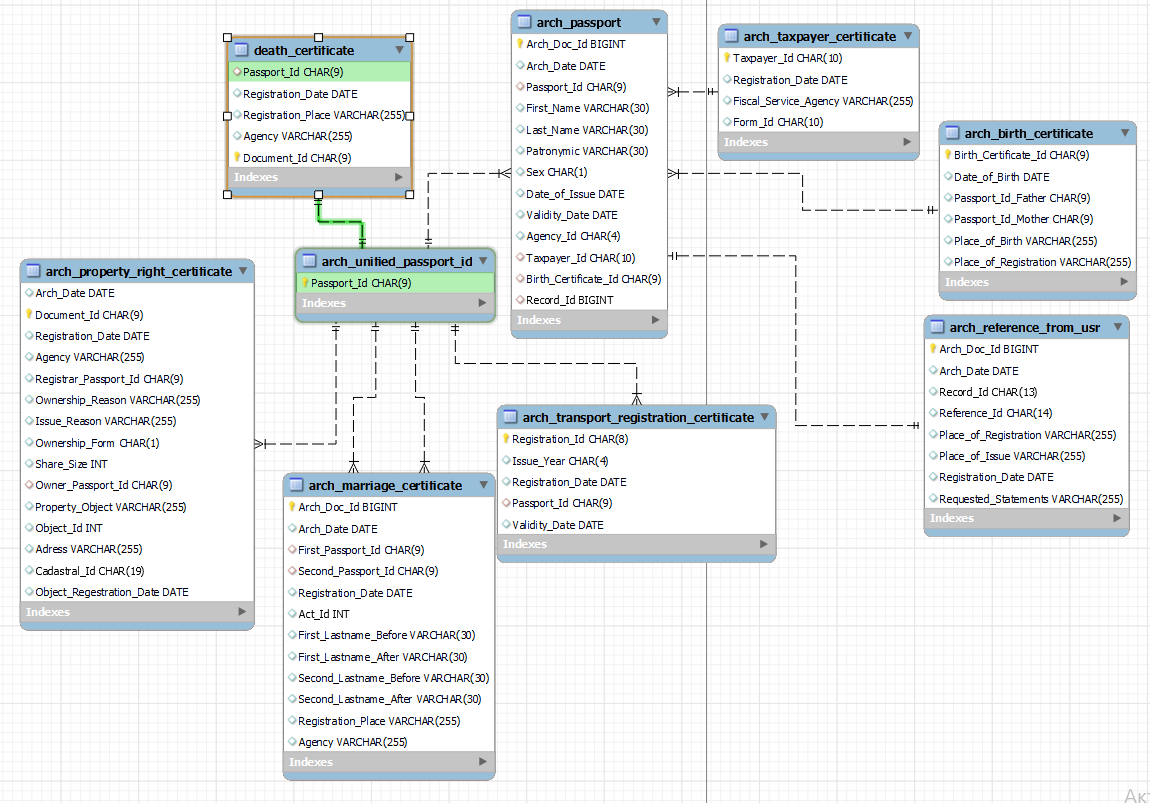


Рисунок 3.2 – Схема архівних даних

3.2.9 Таблиця death\_certificate

Зберігає дані про номер паспорту, дату смерті, місце реєстрації та орган, що видав свідоцтво.

Ключові поля:

* Document\_Id типу char(9) not null.

Унікальних полів не має.

Зовнішні ключ - Passport\_Id типу char(9) для зв’язку з таблицею arch\_passport через таблицю arch\_unified\_passport\_id.

3.2.10 Таблиця arch\_unified\_passport\_id

Являє собою проміжну таблицю між таблицями death\_certificate, arch\_property\_right\_certificate, arch\_marriage\_certificate, arch\_transport\_registration\_certificate та arch\_passport. Реалізує з’єднання М:М між вище перерахованими таблицями.

Ключові поля:

* Passport\_Id типу char(9)

Унікальних полів немає. Зовнішніх ключів не має.

3.2.11 Таблиця arch\_passport

Зберігає всі дані, що і таблиця passport, а також архівний номер документу та дату архівації. Заповнюється при оформлені нового паспорту(в тому числі і першого, бо деякі дані, з СПН зберігаються і в таблиця passport) або після смерті.

Ключові поля:

* Arch\_Doc\_Id типу bigint not null.

Унікальних полів не має.

Зовнішні ключі:

* Passport\_Id типу char(9);
* Birth\_Certificate\_Id типу char(9);
* Taxpayer\_Id типу char(10);
* Record\_Id типу char(13).

Зовнішні ключі зв’язують дану таблицю з таблицями arch\_unified\_passport\_id (M : 1), arch\_birth\_certificate (M : 1), arch\_taxpayer\_certificate (M : 1), arch\_reference\_from\_usr (1 : 1).

3.2.12 Таблиця arch\_birth\_certificate

Зберігає всі дані, що і таблиця birth\_certificate. Заповнюється тільки після смерті громадянина.

Ключові поля:

* Birth\_Certificate\_Id типу char(9) not null.

Унікальних полів не має. Зовнішніх ключів не має.

3.2.13 Таблиця arch\_taxpayer\_certificate

Зберігає всі дані, що і таблиця taxpayer\_certificate. Заповнюється тільки після смерті громадянина.

Ключові поля:

* Taxpayer\_Id типу char(10) not null.

Унікальних полів не має. Зовнішніх ключів не має.

3.2.14 Таблиця arch\_reference\_from\_usr

Зберігає всі дані, що і таблиця reference\_from\_usr, та номер архівного документу та дату архівації. Заповнюється даними після оформлення нового місця проживання або смерті.

Ключові поля:

* Arch\_Doc\_Id типу bigint not null.

Унікальні поля відсутні. Зовнішні ключі відсутні.

3.2.15 Таблиця arch\_marriage\_certificate

Зберігає всі дані, що і таблиця marriage\_certificate, та номер архівного документу та дату архівації. Заповнюється даними після розлучення, або смерті одного з членів шлюбу.

Ключові поля:

* Arch\_Doc\_Id типу bigint not null.

Унікальні полі відсутні.

Зовнішні ключі:

* First\_Passport\_Id типу char(9);
* Second\_Passsport\_Id типу char(9).

Використовуються для зв’язку з arch\_passport через таблицю arch\_unified\_passport\_id.

3.2.16 Таблиця arch\_transport\_registration\_certificate

Зберігає всі дані, що і таблиця transport\_registration\_certificate.

Коли громадянин помирає, всі дані, що стосуються його в таблиці transport\_registration\_certificate переміщуються в архівну. Сенс архівної таблиці полягає в тому, що перед тим як нерухомість буде розділена між нащадками, ці дані мають десь зберігатись. Саме для цього і існує описувана таблиця.

Ключові поля:

* Registration\_Id типу char(8) not null.

Унікальні поля відсутні.

Зовнішній ключ – Passport\_Id зв’язується з таблицею arch\_passport через таблицю arch\_unified\_passport\_id М : М.

3.2.17 Таблиця arch\_property\_right\_certificate

Зберігає всі дані, що і таблиця property\_right\_certificate та дату архівації.

Коли громадянин помирає, всі дані, що стосуються його в таблиці property\_right\_certificate переміщуються в архівну. Сенс архівної таблиці полягає в тому, що перед тим як нерухомість буде розділена між нащадками, ці дані мають десь зберігатись. Саме для цього і існує описувана таблиця.

Ключові поля:

* Document\_Id типу char(9) not null.

Унікальні поля відсутні.

Зовнішній ключ – Passport\_Id зв’язується з таблицею arch\_passport через таблицю arch\_unified\_passport\_id М : М.

## 3.3 Робота над запитами

Абсолютно всі запити до БД реалізовані у вигляді виклику вбудованих процедур, функцій та подань з передачею потрібних їм параметрів(окрім останніх). Це дозволяє скоротити витрати трафіку на передачу тексту запиту до MySQL- серверу, а також зробити процес відлагоджування більш зручним[6].

Програмний застосунок викликає лише деяку частку вбудованих процедур, які в свою чергу викликають функції, інші вбудовані процедури та використовують подання.

Процедури, що використовуються іншими процедурам, в цілому є процедурами архівації.

Функції, майже всі, повертають булеве значення, наприклад: чи існує номер паспорту? Чи має паспорт певний КПП? Чи має він взагалі КПП? Чи був паспорт виданий? Чи існує ім’я користувача? Чи вірний пароль? І так далі.

В залежності від значення, що повертають функції може відбуватися архівація певного запису перед створення нового, або відбуватися кидання ексепшену.

Деякі запити описувані в таблицях нижче використовують подання passport\_data. Це подання має колонки з таких таблиць:

* passport (всі, включно з КПП);
* birth\_certificate(всі);
* reference\_from\_usr(всі);
* marriage\_certificate (відомості про номер акту, номери паспортів чоловіка та дружини).

Нижче приведена таблиці 3.1 3.2 та 3.3 опису всіх розроблених вбудованих процедур, що викликаються застосунком. Всі інші функції та процедури, на мою думку не має сенсу описувати, але код усіх процедур та функцій розміщено у додатку Б.

Таблиця 3.1 – Вбудовані процедури, що викликаються застосунком при взаємодії з формою CreateForm(створення документів).

|  |  |
| --- | --- |
| Назва процедури та параметри | Опис роботи процедури |
| sign\_up\_user(in username varchar(12), in user\_password varchar(32), in permission char(1), in passport\_id char(9)) | Вносить значення параметрів до таблиці users, якщо ім’я користувача не зайнято, номер паспорту існує в таблиці passport та не має інших облікових записів з таким номером паспорту. |
| sign\_in\_user(in username varchar(12), in user\_password varchar(32), out permission char(1), out passport\_id char(9)) | Якщо параметри username та user\_password існують та знаходяться в одному рядку таблиці users то повертає відповідний рівень доступу та паспорт користувача. |

Продовження таблиці 3.1

|  |  |
| --- | --- |
| record\_birth(in passport\_id char(9), in first\_name varchar(30), in last\_name varchar(30), in patronimic varchar(30), in sex char(1), birth\_cert\_id char(9),in date\_of\_birth date, in passport\_f char(9), in passport\_m char(9), in place\_of\_birth varchar(255), in place\_of\_reg varchar(255)) | Приймає параметри про народження, та перевіряє чи існує номер паспорту та номер свідоцтва про народження, якщо ні то робить новий запис в таблицях. |
| record\_reference\_from\_usr(in passport char(9), in record char(13), in reference\_id char(14), in place\_of\_reg varchar(255), in place\_of\_issue varchar(255), in requested\_statements varchar(255)) | Записує відомості про місце проживання якщо паспорт існує. Якщо на цей паспорт існує зареєстроване місце проживання, то архівує старий запис і створює новий. |
| record\_taxpayer\_certificate(in passport\_id char(9), in taxpayer\_id char(10), in fiscal\_service\_agency varchar(255), in form\_id char(10)) | Записує відомості про КПП, якщо номер паспорту існує, КПП не існує та паспорт не має КПП. |
| record\_passport(in passport\_id char(9), in first\_name varchar(30), in last\_name varchar(30), in patronimic varchar(30), in sex char(1), in date\_of\_issue date, in agency\_id char(4), in taxpayer\_id char(10), in birth\_cert\_id char(9), in record\_id char(13)) | Записує відомості про паспорт, якщо номер паспорту існує(вперше створюється при оформленні народження), існує КПП, існує свідоцтво про РМП, існує свідоцтво про народження та вони всі відповідають даному номеру паспорту. Архівує старі відомості. |

Продовження таблиці 3.1

|  |  |
| --- | --- |
| record\_marriage(in passport\_h char(9), in passport\_w char(9), in reg\_place varchar(255), in agency varchar(255), in act\_id int) | Записує відомості про шлюб, якщо існують обидва номери паспорту(чоловіка та дружини). Якщо існує інший шлюб пов’язаний з одним з номерів паспортів, то архівує його (Це можна вважати розлученням). |
| record\_transport\_registration\_certificate(in reg\_id char(8), in issue\_year char(4), in passport\_id char(9)) | Записує дані про транспортний засіб, якщо номер паспорту існує. |
| record\_property\_right\_certificate(in document\_id char(9), in agency varchar(255), in registrar\_passport\_id char(9), in ownership\_reason varchar(255), in issue\_reason varchar(255), in ownership\_form char(1), in share\_size int, in passport\_id char(9), in property\_object varchar(255), in object\_id int, in adress varchar(255), in cadastral\_id char(19), in object\_reg\_date date) | Записую відомості про нерухоме майно, якщо номер паспорту володаря існує. |
| record\_death(in passport\_id char(9), in document\_id char(9), in reg\_place varchar(255), in agency varchar(255)) | Якщо паспорт померлого існує, то архівує абсолютно всі документи пов’язані з ним, та записує відомості про смерть в таблицю death\_certificate. |

Таблиця 3.2 – Вбудовані процедури, що викликаються застосунком, при взаємодії з формою SearchForm(пошук за всіма документами всіх громадян).

|  |  |
| --- | --- |
| Назва процедури та параметри | Опис роботи процедури |
| find\_car\_by\_registration\_id(in reg\_id char(8)) | Шукає всі записи з таблиці transport\_certificate за реєстраційним номером. Сортує за датою реєстрації. |
| find\_property\_by\_owner\_passport\_id(in passport char(9)) | Шукає всі записи з таблиці property\_right\_certificate за номером паспорту. Сортує за датою реєстрації. |
| find\_children\_by\_dead\_parent(in doc\_id char(9)) | Шукає всі записи з подання passport\_data, де в колонках Passport\_Id\_Father або Passport\_Id\_mother вказано паспорт, що належить свідоцтву про смерть з відповідним номером документу. Тобто шукає живих нащадків. |
| find\_passport\_data(in passport char(9), in tax char(10), in ref char(13), in first\_name varchar(30), in last\_name varchar(30), in patronimic varchar(30), in place varchar(255)) | Ця процедура приймає параметри щодо номеру документу (паспорт, КПП, довідка РМП), ПІБ, та місця реєстрації. Виконує пошук за вказаними параметрами. При чому не обов’язково передавати всі параметри або взагалі іх передавати. Повертає всі відповідні записи з подання passport\_data. |

Таблиця 3.3 – Вбудовані процедури, що викликаються при взаємодії з формою PersonalForm, тобто повертають дані для формування звітів.

|  |  |
| --- | --- |
| Назва процедури та параметри | Опис роботи процедури |
| get\_passport(in passport char(9)) | Шукає паспортні дані(ПІБ, стать, дату видачі, дату дійсності, орган, що видав, КПП, номер свідоцтва про РМП, дату народження, місце народження) за номером паспорту в поданні passport\_data. |
| get\_taxpayer\_certificate(in passport char(9)) | Шукає дані довідки про КПП (ПІБ, дату народження, та всі колонки таблиці taxpayer\_certificate) в таблицях passport, birth\_certificate, taxpayer\_certificate. |
| get\_reference\_from\_usr(in passport char(9)) | Шукає дані довідки РМП (всі поля, але замість паспорту – ПІБ та дата народження) з таблиць passport, birth\_certificate та reference\_from\_usr. |
| get\_birth\_certificate(in passport char(9)) | Шукає дані СПН (всі поля, а також ПІБ, та замість номерів паспортів батьків їх ПІБ). Якщо паспортів батьків не існує(в фондовій таблиці то шукає в архівній). |
| get\_marriage\_certificate(in passport char(9)) | Шукає дані про шлюб(всі поля, але номера паспортів замінюються на ПІБ). |
| get\_transport\_certificate(in passport char(9)) | Шукає всі дані про транспорт(всі поля, але замість номеру паспорту ПІБ та місце проживання). |
| get\_property\_ownership\_certificate(in passport char(9)) | Шукає всі дані про нерухомість (всі поля, але замість номеру паспорту володаря та реєстратора ПІБ та КПП). |

## 3.4 Робота над звітами

Розробка звітів реалізовувалася за допомоги розширення Microsoft RDLC Report Designer для WindowsForms[7];

Всі звіти являють собою документи (паспорт, КПП, свідоцтво про РМП, СПН, свідоцтво про реєстрацію транспорту, свідоцтво про право власності на нерухоме майно, свідоцтво про шлюб) ;

Звіти формуються, коли користувач взаємодіє в формою PersonalForm, що має поле типу User, яке окрім властивостей Username, Password та Permission, має ще PassportId;

Алгоритм формування звіту на прикладі формування паспорту;

* Після натискання кнопки «Паспорт» на формі відбувається звернення до методу ReadPassportForReport класу Database з передачею номеру паспорту як параметру;
* Метод ReadPassportForReport за допомоги методу CreateReaderForPassportReport створює об’єкт reader типу MySqlDataReader;
* Метод CreateReaderForPassportReport класу Database викликає вбудовану процедуру get\_passport, передаючи їй параметр passport\_id та повертає об’єкт типу MySqlDataReader;
* Потім метод ReadPassportForReport перевіряє чи не дорівнює reader null? Якщо так то кидає ексепшен з текстом «Reader is not set». Якщо reader не має рядків, то кидає ексепшен з текстом «Нічого не знайдено» ;
* Нарешті метод ReadPassportForReport зчитує повернутий рядок (він завжди один), та упаковує його в List<string>, де кожен string це певний параметр звіту;
* Список параметрів звіту повертається до форми PersonalForm, що створює форму ReportPassportForm, передаючи їй список параметрів;
* Форма ReportPassportForm має елемент типу ReportViewer, до якого під’єднано шаблон Passport.rdlc. В методі ReportPassportForm\_Load кожному параметру звіту задається значення зі списку параметрів, що був переданий в конструктор;
* І нарешті, коли форма ReportPassportForm створена, форма PersonalForm її показує.

На рисунках 3.3 – 3.9 зображено шаблони всіх звітів.

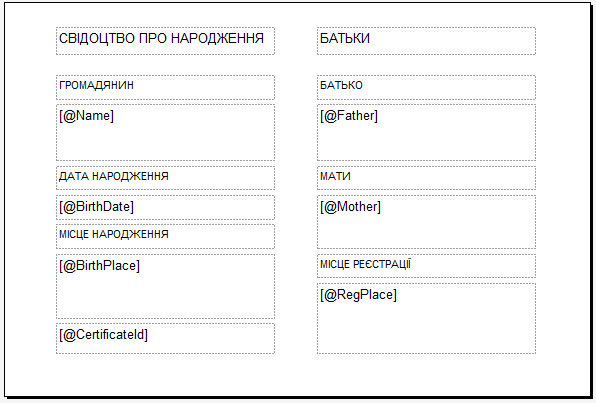


Рисунок 3.3 – Шаблон звіту свідоцтва про народження.

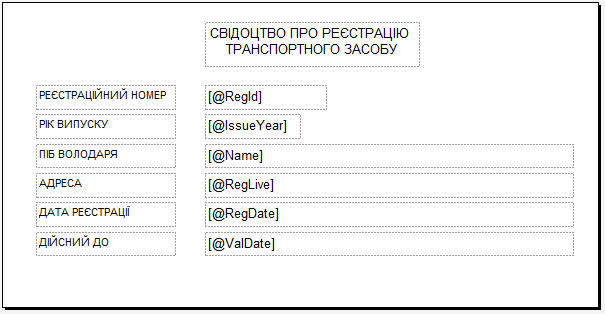


Рисунок 3.4 – Шаблон звіту свідоцтва про реєстрацію транспорту

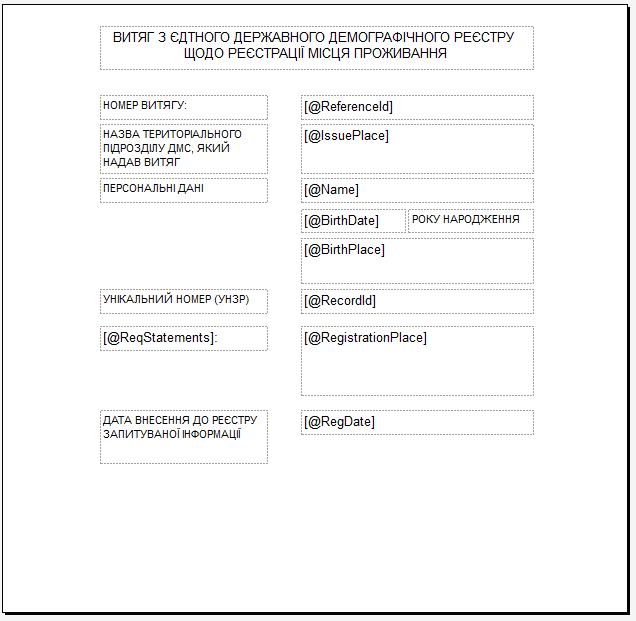


Рисунок 3.5 – Шаблон звіту свідоцтва про реєстрацію місця проживання

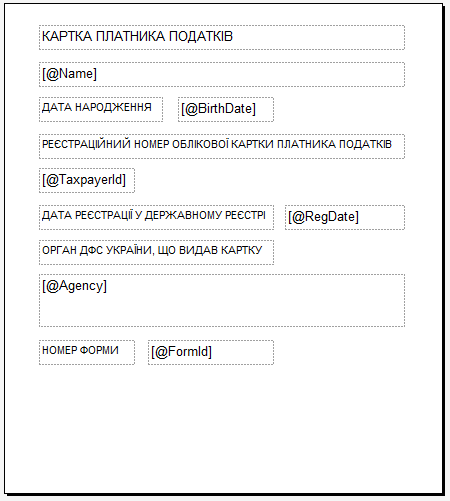


Рисунок 3.6 – Шаблон звіту картки платника податків

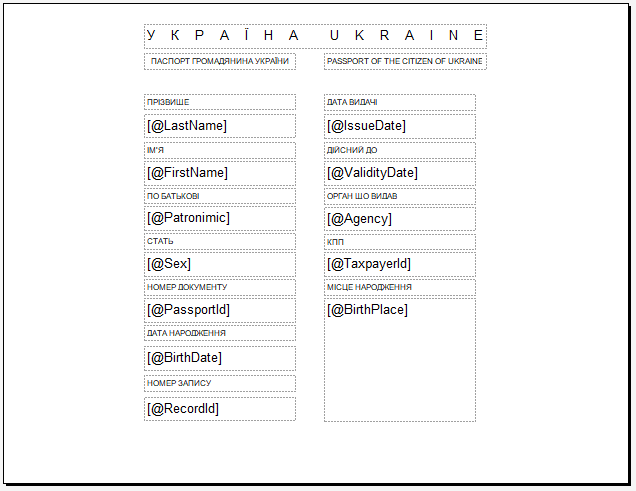


Рисунок 3.7 – Шаблон звіту паспорту

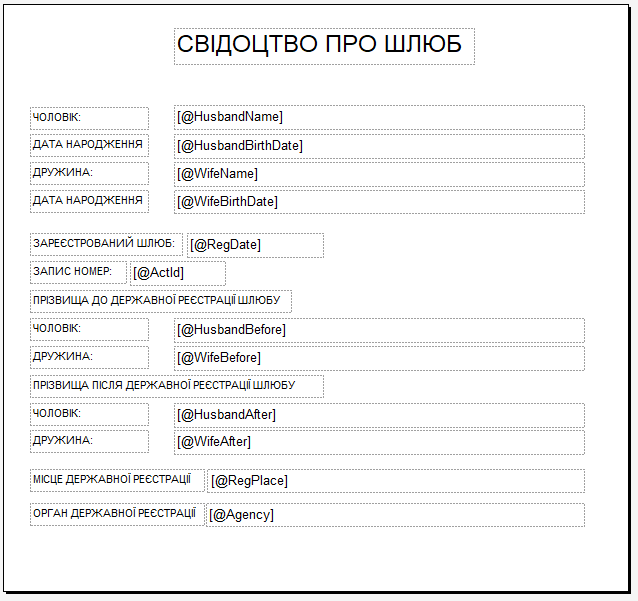


Рисунок 3.8 – Шаблон звіту свідоцтва про шлюб

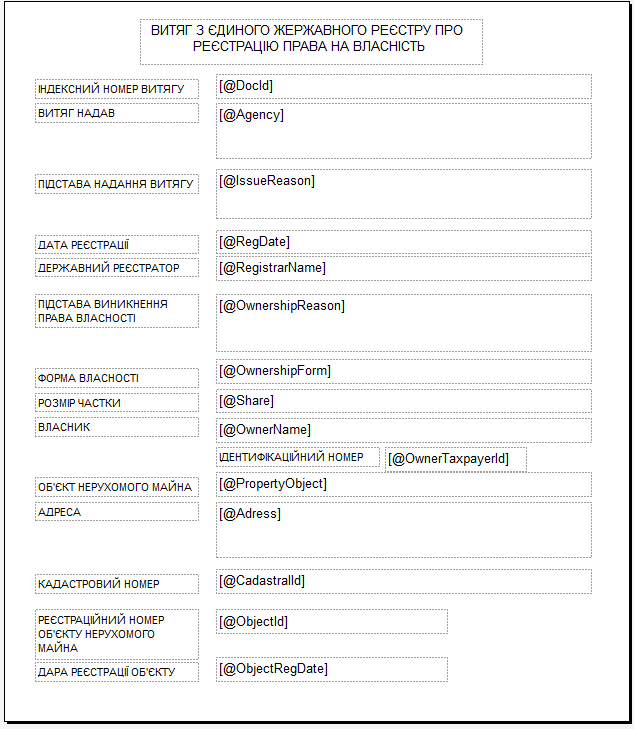


Рисунок 3.9 – Шаблон звіту довідки про реєстрацію нерухомого майна

# 4 ПРИЙНЯТІ РІШЕННЯ ПІД ЧАС РЕАЛІЗАЦІЇ СИСТЕМИ

Під час реалізації програмної системи було прийнято рішення реалізувати всі запити до БД через виклик вбудованих процедур. Причому виклики процедур з заданням параметрів реалізовані в класі Database, який реалізує абсолютно всю взаємодію з БД, в тому числі і відкриття з’єднання.

Опис процедур, що викликаються застосунком (класом Database) наведено в розділі 3.3.

Код всіх розроблених процедур та функцій наведено в додатку Б.

Код класу Database наведено в додатку А, в розділі А.2.

Нижче наведено таблицю 4.1 з описом всіх методів класу Database

Таблиця 4.1 – Методи класу Database

|  |  |
| --- | --- |
| Назва методу і параметри | Опис методу |
| public Database() | Конструктор, нічого не робить |
| public Database(MySqlConnection connection) | Конструктор, приймає параметр типу MySqlConnection, який зберігається в єдиному полі класу, та використовується для відкриття з’єднання перед тим як викликати процедури. |
| public void Close() | Закриває зв’язок з БД |
| public void Open(MySqlConnection connection) | Відкриває зв’язок, що передається як параметр. |
| public bool isConnected() | Перевіряє чи не дорівнює поле \_connеction null. |
| public void SignUp(User user) | Викликає процедуру sign\_up\_user та передає в якості параметрів логін, пароль, рівень дозволу та номер паспорту. |

Продовження таблиці 4.1

|  |  |
| --- | --- |
| public User SignIn(User user) | Викликає процедуру sign\_in\_user та передає в якості параметрів логін та пароль, а приймає повернені параметри рівня дозволу та номеру паспорту. |
| public void RecordBirth(Dictionary<string, string> values) | Викликає процедуру record\_birth та передає в якості параметрів значення текстбоксів з групи «Народження» форми CreateForm |
| public void RecordTaxpayer(Dictionary<string, string> values) | Викликає процедуру record\_taxpayer\_certificate та передає в якості параметрів значення текстбоксів з групи «КПП» форми CreateForm |
| public void RecordLiveCertificate(Dictionary<string, string> values) | Викликає процедуру record\_reference\_from\_usr та передає в якості параметрів значення текстбоксів з групи «Проживання» форми CreateForm |
| public void RecordPassport(Dictionary<string, string> values) | Викликає процедуру record\_passport та передає в якості параметрів значення текстбоксів з групи «Паспорт» форми CreateForm |
| public void RecordMarriage(Dictionary<string, string> values) | Викликає процедуру record\_marriage та передає в якості параметрів значення текстбоксів з групи «Шлюб» форми CreateForm |

Продовження таблиці 4.1

|  |  |
| --- | --- |
| public void RecordPropertyOwnership(Dictionary<string, string> values) | Викликає процедуру record\_property\_right\_certificate та передає в якості параметрів значення текстбоксів з групи «Володіння» форми CreateForm |
| public void RecordCarOwnership(Dictionary<string, string> values) | Викликає процедуру record\_transport\_registration\_certificate та передає в якості параметрів значення текстбоксів з групи «Транспорт» форми CreateForm |
| public void RecordDeath(Dictionary<string, string> values) | Викликає процедуру record\_death та передає в якості параметрів значення текстбоксів з групи «Смерть» форми CreateForm |
| public MySqlDataReader CreateReaderOnPassportData(Dictionary<string, string> values) | Викликає процедуру find\_passport\_data та передає в якості параметрів значення текстбоксів з вкладки «Паспортні дані» форми SearchForm та повертає MySqlDataReader |
| public MySqlDataReader CreateReaderOnTransport(string registrationId) | Викликає процедуру find\_car\_by\_registration\_id та передає в якості параметрів значення реєстраційного номеру транспорту з вкладки «Транспорт» форми SearchForm та повертає MySqlDataReader |

Продовження таблиці 4.1

|  |  |
| --- | --- |
| public MySqlDataReader CreateReaderOnProperty(string DocumentId) | Викликає процедуру find\_property\_by\_owner\_passport\_id та передає в якості параметрів значення номеру паспорту з вкладки «Нерухомість» форми SearchForm та повертає MySqlDataReader |
| public MySqlDataReader CreateReaderOnOffspring(string DocumentId) | Викликає процедуру find\_children\_by\_dead\_parent та передає в якості параметрів значення номеру свідоцтва про смерть з вкладки «Свідоцтво про смерть» форми SearchForm та повертає MySqlDataReader |
| public MySqlDataReader CreateReaderForPassportReport(string PassportId) | Викликає процедуру get\_passport та передає в якості параметрів значення номеру паспорту з PersonalForm та повертає MySqlDataReader |
| public MySqlDataReader CreateReaderForTaxpayerReport(string PassportId) | Викликає процедуру get\_taxpayer\_certificate та передає в якості параметрів значення номеру паспорту з PersonalForm та повертає MySqlDataReader |
| public MySqlDataReader CreateReaderForReferenceFromUSRReport(string PassportId) | Викликає процедуру get\_reference\_from\_usr та передає в якості параметрів значення номеру паспорту з PersonalForm та повертає MySqlDataReader |

Продовження таблиці 4.1

|  |  |
| --- | --- |
| public MySqlDataReader CreateReaderForBirthCertificateReport(string PassportId) | Викликає процедуру get\_birth\_certificate та передає в якості параметрів значення номеру паспорту з PersonalForm та повертає MySqlDataReader |
| public MySqlDataReader CreateReaderForMarriageCertificateReport(string PassportId) | Викликає процедуру get\_marriage\_certificate та передає в якості параметрів значення номеру паспорту з PersonalForm та повертає MySqlDataReader |
| public MySqlDataReader CreateReaderForPropertyRightCertificateReport(string PassportId) | Викликає процедуру get\_property\_ownership\_certificate та передає в якості параметрів значення номеру паспорту з PersonalForm та повертає MySqlDataReader |
| public MySqlDataReader CreateReaderForTransportCertificateReport(string PassportId) | Викликає процедуру get\_transport\_certificate та передає в якості параметрів значення номеру паспорту з PersonalForm та повертає MySqlDataReader |
| public List<string[]> ReadTransportCertificateForReport(string PassportId) | Створює MySqlDataReader за допомоги методу CreateReaderForTransportCertificateReport  Та укомплктовує всі колонки в string[], і всі рядки List<string[]>, повертає список. |

Продовження таблиці 4.1

|  |  |
| --- | --- |
| public List<string[]> ReadPropertyRightCertificateForReport(string PassportId) | Створює MySqlDataReader за допомоги методу CreateReaderForPropertyRightCertificateReport  Та укомплктовує всі колонки в string[], і всі рядки List<string[]>, повертає список. |
| public List<string> ReadMarriageCertificateForReport(string PassportId) | Створює MySqlDataReader за допомоги методу CreateReaderForMarriageCertificateReport  Та укомплктовує всі колонки в List<string> (рядок завжди один), повертає список. |
| public List<string> ReadBirthCertificateForReport(string PassportId) | Створює MySqlDataReader за допомоги методу CreateReaderForBirthCertificateReport  Та укомплктовує всі колонки в List<string> (рядок завжди один), повертає список. |
| public List<string> ReadReferenceFromUSRForReport(string PassportId) | Створює MySqlDataReader за допомоги методу CreateReaderForReferenceFromUSRReport  Та укомплктовує всі колонки в List<string> (рядок завжди один), повертає список. |
| public List<string> ReadTaxpayerCertificateForReport(string PassportId) | Створює MySqlDataReader за допомоги методу CreateReaderForTaxpayerReport  Та укомплктовує всі колонки в List<string> (рядок завжди один), повертає список. |

Продовження таблиці 4.1

|  |  |
| --- | --- |
| public List<string> ReadPassportForReport(string PassportId) | Створює MySqlDataReader за допомоги методу CreateReaderForPassportReport  Та укомплктовує всі колонки в List<string> (рядок завжди один), повертає список. |
| public List<string[]> ReadOffspringData(string deathDocumentId) | Створює MySqlDataReader за допомоги методу ReadOffspringData  Та укомплктовує всі колонки в string[], і всі рядки List<string[]>, повертає список. |
| public List<string[]> ReadPropertyData(string propertyDocumentId) | Створює MySqlDataReader за допомоги методу CreateReaderOnProperty  Та укомплктовує всі колонки в string[], і всі рядки List<string[]>, повертає список. |
| public List<string[]> ReadTransportData(string registrationId) | Створює MySqlDataReader за допомоги методу CreateReaderOnTransport  Та укомплктовує всі колонки в string[], і всі рядки List<string[]>, повертає список. |
| public List<string[]> ReadPassportData(Dictionary<string, string> values) | Створює MySqlDataReader за допомоги методу CreateReaderOnPassportData  Та укомплктовує всі колонки в string[], і всі рядки List<string[]>, повертає список. |

# 5 керівництво ОПЕРАТОРа

Під час запуску програми бачимо головну форму (рисунок 5.1). Вона дозволяє увійти в існуючий обліковий запис або створити новий ввівши ім’я користувача, пароль та номер паспорту.

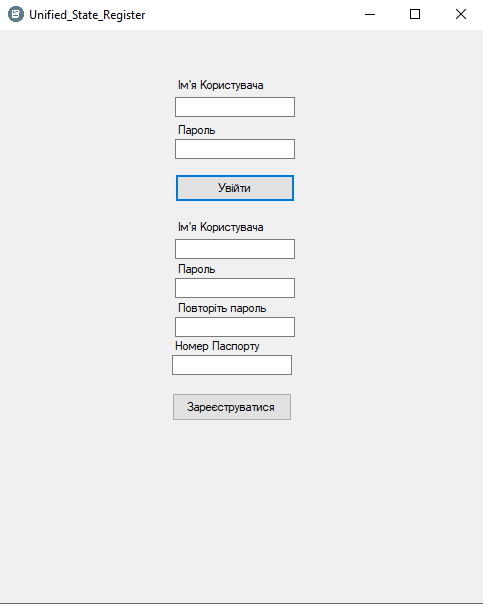


Рисунок 5.1 – Головна форма.

## 5.1 Оформлення документів

Щоб увійти в обліковий запис для оформлення документів треба ввести логін та пароль користувача з рівнем доступу «С». На даний момент в БД існує такий користувач з логіном 4321 та паролем 43214321.

Після авторизації з’являється форма оформлення документів, зображена на рисунку 5.2.

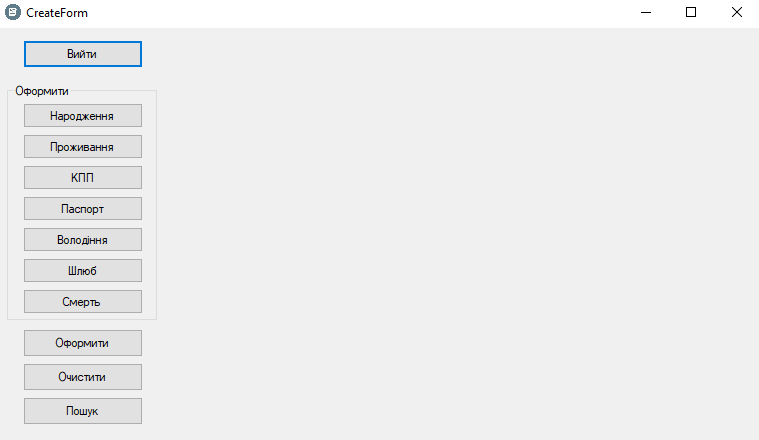


Рисунок 5.2 – Форма оформлення документів.

Щоб оформити будь який документ, треба натиснути на відповідну кнопку з групи «Оформити».

Оформлення документів треба робити в такій послідовності:

* Народження;
* Проживання, КПП;
* Паспорт;
* Все інше(Смерть може бути оформлена будь-коли).

На рисунку 5.3 зображено приклад оформлення народження дитини.

Над деякими полями поруч з назвою в дужках є число, що відповідає кількості чисел, з яких повинен складатися номер.

Поля «Країна», «Область» «Населений пункт» обов’язково заповнювати капсом.

Після заповнення полів треба натиснути кнопку «Оформити». Оформлення завершилося успішно, якщо не виникло жодних меседжбоксів та всі текстбокси очищені.

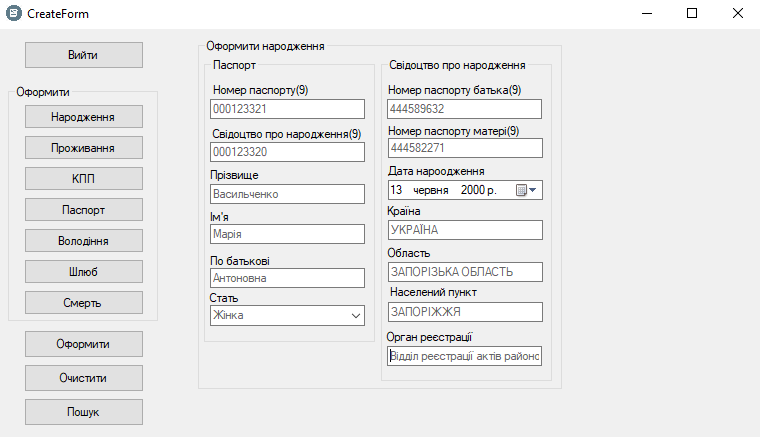


Рисунок 5.3 – Оформлення народження дитини.

На рисунку 5.4 зображено приклад реєстрації місця проживання.

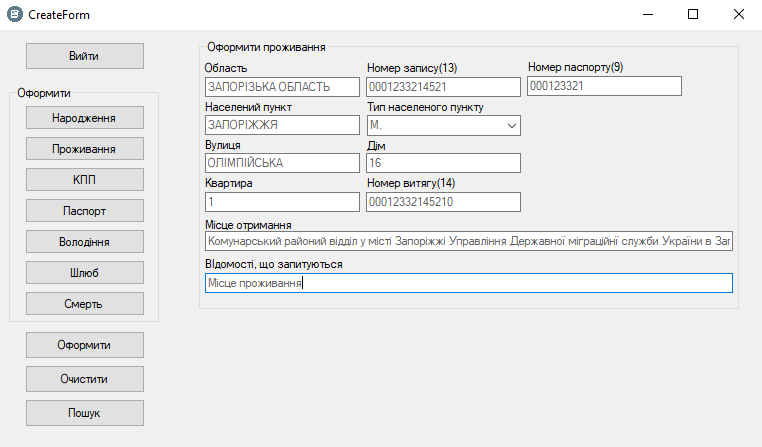


Рисунок 5.4 – Реєстрація місця проживання

На рисунку 5.5 зображено приклад оформлення КПП.

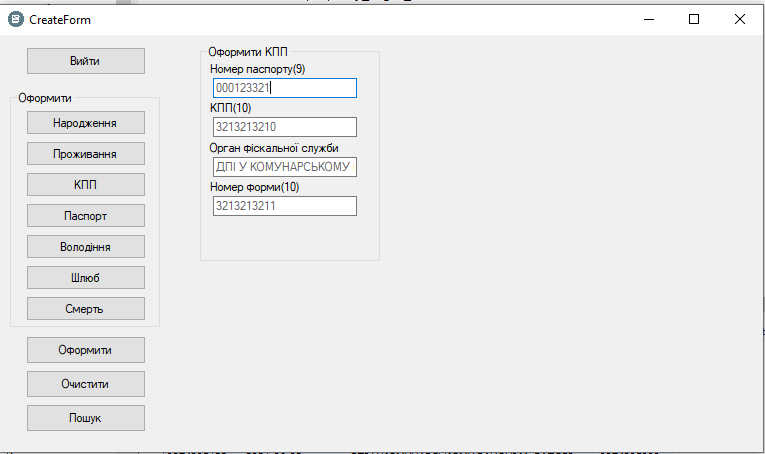


Рисунок 5.5 – Оформлення КПП

На рисунку 5.6 наведено приклад оформлення паспорту.

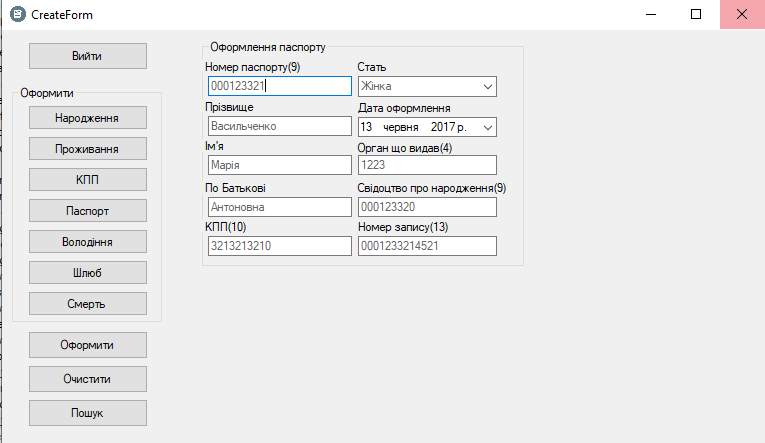


Рисунок 5.6 – Оформлення паспорту

На рисунку 5.7 наведено приклад оформлення нерухомості. Для поля «Номер паспорту реєстратора» використовується номер «000111222». Ця людина існує в БД саме для цієї ролі.

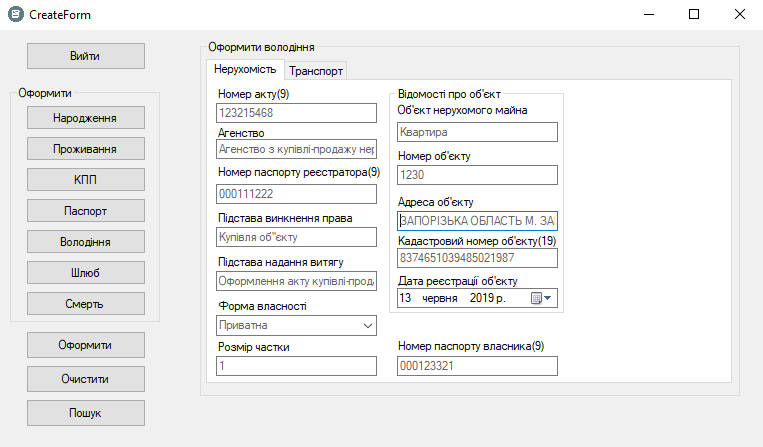


Рисунок 5.7 – Оформлення нерухомого майна

На рисунку 5.8 наведено приклад оформлення транспортного засобу. В поле «Реєстраційний номер» слід вводити 8 знаків, перші 2 та останні 2 з них – латинські літери.

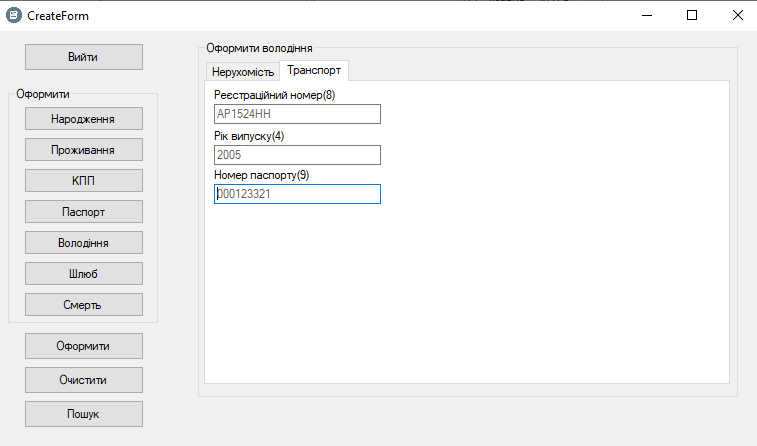


Рисунок 5.8 – Оформлення транспортного засобу

На рисунку 5.9 наведено приклад оформлення шлюбу.

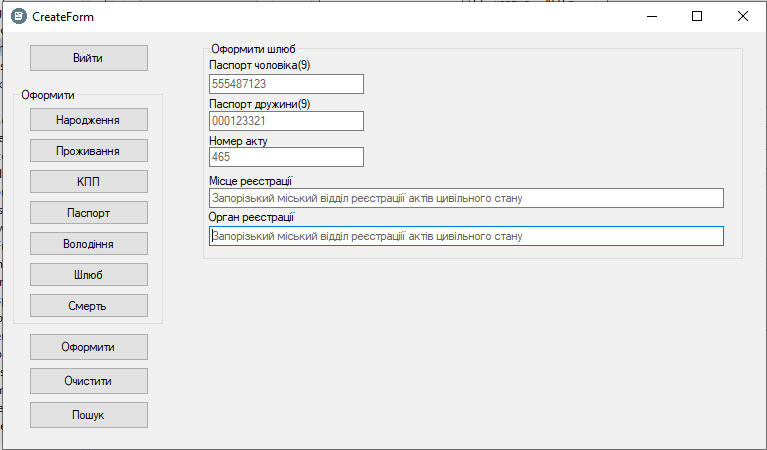


Рисунок 5.9 – Оформлення шлюбу

Після оформлення шлюбу, дружині, слід оформити паспорт з прізвищем чоловіка(за бажанням).

На рисунку 5.10 наведено приклад оформлення смерті(в даному випадку матері цієї громадянки)

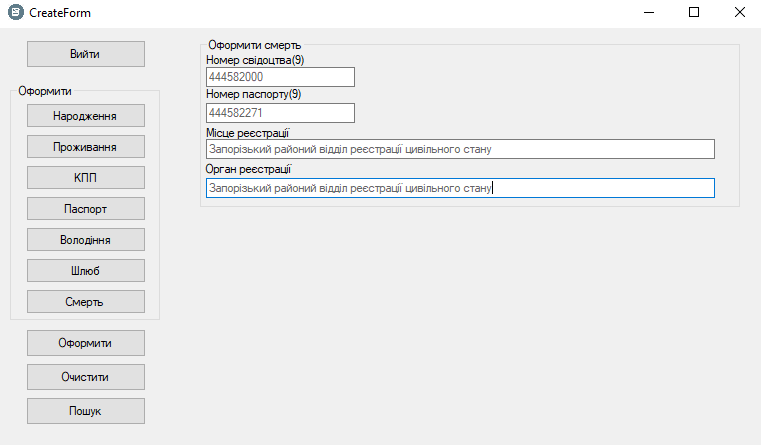


Рисунок 5.10 – Оформлення смерті

## 5.2 Пошук в базі даних

Для повернення на головну форму, треба натиснути кнопку «Вийти» та увійти в обліковий запис з рівнем доступу «А». Такий обліковий запис в БД має логін 1234 та пароль 12341234. Або одразу на формі оформлення документів натиснути кнопку «Пошук». Під час оформлення документів, щоб не занотовувати номери різних документів, які потрібні в оформлені один одного, можна використовувати форму пошуку, для копіювання вже записаних даних.

Форма пошуку зображена на рисунку 5.11.

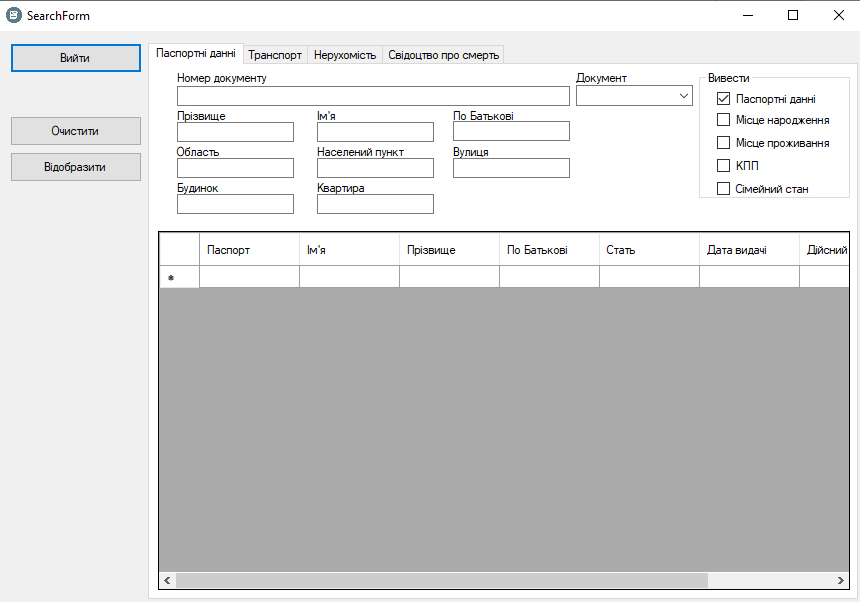


Рисунок 5.11 – Форма пошуку

Перша вкладка форми дозволяє виконувати пошук за паспортними даними, заповнивши потрібні поля певними даними.

Не обов’язково всі полі мають бути заповнені. Не обов’язково в полях писати повну інформацію(тобто повну назву міста, прізвище, тощо).

При введені номеру документу має бути обраний сам документ у комбобоксі поруч.

Кожний чекбокс з групи «Вивести» задає значення видимості певній групі колок таблиці з результатами.

Якщо залишити всі поля порожніми, то будуть відображені абсолютно всі записи, що наявні в БД.

Якщо в рядку відсутні колонки паспортних даних «Дата видачі», «Дійсний до» та «Орган», то це ознаає, что паспорт ще не був оформлений.

Приклад пошуку даних наведено на рисунку 5.12.

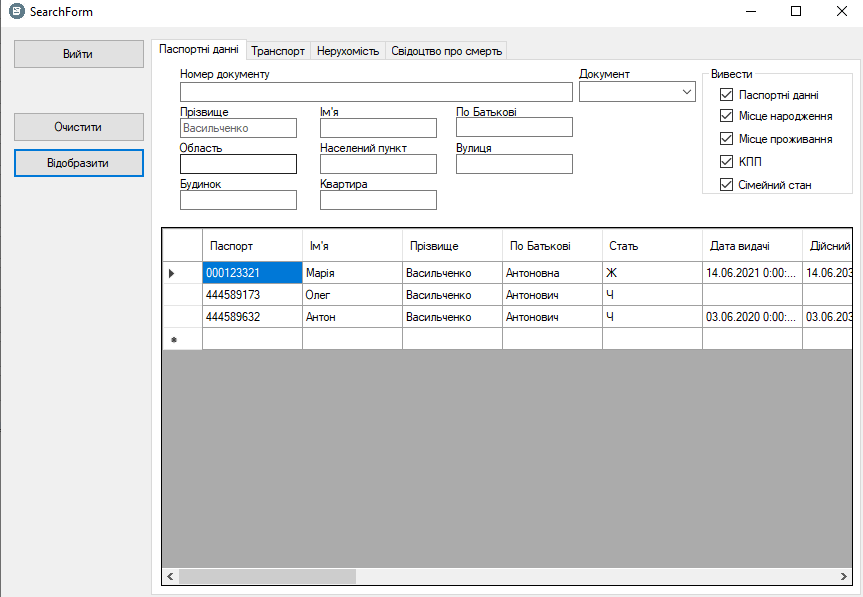


Рисунок 5.13 – Пошук за прізвищем

Приклад пошуку транспорту за реєстраційним номером наведено на рисунку 5.14.

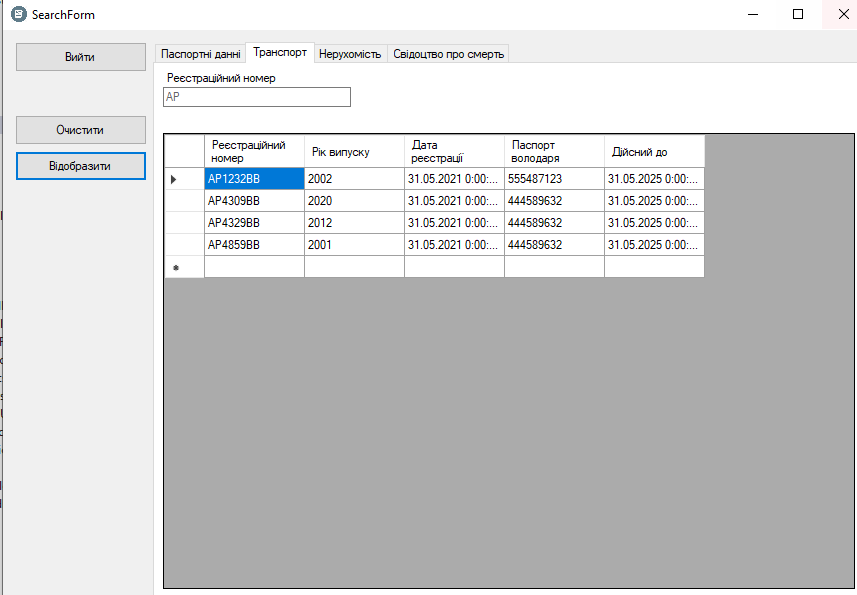


Рисунок 5.14 – Пошук транспорту за реєстраційним номером

Приклад пошуку нерухомості за паспортом володаря наведено на рисунку 5.15.

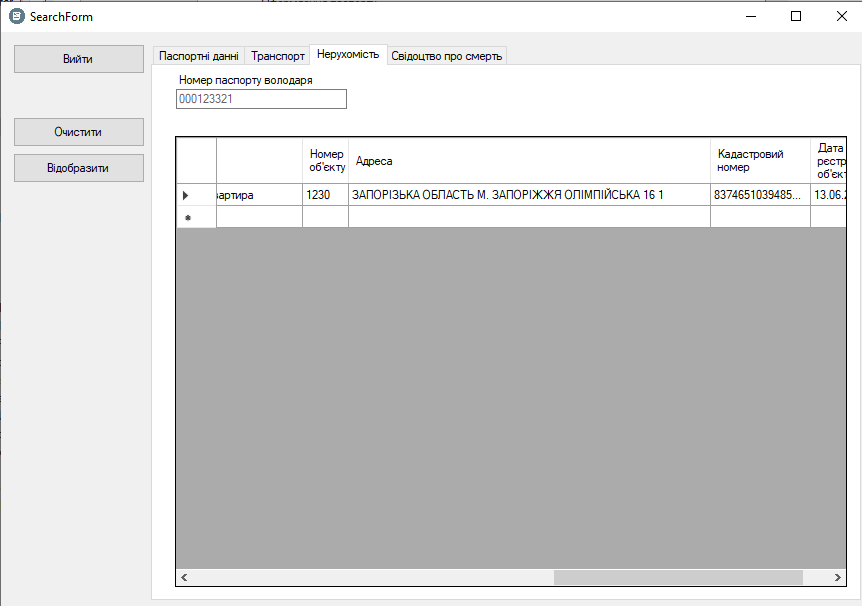


Рисунок 5.15 – Приклад пошуку нерухомості за паспортом

Пошук нащадків за номером свідоцтва про смерть наведено на рисунку 5.16.

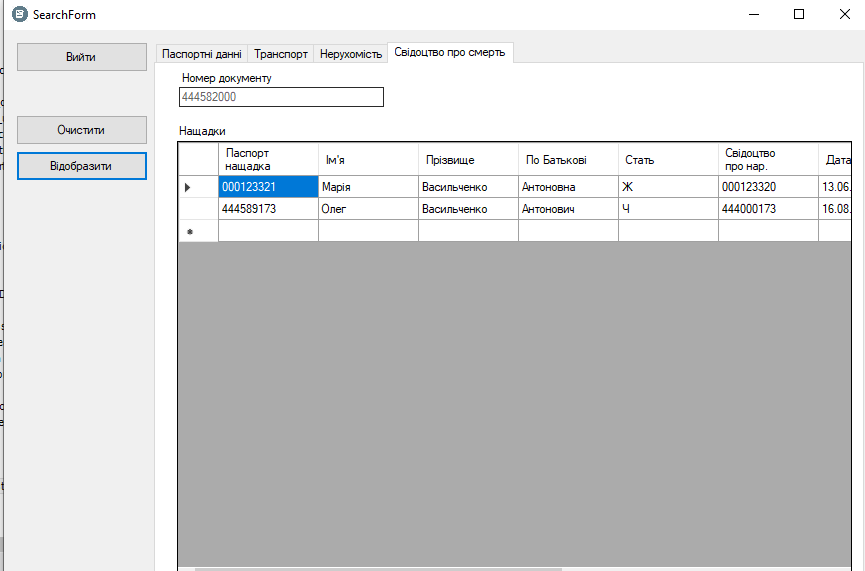


Рисунок 5.16 – Пошук нащадків за номером свідоцтва про смерть.

## 5.3 Перегляд власних документів

Для того, щоб переглянути власні документи, громадянину треба створити обліковий запис зі своїм паспортом, який вже є в реєстрі.

На рисунку 5.17 наведено створення облікового запису громадянки, чиї документи були створені в розділі 5.1

Ім’я користувача: user2

Пароль: 11111111

Далі треба авторизуватися.

Після авторизації з’явиться персональна форма, наведена на рисунку 5.18.

При натисканні на будь-яку кнопку з’являєтся нова форма зі звітом, що являє собою відповідний документ. Якщо документ не оформленно, з’явиться відповідне повідомлення

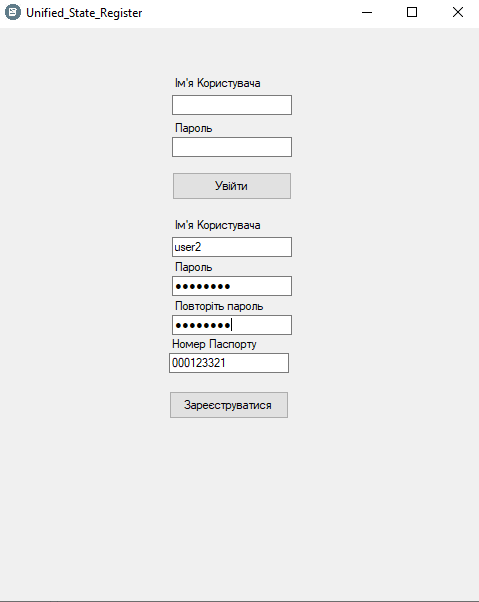


Рисунок 5.17 – Створення особистого кабінету

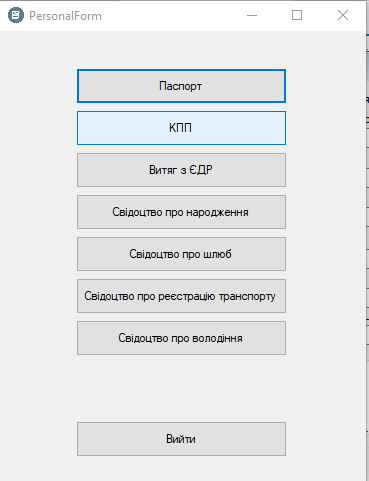


Рисунок 5.18 – Персональна форма

На рисунках 5.19 – 5.25 наведено звіти, які є документами громадянки Васильченко Марії Антоновни, що були оформлені в розділі 5.1.

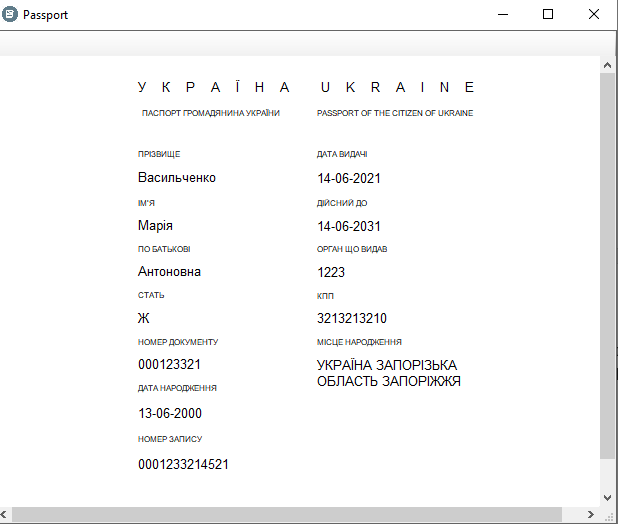


Рисунок 5.19 – Паспорт

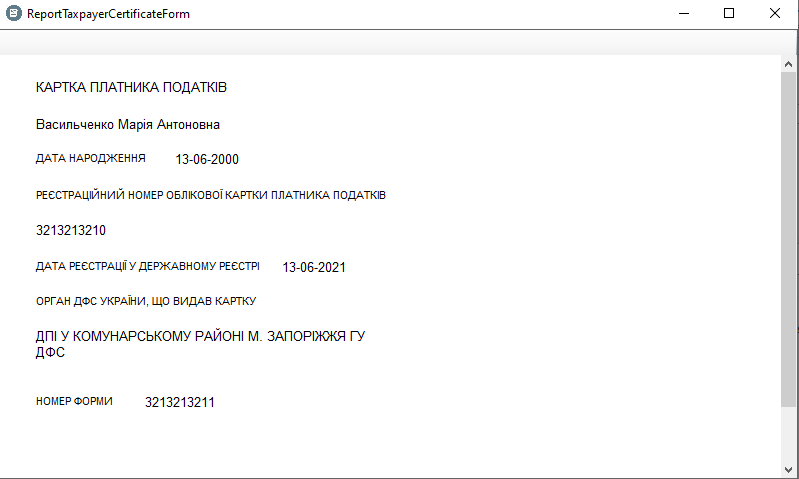


Рисунок 5.20 – КПП

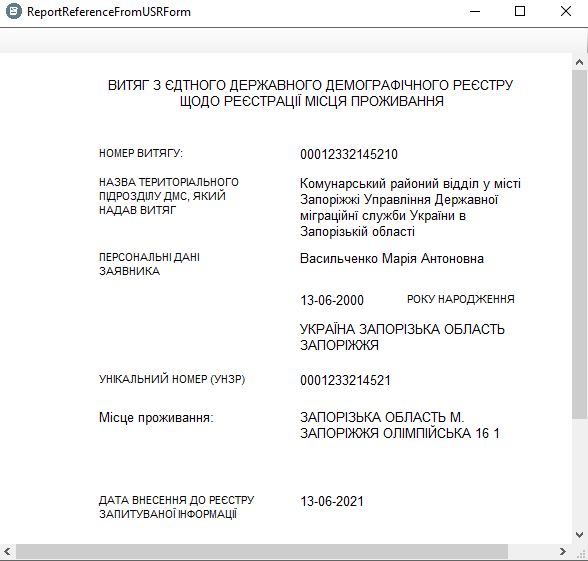


Рисунок 5.21 – Витяг з ЄДР (РМП)

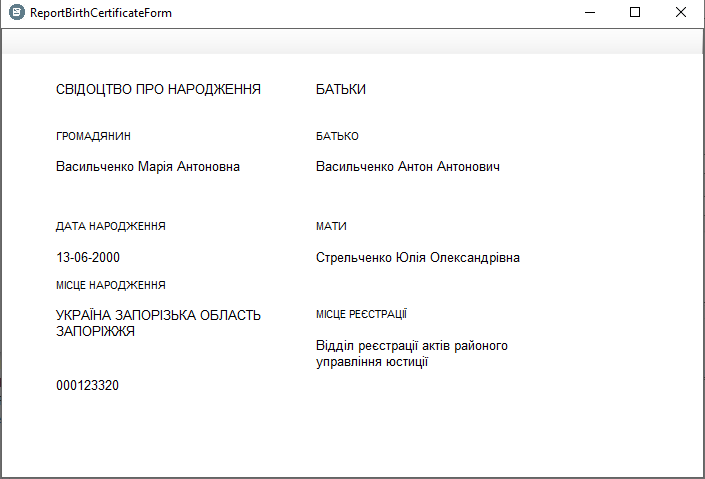


Рисунок 5.22 – СПН

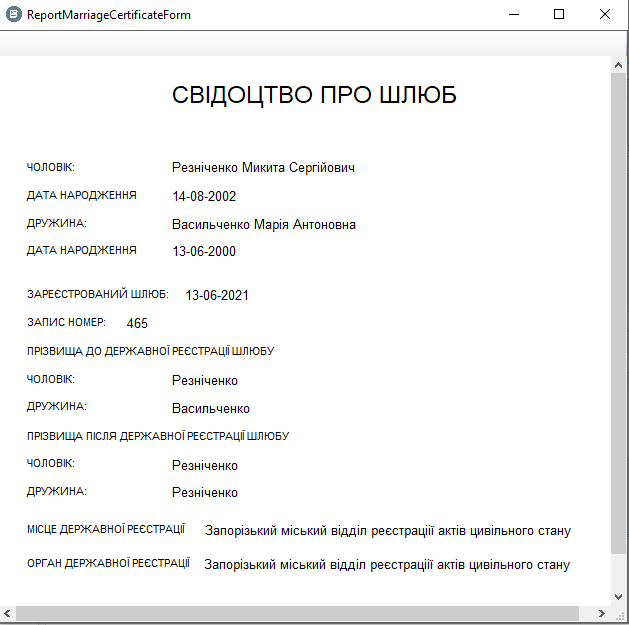


Рисунок 5.23 – Свідоцтво про шлюб(Марія Антоновна, ще не оформила новий паспорт, тому моки що вона Васильченко, а не Резніченко)

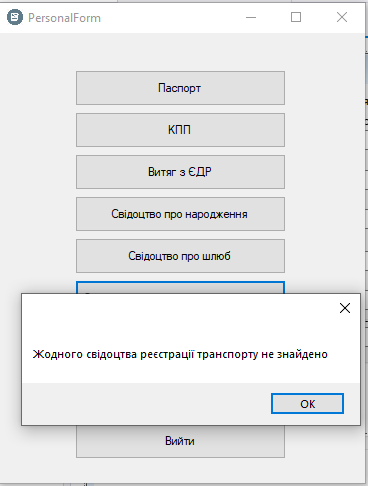


Рисунок 5.24 – Свідоцтво про реєстрацію транспорту. (А ось транспорту в неї немає, тому що в розділі 5.1 я забув натиснути кнопку «Оформити»)

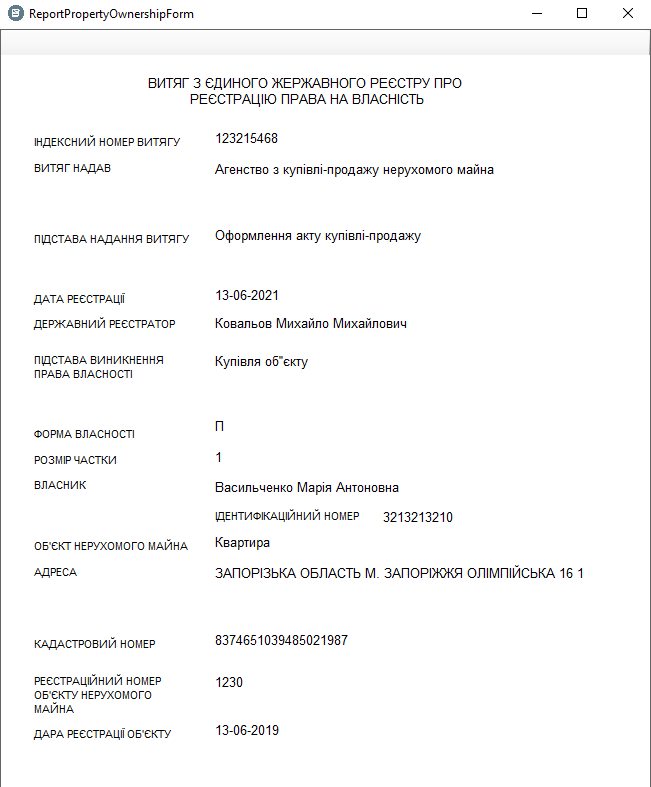


Рисунок 5.25 – Свідоцтво про реєстрацію нерухомого майна

# ВИСНОВКИ

Розроблена програмна система в курсовому проекті є насамперед демонстрацією моїх навичок та розуміння роботи з базами даних. Для розробки програмного застосунку використовувалася платформа .Net та мова програмування C#.

Для розробки графічного інтерфейсу використовувалися компоненти WindowsForms.

Для розробки бази даних використовувалася СКБД MySQL.

Для налагодження взаємодії маж застосунком та СКБД використовувалися класи компоненту MySQLConnector-Net.

Для розробки звітів використовувалося розширення Microsoft RDLC Report Designer.

Загалом було створено 17 таблиць, 1 подання, 28 вбудованих процедур, 15 функцій, 1 клас взаємодії з БД, 11 форм (7 з яких для відображення звітів), 7 звітів.

# ПЕРЕЛІК ПОСИЛАНЬ

1. Що таке Дія [Eлектрон. ресурс]. – Режим доступу: <https://plan2.diia.gov.ua>.
2. eGovernment [Electronic Resource] – Access: <https://digital-strategy.ec.europa.eu/en/policies/egovernment>.
3. MySQL [Electronic Resource] – Access: <https://dev.mysql.com/doc/>.
4. C Sharp [Electronic Resource] – Access: <https://ru.wikipedia.org/wiki/C_Sharp>.
5. Microsoft RDLC Report Designer [Electronic Resource] – Access:

<https://marketplace.visualstudio.com/items?itemName=ProBITools.MicrosoftRdlcReportDesignerforVisualStudio-18001>.

1. Choosing Between Views, Functions, and Stored Procedures in SQL With Exaples [Electronic Resource] – Access: <https://www.youtube.com/watch?v=TSCPXpXL4OI&list=LL&index=4&t=146s>.
2. Пример создания отчета в приложении. Элемент управления ReportViewer. База даних Microsoft Access [Eлектрон. ресурс]. – Режим доступу: <https://www.bestprog.net/ru/2017/12/26/an-example-of-creating-a-report-in-an-application-created-using-the-windows-forms-application-template-the-microsoft-access-database-mdb_ru/>.

# ДОДАТОК А КОД ПРОГРАМИ

## А.1 Клас Program

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Unified\_State\_Register

{

static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Database DB;

try

{

DB = new Database(DBUtils.GetDBConnection());

}

catch (Exception e)

{

MessageBox.Show("Error: " + e.Message);

return;

}

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new MainForm(DB));

}

}

}

## А.2 Клас Database

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data;

using MySql.Data.MySqlClient;

namespace Unified\_State\_Register

{

public class Database

{

private MySqlConnection \_connection;

public Database()

{

\_connection = null;

}

public Database(MySqlConnection connection)

{

\_connection = null;

Open(connection);

}

public void Close()

{

if (\_connection != null)

{

\_connection.Close();

\_connection = null;

}

}

public void Open(MySqlConnection connection)

{

Close();

\_connection = connection;

\_connection.Open();

}

public bool isConnected()

{

if (\_connection == null)

{

return false;

}

else

{

return true;

}

}

public void SignUp(User user)

{

MySqlCommand cmd = new MySqlCommand("sign\_up\_user", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("username", MySqlDbType.VarChar, 12);

cmd.Parameters.Add("user\_password", MySqlDbType.VarChar, 32);

cmd.Parameters.Add("permission", MySqlDbType.VarChar, 1);

cmd.Parameters.Add("passport\_id", MySqlDbType.VarChar, 9);

cmd.Parameters["username"].Value = user.Username;

cmd.Parameters["user\_password"].Value = user.Password;

cmd.Parameters["permission"].Value = user.Permission;

cmd.Parameters["passport\_id"].Value = user.PassportId;

cmd.Parameters["username"].Direction = ParameterDirection.Input;

cmd.Parameters["user\_password"].Direction = ParameterDirection.Input;

cmd.Parameters["permission"].Direction = ParameterDirection.Input;

cmd.Parameters["passport\_id"].Direction = ParameterDirection.Input;

cmd.ExecuteNonQuery();

}

public User SignIn(User user)

{

MySqlCommand cmd = new MySqlCommand("sign\_in\_user", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("username", MySqlDbType.VarChar, 12);

cmd.Parameters.Add("user\_password", MySqlDbType.VarChar, 32);

cmd.Parameters.Add("passport\_id", MySqlDbType.VarChar, 9);

cmd.Parameters.Add("permission", MySqlDbType.VarChar, 1);

cmd.Parameters["username"].Value = user.Username;

cmd.Parameters["user\_password"].Value = user.Password;

cmd.Parameters["username"].Direction = ParameterDirection.Input;

cmd.Parameters["user\_password"].Direction = ParameterDirection.Input;

cmd.Parameters["passport\_id"].Direction = ParameterDirection.Output;

cmd.Parameters["permission"].Direction = ParameterDirection.Output;

cmd.ExecuteNonQuery();

if (cmd.Parameters["passport\_id"].Value == DBNull.Value)

{

return new User(user.Username,

user.Password,

null,

Convert.ToChar(cmd.Parameters["permission"].Value));

}

return new User(user.Username,

user.Password,

Convert.ToString(cmd.Parameters["passport\_id"].Value),

Convert.ToChar(cmd.Parameters["permission"].Value));

}

public void RecordBirth(Dictionary<string, string> values)

{

MySqlCommand cmd = new MySqlCommand("record\_birth", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("passport\_id", MySqlDbType.VarChar, 9);

cmd.Parameters.Add("place\_of\_reg", MySqlDbType.VarChar, 255);

cmd.Parameters.Add("birth\_cert\_id", MySqlDbType.VarChar, 9);

cmd.Parameters.Add("first\_name", MySqlDbType.VarChar, 30);

cmd.Parameters.Add("last\_name", MySqlDbType.VarChar, 30);

cmd.Parameters.Add("patronimic", MySqlDbType.VarChar, 30);

cmd.Parameters.Add("passport\_m", MySqlDbType.VarChar, 9);

cmd.Parameters.Add("passport\_f", MySqlDbType.VarChar, 9);

cmd.Parameters.Add("date\_of\_birth", MySqlDbType.Date);

cmd.Parameters.Add("sex", MySqlDbType.VarChar, 1);

cmd.Parameters.Add("place\_of\_birth", MySqlDbType.VarChar, 255);

cmd.Parameters["passport\_id"].Value = values["passport"];

cmd.Parameters["place\_of\_reg"].Value = values["agency"];

cmd.Parameters["birth\_cert\_id"].Value = values["birth\_cert"];

cmd.Parameters["first\_name"].Value = values["first\_name"];

cmd.Parameters["last\_name"].Value = values["last\_name"];

cmd.Parameters["patronimic"].Value = values["patronimic"];

cmd.Parameters["passport\_m"].Value = values["mother"];

cmd.Parameters["passport\_f"].Value = values["father"];

cmd.Parameters["date\_of\_birth"].Value = Convert.ToDateTime(values["date"]);

cmd.Parameters["sex"].Value = Convert.ToString(values["sex"][0]);

cmd.Parameters["place\_of\_birth"].Value = values["country"] + " " + values["region"] + " " + values["place"];

cmd.Parameters["passport\_id"].Direction = ParameterDirection.Input;

cmd.Parameters["place\_of\_reg"].Direction = ParameterDirection.Input;

cmd.Parameters["birth\_cert\_id"].Direction = ParameterDirection.Input;

cmd.Parameters["first\_name"].Direction = ParameterDirection.Input;

cmd.Parameters["last\_name"].Direction = ParameterDirection.Input;

cmd.Parameters["patronimic"].Direction = ParameterDirection.Input;

cmd.Parameters["passport\_m"].Direction = ParameterDirection.Input;

cmd.Parameters["passport\_f"].Direction = ParameterDirection.Input;

cmd.Parameters["date\_of\_birth"].Direction = ParameterDirection.Input;

cmd.Parameters["sex"].Direction = ParameterDirection.Input;

cmd.Parameters["place\_of\_birth"].Direction = ParameterDirection.Input;

cmd.ExecuteNonQuery();

}

public void RecordTaxpayer(Dictionary<string, string> values)

{

MySqlCommand cmd = new MySqlCommand("record\_taxpayer\_certificate", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("passport\_id", MySqlDbType.VarChar, 9);

cmd.Parameters.Add("taxpayer\_id", MySqlDbType.VarChar, 10);

cmd.Parameters.Add("fiscal\_service\_agency", MySqlDbType.VarChar, 255);

cmd.Parameters.Add("form\_id", MySqlDbType.VarChar, 10);

cmd.Parameters["passport\_id"].Value = values["passport"];

cmd.Parameters["taxpayer\_id"].Value = values["taxpayer"];

cmd.Parameters["fiscal\_service\_agency"].Value = values["agency"];

cmd.Parameters["form\_id"].Value = values["form\_id"];

cmd.Parameters["passport\_id"].Direction = ParameterDirection.Input;

cmd.Parameters["taxpayer\_id"].Direction = ParameterDirection.Input;

cmd.Parameters["fiscal\_service\_agency"].Direction = ParameterDirection.Input;

cmd.Parameters["form\_id"].Direction = ParameterDirection.Input;

if (cmd.Parameters["taxpayer\_id"].Value == DBNull.Value)

throw new Exception("taxpayer null");

cmd.ExecuteNonQuery();

}

public void RecordLiveCertificate(Dictionary<string, string> values)

{

MySqlCommand cmd = new MySqlCommand("record\_reference\_from\_usr", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("passport", MySqlDbType.VarChar, 9);

cmd.Parameters.Add("record", MySqlDbType.VarChar, 13);

cmd.Parameters.Add("reference\_id", MySqlDbType.VarChar, 14);

cmd.Parameters.Add("requested\_statements", MySqlDbType.VarChar, 255);

cmd.Parameters.Add("place\_of\_issue", MySqlDbType.VarChar, 255);

cmd.Parameters.Add("place\_of\_reg", MySqlDbType.VarChar, 255);

cmd.Parameters["passport"].Value = values["passport"];

cmd.Parameters["record"].Value = values["record"];

cmd.Parameters["reference\_id"].Value = values["reference"];

cmd.Parameters["requested\_statements"].Value = values["statements"];

cmd.Parameters["place\_of\_issue"].Value = values["issue"];

cmd.Parameters["place\_of\_reg"].Value = values["region"] + " " + values["place\_type"] + " " + values["place"] + " " + values["street"] + " " + values["house"] + " " + values["appartment"];

cmd.Parameters["passport"].Direction = ParameterDirection.Input;

cmd.Parameters["record"].Direction = ParameterDirection.Input;

cmd.Parameters["reference\_id"].Direction = ParameterDirection.Input;

cmd.Parameters["requested\_statements"].Direction = ParameterDirection.Input;

cmd.Parameters["place\_of\_issue"].Direction = ParameterDirection.Input;

cmd.Parameters["place\_of\_reg"].Direction = ParameterDirection.Input;

cmd.ExecuteNonQuery();

}

public void RecordPassport(Dictionary<string, string> values)

{

MySqlCommand cmd = new MySqlCommand("record\_passport", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("passport\_id", MySqlDbType.VarChar, 9);

cmd.Parameters.Add("first\_name", MySqlDbType.VarChar, 30);

cmd.Parameters.Add("last\_name", MySqlDbType.VarChar, 30);

cmd.Parameters.Add("patronimic", MySqlDbType.VarChar, 30);

cmd.Parameters.Add("sex", MySqlDbType.VarChar, 1);

cmd.Parameters.Add("date\_of\_issue", MySqlDbType.Date);

cmd.Parameters.Add("agency\_id", MySqlDbType.VarChar, 4);

cmd.Parameters.Add("birth\_cert\_id", MySqlDbType.VarChar, 9);

cmd.Parameters.Add("taxpayer\_id", MySqlDbType.VarChar, 10);

cmd.Parameters.Add("record\_id", MySqlDbType.VarChar, 13);

cmd.Parameters["passport\_id"].Value = values["passport"];

cmd.Parameters["first\_name"].Value = values["first\_name"];

cmd.Parameters["last\_name"].Value = values["last\_name"];

cmd.Parameters["patronimic"].Value = values["patronimic"];

cmd.Parameters["sex"].Value = Convert.ToString(values["sex"][0]);

cmd.Parameters["date\_of\_issue"].Value = Convert.ToDateTime(values["date"]);

cmd.Parameters["agency\_id"].Value = values["agency"];

cmd.Parameters["birth\_cert\_id"].Value = values["birth\_cert"];

cmd.Parameters["taxpayer\_id"].Value = values["taxpayer"];

cmd.Parameters["record\_id"].Value = values["place"];

cmd.Parameters["passport\_id"].Direction = ParameterDirection.Input;

cmd.Parameters["first\_name"].Direction = ParameterDirection.Input;

cmd.Parameters["last\_name"].Direction = ParameterDirection.Input;

cmd.Parameters["patronimic"].Direction = ParameterDirection.Input;

cmd.Parameters["sex"].Direction = ParameterDirection.Input;

cmd.Parameters["date\_of\_issue"].Direction = ParameterDirection.Input;

cmd.Parameters["agency\_id"].Direction = ParameterDirection.Input;

cmd.Parameters["birth\_cert\_id"].Direction = ParameterDirection.Input;

cmd.Parameters["taxpayer\_id"].Direction = ParameterDirection.Input;

cmd.Parameters["record\_id"].Direction = ParameterDirection.Input;

cmd.ExecuteNonQuery();

}

public void RecordMarriage(Dictionary<string, string> values)

{

MySqlCommand cmd = new MySqlCommand("record\_marriage", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("passport\_h", MySqlDbType.VarChar, 9);

cmd.Parameters.Add("passport\_w", MySqlDbType.VarChar, 9);

cmd.Parameters.Add("reg\_place", MySqlDbType.VarChar, 255);

cmd.Parameters.Add("agency", MySqlDbType.VarChar, 255);

cmd.Parameters.Add("act\_id", MySqlDbType.Int32);

cmd.Parameters["passport\_h"].Value = values["husband"];

cmd.Parameters["passport\_w"].Value = values["wife"];

cmd.Parameters["reg\_place"].Value = values["place"];

cmd.Parameters["agency"].Value = values["agency"];

cmd.Parameters["act\_id"].Value = Convert.ToInt32(values["act"]);

cmd.Parameters["passport\_h"].Direction = ParameterDirection.Input;

cmd.Parameters["passport\_w"].Direction = ParameterDirection.Input;

cmd.Parameters["reg\_place"].Direction = ParameterDirection.Input;

cmd.Parameters["agency"].Direction = ParameterDirection.Input;

cmd.Parameters["act\_id"].Direction = ParameterDirection.Input;

cmd.ExecuteNonQuery();

}

public void RecordPropertyOwnership(Dictionary<string, string> values)

{

MySqlCommand cmd = new MySqlCommand("record\_property\_right\_certificate", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("document\_id", MySqlDbType.VarChar, 9);

cmd.Parameters.Add("agency", MySqlDbType.VarChar, 255);

cmd.Parameters.Add("registrar\_passport\_id", MySqlDbType.VarChar, 9);

cmd.Parameters.Add("ownership\_reason", MySqlDbType.VarChar, 255);

cmd.Parameters.Add("issue\_reason", MySqlDbType.VarChar, 255);

cmd.Parameters.Add("ownership\_form", MySqlDbType.VarChar, 1);

cmd.Parameters.Add("share\_size", MySqlDbType.Int32);

cmd.Parameters.Add("passport\_id", MySqlDbType.VarChar, 9);

cmd.Parameters.Add("property\_object", MySqlDbType.VarChar, 255);

cmd.Parameters.Add("object\_id", MySqlDbType.Int32);

cmd.Parameters.Add("adress", MySqlDbType.VarChar, 255);

cmd.Parameters.Add("cadastral\_id", MySqlDbType.VarChar, 19);

cmd.Parameters.Add("object\_reg\_date", MySqlDbType.Date);

cmd.Parameters["document\_id"].Value = values["doc\_id"];

cmd.Parameters["agency"].Value = values["agency"];

cmd.Parameters["registrar\_passport\_id"].Value = values["registrar"];

cmd.Parameters["ownership\_reason"].Value = values["reason"];

cmd.Parameters["issue\_reason"].Value = values["issue"];

cmd.Parameters["ownership\_form"].Value = Convert.ToString(values["form"][0]);

cmd.Parameters["share\_size"].Value = Convert.ToInt32(values["share"]);

cmd.Parameters["passport\_id"].Value = values["passport"];

cmd.Parameters["property\_object"].Value = values["object"];

cmd.Parameters["object\_id"].Value = Convert.ToInt32(values["object\_id"]);

cmd.Parameters["adress"].Value = values["adress"];

cmd.Parameters["cadastral\_id"].Value = values["cadastral"];

cmd.Parameters["object\_reg\_date"].Value = Convert.ToDateTime(values["date"]);

cmd.Parameters["document\_id"].Direction = ParameterDirection.Input;

cmd.Parameters["agency"].Direction = ParameterDirection.Input;

cmd.Parameters["registrar\_passport\_id"].Direction = ParameterDirection.Input;

cmd.Parameters["ownership\_reason"].Direction = ParameterDirection.Input;

cmd.Parameters["issue\_reason"].Direction = ParameterDirection.Input;

cmd.Parameters["share\_size"].Direction = ParameterDirection.Input;

cmd.Parameters["passport\_id"].Direction = ParameterDirection.Input;

cmd.Parameters["property\_object"].Direction = ParameterDirection.Input;

cmd.Parameters["object\_id"].Direction = ParameterDirection.Input;

cmd.Parameters["ownership\_form"].Direction = ParameterDirection.Input;

cmd.Parameters["adress"].Direction = ParameterDirection.Input;

cmd.Parameters["cadastral\_id"].Direction = ParameterDirection.Input;

cmd.Parameters["object\_reg\_date"].Direction = ParameterDirection.Input;

cmd.ExecuteNonQuery();

}

public void RecordCarOwnership(Dictionary<string, string> values)

{

MySqlCommand cmd = new MySqlCommand("record\_transport\_registration\_certificate", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("passport\_id", MySqlDbType.VarChar, 9);

cmd.Parameters.Add("issue\_year", MySqlDbType.VarChar, 4);

cmd.Parameters.Add("reg\_id", MySqlDbType.VarChar, 8);

cmd.Parameters["passport\_id"].Value = values["passport"];

cmd.Parameters["issue\_year"].Value = values["year"];

cmd.Parameters["reg\_id"].Value = values["number"];

cmd.Parameters["passport\_id"].Direction = ParameterDirection.Input;

cmd.Parameters["issue\_year"].Direction = ParameterDirection.Input;

cmd.Parameters["reg\_id"].Direction = ParameterDirection.Input;

cmd.ExecuteNonQuery();

}

public void RecordDeath(Dictionary<string, string> values)

{

MySqlCommand cmd = new MySqlCommand("record\_death", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("passport\_id", MySqlDbType.VarChar, 9);

cmd.Parameters.Add("document\_id", MySqlDbType.VarChar, 9);

cmd.Parameters.Add("reg\_place", MySqlDbType.VarChar, 255);

cmd.Parameters.Add("agency", MySqlDbType.VarChar, 255);

cmd.Parameters["passport\_id"].Value = values["passport"];

cmd.Parameters["document\_id"].Value = values["cert"];

cmd.Parameters["reg\_place"].Value = values["place"];

cmd.Parameters["agency"].Value = values["agency"];

cmd.Parameters["passport\_id"].Direction = ParameterDirection.Input;

cmd.Parameters["document\_id"].Direction = ParameterDirection.Input;

cmd.Parameters["reg\_place"].Direction = ParameterDirection.Input;

cmd.Parameters["agency"].Direction = ParameterDirection.Input;

cmd.ExecuteNonQuery();

}

public MySqlDataReader CreateReaderOnPassportData(Dictionary<string, string> values)

{

MySqlCommand cmd = new MySqlCommand("find\_passport\_data", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("passport", MySqlDbType.VarChar, 9);

cmd.Parameters.Add("tax", MySqlDbType.VarChar, 10);

cmd.Parameters.Add("ref", MySqlDbType.VarChar, 13);

cmd.Parameters.Add("first\_name", MySqlDbType.VarChar, 30);

cmd.Parameters.Add("last\_name", MySqlDbType.VarChar, 30);

cmd.Parameters.Add("patronimic", MySqlDbType.VarChar, 30);

cmd.Parameters.Add("place", MySqlDbType.VarChar, 255);

cmd.Parameters["passport"].Value = values["passport"];

cmd.Parameters["tax"].Value = values["taxpayer"];

cmd.Parameters["ref"].Value = values["reference"];

cmd.Parameters["first\_name"].Value = values["first\_name"];

cmd.Parameters["last\_name"].Value = values["last\_name"];

cmd.Parameters["patronimic"].Value = values["patronimic"];

cmd.Parameters["place"].Value = values["place"];

cmd.Parameters["passport"].Direction = ParameterDirection.Input;

cmd.Parameters["tax"].Direction = ParameterDirection.Input;

cmd.Parameters["ref"].Direction = ParameterDirection.Input;

cmd.Parameters["first\_name"].Direction = ParameterDirection.Input;

cmd.Parameters["last\_name"].Direction = ParameterDirection.Input;

cmd.Parameters["patronimic"].Direction = ParameterDirection.Input;

cmd.Parameters["place"].Direction = ParameterDirection.Input;

return cmd.ExecuteReader();

}

public MySqlDataReader CreateReaderOnTransport(string registrationId)

{

MySqlCommand cmd = new MySqlCommand("find\_car\_by\_registration\_id", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("reg\_id", MySqlDbType.VarChar, 8);

cmd.Parameters["reg\_id"].Value = registrationId;

cmd.Parameters["reg\_id"].Direction = ParameterDirection.Input;

return cmd.ExecuteReader();

}

public MySqlDataReader CreateReaderOnProperty(string DocumentId)

{

MySqlCommand cmd = new MySqlCommand("find\_property\_by\_owner\_passport\_id", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("passport", MySqlDbType.VarChar, 9);

cmd.Parameters["passport"].Value = DocumentId;

cmd.Parameters["passport"].Direction = ParameterDirection.Input;

return cmd.ExecuteReader();

}

public MySqlDataReader CreateReaderOnOffspring(string DocumentId)

{

MySqlCommand cmd = new MySqlCommand("find\_children\_by\_dead\_parent", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("doc\_id", MySqlDbType.VarChar, 9);

cmd.Parameters["doc\_id"].Value = DocumentId;

cmd.Parameters["doc\_id"].Direction = ParameterDirection.Input;

return cmd.ExecuteReader();

}

public MySqlDataReader CreateReaderForPassportReport(string PassportId)

{

MySqlCommand cmd = new MySqlCommand("get\_passport", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("passport", MySqlDbType.VarChar, 9);

cmd.Parameters["passport"].Value = PassportId;

cmd.Parameters["passport"].Direction = ParameterDirection.Input;

return cmd.ExecuteReader();

}

public MySqlDataReader CreateReaderForTaxpayerReport(string PassportId)

{

MySqlCommand cmd = new MySqlCommand("get\_taxpayer\_certificate", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("passport", MySqlDbType.VarChar, 9);

cmd.Parameters["passport"].Value = PassportId;

cmd.Parameters["passport"].Direction = ParameterDirection.Input;

return cmd.ExecuteReader();

}

public MySqlDataReader CreateReaderForReferenceFromUSRReport(string PassportId)

{

MySqlCommand cmd = new MySqlCommand("get\_reference\_from\_usr", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("passport", MySqlDbType.VarChar, 9);

cmd.Parameters["passport"].Value = PassportId;

cmd.Parameters["passport"].Direction = ParameterDirection.Input;

return cmd.ExecuteReader();

}

public MySqlDataReader CreateReaderForBirthCertificateReport(string PassportId)

{

MySqlCommand cmd = new MySqlCommand("get\_birth\_certificate", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("passport", MySqlDbType.VarChar, 9);

cmd.Parameters["passport"].Value = PassportId;

cmd.Parameters["passport"].Direction = ParameterDirection.Input;

return cmd.ExecuteReader();

}

public MySqlDataReader CreateReaderForMarriageCertificateReport(string PassportId)

{

MySqlCommand cmd = new MySqlCommand("get\_marriage\_certificate", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("passport", MySqlDbType.VarChar, 9);

cmd.Parameters["passport"].Value = PassportId;

cmd.Parameters["passport"].Direction = ParameterDirection.Input;

return cmd.ExecuteReader();

}

public MySqlDataReader CreateReaderForPropertyRightCertificateReport(string PassportId)

{

MySqlCommand cmd = new MySqlCommand("get\_property\_ownership\_certificate", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("passport", MySqlDbType.VarChar, 9);

cmd.Parameters["passport"].Value = PassportId;

cmd.Parameters["passport"].Direction = ParameterDirection.Input;

return cmd.ExecuteReader();

}

public MySqlDataReader CreateReaderForTransportCertificateReport(string PassportId)

{

MySqlCommand cmd = new MySqlCommand("get\_transport\_certificate", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("passport", MySqlDbType.VarChar, 9);

cmd.Parameters["passport"].Value = PassportId;

cmd.Parameters["passport"].Direction = ParameterDirection.Input;

return cmd.ExecuteReader();

}

public List<string[]> ReadTransportCertificateForReport(string PassportId)

{

const int transportCertificateReaderFieldsAmount = 6;

List<string[]> results = new List<string[]>();

using (MySqlDataReader reader = CreateReaderForTransportCertificateReport(PassportId))

{

if (reader == null)

{

throw new Exception("Reader is not set");

}

if (!reader.HasRows)

{

throw new Exception("Нічого не знайдено");

}

while (reader.Read())

{

results.Add(new string[transportCertificateReaderFieldsAmount]);

for (int i = 0; i < transportCertificateReaderFieldsAmount; ++i)

{

results[results.Count - 1][i] = reader[i].ToString();

}

}

}

return results;

}

public List<string[]> ReadPropertyRightCertificateForReport(string PassportId)

{

const int propertyRightCertificateReaderFieldsAmount = 15;

List<string[]> results = new List<string[]>();

using (MySqlDataReader reader = CreateReaderForPropertyRightCertificateReport(PassportId))

{

if (reader == null)

{

throw new Exception("Reader is not set");

}

if (!reader.HasRows)

{

throw new Exception("Нічого не знайдено");

}

while (reader.Read())

{

results.Add(new string[propertyRightCertificateReaderFieldsAmount]);

for (int i = 0; i < propertyRightCertificateReaderFieldsAmount; ++i)

{

results[results.Count - 1][i] = reader[i].ToString();

}

}

}

return results;

}

public List<string> ReadMarriageCertificateForReport(string PassportId)

{

const int marriageCertificateReaderFieldsAmount = 12;

List<string> results = new List<string>();

using (MySqlDataReader reader = CreateReaderForMarriageCertificateReport(PassportId))

{

if (reader == null)

{

throw new Exception("Reader is not set");

}

if (!reader.HasRows)

{

throw new Exception("Нічого не знайдено");

}

while (reader.Read())

{

for (int i = 0; i < marriageCertificateReaderFieldsAmount; ++i)

{

results.Add(reader[i].ToString());

}

}

}

return results;

}

public List<string> ReadBirthCertificateForReport(string PassportId)

{

const int birthCertificateReaderFieldsAmount = 7;

List<string> results = new List<string>();

using (MySqlDataReader reader = CreateReaderForBirthCertificateReport(PassportId))

{

if (reader == null)

{

throw new Exception("Reader is not set");

}

if (!reader.HasRows)

{

throw new Exception("Нічого не знайдено");

}

while (reader.Read())

{

for (int i = 0; i < birthCertificateReaderFieldsAmount; ++i)

{

results.Add(reader[i].ToString());

}

}

}

return results;

}

public List<string> ReadReferenceFromUSRForReport(string PassportId)

{

const int referenceFromUSRReaderFieldsAmount = 9;

List<string> results = new List<string>();

using (MySqlDataReader reader = CreateReaderForReferenceFromUSRReport(PassportId))

{

if (reader == null)

{

throw new Exception("Reader is not set");

}

if (!reader.HasRows)

{

throw new Exception("Нічого не знайдено");

}

while (reader.Read())

{

for (int i = 0; i < referenceFromUSRReaderFieldsAmount; ++i)

{

results.Add(reader[i].ToString());

}

}

}

return results;

}

public List<string> ReadTaxpayerCertificateForReport(string PassportId)

{

const int taxpayerCertificateReaderFieldsAmount = 6;

List<string> results = new List<string>();

using (MySqlDataReader reader = CreateReaderForTaxpayerReport(PassportId))

{

if (reader == null)

{

throw new Exception("Reader is not set");

}

if (!reader.HasRows)

{

throw new Exception("Нічого не знайдено");

}

while (reader.Read())

{

for (int i = 0; i < taxpayerCertificateReaderFieldsAmount; ++i)

{

results.Add(reader[i].ToString());

}

}

}

return results;

}

public List<string> ReadPassportForReport(string PassportId)

{

const int passportReaderFieldsAmount = 12;

List<string> results = new List<string>();

using (MySqlDataReader reader = CreateReaderForPassportReport(PassportId))

{

if (reader == null)

{

throw new Exception("Reader is not set");

}

if (!reader.HasRows)

{

throw new Exception("Нічого не знайдено");

}

while (reader.Read())

{

for (int i = 0; i < passportReaderFieldsAmount; ++i)

{

results.Add(reader[i].ToString());

}

}

}

return results;

}

public List<string[]> ReadOffspringData(string deathDocumentId)

{

const int offspringDataReaderFieldsAmount = 11;

List<string[]> results = new List<string[]>();

using (MySqlDataReader reader = CreateReaderOnOffspring(deathDocumentId))

{

if (reader == null)

{

throw new Exception("Reader is not set");

}

if (!reader.HasRows)

{

throw new Exception("Нічого не знайдено");

}

while (reader.Read())

{

results.Add(new string[offspringDataReaderFieldsAmount]);

for (int i = 0; i < offspringDataReaderFieldsAmount; ++i)

{

results[results.Count - 1][i] = reader[i].ToString();

}

}

}

return results;

}

public List<string[]> ReadPropertyData(string propertyDocumentId)

{

const int propertyDataReaderFieldsAmount = 14;

List<string[]> results = new List<string[]>();

using (MySqlDataReader reader = CreateReaderOnProperty(propertyDocumentId))

{

if (reader == null)

{

throw new Exception("Reader is not set");

}

if (!reader.HasRows)

{

throw new Exception("Нічого не знайдено");

}

while (reader.Read())

{

results.Add(new string[propertyDataReaderFieldsAmount]);

for (int i = 0; i < propertyDataReaderFieldsAmount; ++i)

{

results[results.Count - 1][i] = reader[i].ToString();

}

}

}

return results;

}

public List<string[]> ReadTransportData(string registrationId)

{

const int transportDataReaderFieldsAmount = 5;

List<string[]> results = new List<string[]>();

using (MySqlDataReader reader = CreateReaderOnTransport(registrationId))

{

if (reader == null)

{

throw new Exception("Reader is not set");

}

if (!reader.HasRows)

{

throw new Exception("Нічого не знайдено");

}

while (reader.Read())

{

results.Add(new string[transportDataReaderFieldsAmount]);

for (int i = 0; i < transportDataReaderFieldsAmount; ++i)

{

results[results.Count - 1][i] = reader[i].ToString();

}

}

}

return results;

}

public List<string[]> ReadPassportData(Dictionary<string, string> values)

{

const int passportDataReaderFieldsAmount = 24;

List<string[]> results = new List<string[]>();

using (MySqlDataReader reader = CreateReaderOnPassportData(values))

{

if (reader == null)

{

throw new Exception("Reader is not set");

}

if (!reader.HasRows)

{

throw new Exception("Нічого не знайдено");

}

while (reader.Read())

{

results.Add(new string[passportDataReaderFieldsAmount]);

for (int i = 0; i < passportDataReaderFieldsAmount; ++i)

{

results[results.Count - 1][i] = reader[i].ToString();

}

}

}

return results;

}

}

}

## А.3 Клас DBUtils

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using MySql.Data.MySqlClient;

namespace Unified\_State\_Register

{

class DBUtils

{

public static MySqlConnection GetDBConnection(string host = "127.0.0.1",

int port = 3306,

string database = "unified\_state\_register",

string username = "root",

string password = "root")

{

String connString = "Server=" + host + ";Database=" + database

+ ";port=" + port + ";User Id=" + username

+ ";password=" + password;

MySqlConnection connection = new MySqlConnection(connString);

return connection;

}

}

}

## А.4 Клас Users

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Unified\_State\_Register

{

public class User

{

private string \_username;

private string \_password;

private char \_permission;

private string \_passportId;

public User(string username, string password, string passportId = null , char permission = 'S')

{

setUsername(username);

setPassword(password);

setPermission(permission);

setPassportId(passportId);

}

public string Username

{

get

{

return \_username;

}

}

public string Password

{

get

{

return \_password;

}

}

public char Permission

{

get

{

return \_permission;

}

}

public string PassportId

{

get

{

return \_passportId;

}

}

private void setUsername(string username)

{

if (username.Length < 4 || username.Length > 12)

{

throw UserException.usernameLengthError();

}

foreach (var ch in username)

{

if (Char.IsNumber(ch) || Char.IsLetter(ch))

{

continue;

}

else

{

throw UserException.usernameError();

}

}

\_username = username;

}

private void setPassword(string password)

{

if (password.Length < 8 || password.Length > 32)

{

throw UserException.passwordLengthError();

}

foreach (var ch in password)

{

if (Char.IsNumber(ch) || Char.IsLetter(ch))

{

continue;

}

else

{

throw UserException.passwordError();

}

}

\_password = password;

}

private void setPassportId(string passportId)

{

if (\_permission != 'S')

{

\_passportId = null;

return;

}

if (passportId != null)

{

if (passportId.Length != 9)

{

throw UserException.passportIdError();

}

foreach (var ch in passportId)

{

if (Char.IsNumber(ch))

{

continue;

}

else

{

throw UserException.passportIdError();

}

}

\_passportId = passportId;

}

}

private void setPermission(char permission)

{

if (permission != 'A' && permission != 'S' && permission != 'C')

{

throw UserException.permissionError();

}

\_permission = permission;

}

}

}

## А.5 Клас UserException

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Unified\_State\_Register

{

class UserException

{

public static Exception usernameError()

{

return new Exception("Не вірний логін. Логін має складатися з літер та цифр.");

}

public static Exception usernameLengthError()

{

return new Exception("Не вірний логін. Логін має бути від 4 до 12 символів");

}

public static Exception passwordError()

{

return new Exception("Не вірний пароль. Пароль має складатися з літер та цифр.");

}

public static Exception passwordLengthError()

{

return new Exception("Не вірний пароль. Пароль має від 8 до 32 символів");

}

public static Exception permissionError()

{

return new Exception("Невідомий тип доступу.");

}

public static Exception passportIdError()

{

return new Exception("Невірний номер паспорту.");

}

}

}

## А.6 Клас MainForm

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

namespace Unified\_State\_Register

{

public partial class MainForm : Form

{

private Database \_DB;

private SearchForm \_sForm;

private CreateForm \_cForm;

private PersonalForm \_pForm;

private User \_signedUser;

public MainForm(Database DB)

{

InitializeComponent();

\_DB = DB;

tbSignInPassword.UseSystemPasswordChar = true;

tbSignUpPassword.UseSystemPasswordChar = true;

tbSignUpPasswordRepeat.UseSystemPasswordChar = true;

}

public void makeVisible()

{

this.Visible = true;

if (\_sForm != null)

{

\_sForm.Visible = false;

\_sForm.Dispose();

}

if (\_cForm != null)

{

\_cForm.Visible = false;

\_cForm.Dispose();

}

if (\_pForm != null)

{

\_pForm.Visible = false;

\_pForm.Dispose();

}

\_signedUser = null;

}

public void openSearchForm()

{

\_sForm = new SearchForm(this, \_DB);

\_sForm.Show();

}

private void btnSignIn\_Click(object sender, EventArgs e)

{

try

{

\_signedUser = new User(tbSignInName.Text, tbSignInPassword.Text);

\_signedUser = \_DB.SignIn(\_signedUser);

}

catch (Exception ex)

{

MessageBox.Show("Помилка: " + ex.Message);

return;

}

tbSignInName.Clear();

tbSignInPassword.Clear();

if (\_signedUser.Permission == 'C')

{

\_cForm = new CreateForm(this, \_DB);

this.Visible = false;

\_cForm.Show();

}

if (\_signedUser.Permission == 'A')

{

\_sForm = new SearchForm(this, \_DB);

this.Visible = false;

\_sForm.Show();

}

if (\_signedUser.Permission == 'S')

{

\_pForm = new PersonalForm(this, \_DB, \_signedUser);

this.Visible = false;

\_pForm.Show();

}

}

private void btnSignUp\_Click(object sender, EventArgs e)

{

User newUser = null;

if (tbSignUpPassword.Text != tbSignUpPasswordRepeat.Text)

{

MessageBox.Show("Паролі не співпадають");

}

else

{

try

{

newUser = new User(tbSignUpName.Text, tbSignUpPassword.Text, tbSignUpPassportId.Text);

\_DB.SignUp(newUser);

}

catch (Exception ex)

{

MessageBox.Show("Помилка: " + ex.Message);

}

}

tbSignUpName.Clear();

tbSignUpPassword.Clear();

tbSignUpPasswordRepeat.Clear();

tbSignUpPassportId.Clear();

}

}

}

## А.7 Клас CreateForm

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Unified\_State\_Register

{

public partial class CreateForm : Form

{

private MainForm \_mForm;

private SearchForm \_sForm;

private Database \_DB;

private string \_mode;

public CreateForm(MainForm mForm, Database DB)

{

InitializeComponent();

\_DB = DB;

\_mForm = mForm;

\_sForm = null;

}

private void btnSearch\_Click(object sender, EventArgs e)

{

\_mForm.openSearchForm();

}

private void btnBirth\_Click(object sender, EventArgs e)

{

clearTextBoxes();

\_mode = "BIRTH";

makeVisible(\_mode);

}

private void btnPassport\_Click(object sender, EventArgs e)

{

clearTextBoxes();

\_mode = "PASSPORT";

makeVisible(\_mode);

}

private void btnLivePlace\_Click(object sender, EventArgs e)

{

clearTextBoxes();

\_mode = "LIVE";

makeVisible(\_mode);

}

private void btnOwnership\_Click(object sender, EventArgs e)

{

clearTextBoxes();

\_mode = "OWNERSHIP";

makeVisible(\_mode);

}

private void btnMarriage\_Click(object sender, EventArgs e)

{

clearTextBoxes();

\_mode = "MARRIAGE";

makeVisible(\_mode);

}

private void btnDeath\_Click(object sender, EventArgs e)

{

clearTextBoxes();

\_mode = "DEATH";

makeVisible(\_mode);

}

private void btnTaxpayer\_Click(object sender, EventArgs e)

{

clearTextBoxes();

\_mode = "TAXPAYER";

makeVisible(\_mode);

}

private void makeAllInvisible()

{

gbRegistrateBirth.Visible = false;

gbRegistrateLive.Visible = false;

gbRegistrateMarriage.Visible = false;

gbRegistrateOwnership.Visible = false;

gbRegistratePassport.Visible = false;

gbRegistrateDeath.Visible = false;

gbRegistrateTaxpayer.Visible = false;

}

private void makeVisible(string groupName)

{

makeAllInvisible();

switch(groupName)

{

case "BIRTH":

gbRegistrateBirth.Visible = true;

break;

case "PASSPORT":

gbRegistratePassport.Visible = true;

break;

case "LIVE":

gbRegistrateLive.Visible = true;

break;

case "MARRIAGE":

gbRegistrateMarriage.Visible = true;

break;

case "OWNERSHIP":

gbRegistrateOwnership.Visible = true;

break;

case "DEATH":

gbRegistrateDeath.Visible = true;

break;

case "TAXPAYER":

gbRegistrateTaxpayer.Visible = true;

break;

}

}

private void clearTextBoxes()

{

tbActIdMarriage.Clear();

tbAdressProperty.Clear();

tbAgencyBirth.Clear();

tbAgencyProperty.Clear();

tbAgencyReg.Clear();

tbAgencyTaxpayer.Clear();

tbAppartmentLiveC.Clear();

tbBirthCertBirth.Clear();

tbBirthReg.Clear();

tbCadastralIdProperty.Clear();

tbCertificateIdDeath.Clear();

tbCountryBirth.Clear();

tbDeadPassportIdDeath.Clear();

tbDeathRegistrationAgencyDeath.Clear();

tbDeathRegistrationPlaceDeath.Clear();

tbDocumentIdProperty.Clear();

tbFatherPassport.Clear();

tbFirstNameBirth.Clear();

tbFormTaxpayer.Clear();

tbHouseLiveC.Clear();

tbHusbandPassportMarriage.Clear();

tbIdTaxpayer.Clear();

tbIssuePlaceLiveC.Clear();

tbIssueReasonProperty.Clear();

tbIssueYearCar.Clear();

tbLastNameBirth.Clear();

tbLastNameReg.Clear();

tbMotherPassport.Clear();

tbNameReg.Clear();

tbObjectIdProperty.Clear();

tbObjectProperty.Clear();

tbOwnerPassportProperty.Clear();

tbOwnershipReasonProperty.Clear();

tbPassportBirth.Clear();

tbPassportIdCar.Clear();

tbPassportIdLiveC.Clear();

tbPassportReg.Clear();

tbPassportTaxpayer.Clear();

tbPatronimicBirth.Clear();

tbPatronimicReg.Clear();

tbPlaceBirth.Clear();

tbPlaceLiveC.Clear();

tbPlaceReg.Clear();

tbRecordIdLiveC.Clear();

tbReferenceIdLiveC.Clear();

tbRegionBirth.Clear();

tbRegionLiveC.Clear();

tbRegistationAgencyMarriage.Clear();

tbRegisterNumberCar.Clear();

tbRegistrarPassportProperty.Clear();

tbRegistrationPlaceMarriage.Clear();

tbRequestedStatementsLiveC.Clear();

tbShareProperty.Clear();

tbStreetLiveC.Clear();

tbTaxpayerReg.Clear();

tbWifePassportMarriage.Clear();

}

private void btnSignOut\_Click(object sender, EventArgs e)

{

if (\_sForm != null)

{

\_sForm.Visible = false;

\_sForm.Dispose();

}

\_mForm.makeVisible();

}

private void btnRegistrate\_Click(object sender, EventArgs e)

{

try

{

switch (\_mode)

{

case "BIRTH":

\_DB.RecordBirth(getDataFromBirthGroup());

break;

case "PASSPORT":

\_DB.RecordPassport(getDataFromPassportGroup());

//\_DB.RecordPhotoAndSignature(getDataPhotoAndSignatureGroup());

break;

case "LIVE":

\_DB.RecordLiveCertificate(getDataFromLiveCertificateGroup());

break;

case "MARRIAGE":

\_DB.RecordMarriage(getDataFromMarriageGroup());

break;

case "OWNERSHIP":

if (tabProperty.SelectedTab == tabProperty.TabPages["tabPage1"])

{

\_DB.RecordPropertyOwnership(getDataFromPropertyOwnershipGroup());

}

if (tabProperty.SelectedTab == tabProperty.TabPages["tabPage2"])

{

\_DB.RecordCarOwnership(getDataFromCarOwnershipGroup());

}

break;

case "DEATH":

\_DB.RecordDeath(getDataFromDeathGroup());

break;

case "TAXPAYER":

\_DB.RecordTaxpayer(getDataFromTaxpayerGroup());

break;

}

}

catch (Exception ex)

{

MessageBox.Show("Помилка: " + ex.Message);

return;

}

clearTextBoxes();

}

private Dictionary<string, string> getDataFromBirthGroup()

{

Dictionary<string, string> textBoxesData = new Dictionary<string, string>();

textBoxesData.Add("passport", tbPassportBirth.Text);

textBoxesData.Add("agency", tbAgencyBirth.Text);

textBoxesData.Add("birth\_cert", tbBirthCertBirth.Text);

textBoxesData.Add("country", tbCountryBirth.Text);

textBoxesData.Add("first\_name", tbFirstNameBirth.Text);

textBoxesData.Add("last\_name", tbLastNameBirth.Text);

textBoxesData.Add("patronimic", tbPatronimicBirth.Text);

textBoxesData.Add("place", tbPlaceBirth.Text);

textBoxesData.Add("region", tbRegionBirth.Text);

textBoxesData.Add("mother", tbMotherPassport.Text);

textBoxesData.Add("father", tbFatherPassport.Text);

textBoxesData.Add("date", dateBirth.Text);

textBoxesData.Add("sex", cbSexBirth.Text);

return textBoxesData;

}

private Dictionary<string, string> getDataFromPassportGroup()

{

Dictionary<string, string> textBoxesData = new Dictionary<string, string>();

textBoxesData.Add("agency", tbAgencyReg.Text);

textBoxesData.Add("birth\_cert", tbBirthReg.Text);

textBoxesData.Add("last\_name", tbLastNameReg.Text);

textBoxesData.Add("first\_name", tbNameReg.Text);

textBoxesData.Add("passport", tbPassportReg.Text);

textBoxesData.Add("patronimic", tbPatronimicReg.Text);

textBoxesData.Add("place", tbPlaceReg.Text);

textBoxesData.Add("taxpayer", tbTaxpayerReg.Text);

textBoxesData.Add("date", dateReg.Text);

textBoxesData.Add("sex", cbSexReg.Text);

return textBoxesData;

}

/\*private Dictionary<string, string> getDataPhotoAndSignatureGroup()

{

Dictionary<string, string> textBoxesData = new Dictionary<string, string>();

textBoxesData.Add("passport", tbPassportReg.Text);

textBoxesData.Add("photo", tbPhotoAdressReg.Text);

textBoxesData.Add("signature", tbSignatureReg.Text);

return textBoxesData;

}\*/

private Dictionary<string, string> getDataFromLiveCertificateGroup()

{

Dictionary<string, string> textBoxesData = new Dictionary<string, string>();

textBoxesData.Add("appartment", tbAppartmentLiveC.Text);

textBoxesData.Add("house", tbHouseLiveC.Text);

textBoxesData.Add("issue", tbIssuePlaceLiveC.Text);

textBoxesData.Add("place", tbPlaceLiveC.Text);

textBoxesData.Add("record", tbRecordIdLiveC.Text);

textBoxesData.Add("reference", tbReferenceIdLiveC.Text);

textBoxesData.Add("region", tbRegionLiveC.Text);

textBoxesData.Add("statements", tbRequestedStatementsLiveC.Text);

textBoxesData.Add("street", tbStreetLiveC.Text);

textBoxesData.Add("place\_type", cbPlaceTypeLiveC.Text);

textBoxesData.Add("passport", tbPassportIdLiveC.Text);

return textBoxesData;

}

private Dictionary<string, string> getDataFromMarriageGroup()

{

Dictionary<string, string> textBoxesData = new Dictionary<string, string>();

textBoxesData.Add("act", tbActIdMarriage.Text);

textBoxesData.Add("husband", tbHusbandPassportMarriage.Text);

textBoxesData.Add("agency", tbRegistationAgencyMarriage.Text);

textBoxesData.Add("place", tbRegistrationPlaceMarriage.Text);

textBoxesData.Add("wife", tbWifePassportMarriage.Text);

return textBoxesData;

}

private Dictionary<string, string> getDataFromDeathGroup()

{

Dictionary<string, string> textBoxesData = new Dictionary<string, string>();

textBoxesData.Add("cert", tbCertificateIdDeath.Text);

textBoxesData.Add("passport", tbDeadPassportIdDeath.Text);

textBoxesData.Add("place", tbDeathRegistrationPlaceDeath.Text);

textBoxesData.Add("agency", tbDeathRegistrationAgencyDeath.Text);

return textBoxesData;

}

private Dictionary<string, string> getDataFromPropertyOwnershipGroup()

{

Dictionary<string, string> textBoxesData = new Dictionary<string, string>();

textBoxesData.Add("adress", tbAdressProperty.Text);

textBoxesData.Add("agency", tbAgencyProperty.Text);

textBoxesData.Add("cadastral", tbCadastralIdProperty.Text);

textBoxesData.Add("doc\_id", tbDocumentIdProperty.Text);

textBoxesData.Add("issue", tbIssueReasonProperty.Text);

textBoxesData.Add("object\_id", tbObjectIdProperty.Text);

textBoxesData.Add("object", tbObjectProperty.Text);

textBoxesData.Add("passport", tbOwnerPassportProperty.Text);

textBoxesData.Add("reason", tbOwnershipReasonProperty.Text);

textBoxesData.Add("registrar", tbRegistrarPassportProperty.Text);

textBoxesData.Add("share", tbShareProperty.Text);

textBoxesData.Add("date", dateObjectRegistrationProperty.Text);

textBoxesData.Add("form", cbOwnershipFormProperty.Text);

return textBoxesData;

}

private Dictionary<string, string> getDataFromCarOwnershipGroup()

{

Dictionary<string, string> textBoxesData = new Dictionary<string, string>();

textBoxesData.Add("year", tbIssueYearCar.Text);

textBoxesData.Add("passport", tbPassportIdCar.Text);

textBoxesData.Add("number", tbRegisterNumberCar.Text);

return textBoxesData;

}

private Dictionary<string, string> getDataFromTaxpayerGroup()

{

Dictionary<string, string> textBoxesData = new Dictionary<string, string>();

textBoxesData.Add("agency", tbAgencyTaxpayer.Text);

textBoxesData.Add("form\_id", tbFormTaxpayer.Text);

textBoxesData.Add("taxpayer", tbIdTaxpayer.Text);

textBoxesData.Add("passport", tbPassportTaxpayer.Text);

return textBoxesData;

}

private void btnClear\_Click(object sender, EventArgs e)

{

clearTextBoxes();

}

}

}

## А.8 Клас SearchForm

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Unified\_State\_Register

{

public partial class SearchForm : Form

{

private MainForm \_mForm;

private Database \_DB;

public SearchForm(MainForm mForm, Database DB)

{

InitializeComponent();

CreateColumnsInDataGrids();

ShowColumnsByCheckBoxes();

\_DB = DB;

\_mForm = mForm;

}

private void btnSignOut\_Click(object sender, EventArgs e)

{

\_mForm.makeVisible();

}

private void btnShow\_Click(object sender, EventArgs e)

{

try

{

if (tabControl.SelectedTab == tabControl.TabPages["tabPassport"])

{

dataGridPassport.Rows.Clear();

fillDataGridPassport(\_DB.ReadPassportData(ReadPassportTexboxes()));

}

if (tabControl.SelectedTab == tabControl.TabPages["tabTransport"])

{

dataGridCar.Rows.Clear();

fillDataGridCar(\_DB.ReadTransportData(tbTransportRegId.Text));

}

if (tabControl.SelectedTab == tabControl.TabPages["tabProperty"])

{

dataGridProperty.Rows.Clear();

fillDataGridProperty(\_DB.ReadPropertyData(tbpropertyDocId.Text));

}

if (tabControl.SelectedTab == tabControl.TabPages["tabDeath"])

{

dataGridDeath.Rows.Clear();

fillDataGridDeath(\_DB.ReadOffspringData(tbDeathId.Text));

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

return;

}

ShowColumnsByCheckBoxes();

}

private void fillDataGridPassport(List<string[]> values)

{

foreach (var value in values)

{

dataGridPassport.Rows.Add(value);

}

}

private void fillDataGridCar(List<string[]> values)

{

foreach (var value in values)

{

dataGridCar.Rows.Add(value);

}

}

private void fillDataGridProperty(List<string[]> values)

{

foreach (var value in values)

{

dataGridProperty.Rows.Add(value);

}

}

private void fillDataGridDeath(List<string[]> values)

{

foreach (var value in values)

{

dataGridDeath.Rows.Add(value);

}

}

private Dictionary<string, string> ReadPassportTexboxes()

{

Dictionary<string, string> data = new Dictionary<string, string>();

data.Add("passport", cbDocType.Text != "Паспорт" ? "%" : string.IsNullOrEmpty(tbDocId.Text) ? "%" : tbDocId.Text);

data.Add("taxpayer", cbDocType.Text != "КПП" ? "%" : string.IsNullOrEmpty(tbDocId.Text) ? "%" : tbDocId.Text);

data.Add("reference", cbDocType.Text != "Витяг з ЄДР" ? "%" : string.IsNullOrEmpty(tbDocId.Text) ? "%" : tbDocId.Text);

data.Add("first\_name", string.IsNullOrEmpty(tbFirstName.Text) ? "%" : tbFirstName.Text);

data.Add("last\_name", string.IsNullOrEmpty(tbLastName.Text) ? "%" : tbLastName.Text);

data.Add("patronimic", string.IsNullOrEmpty(tbPatronimic.Text) ? "%" : tbPatronimic.Text);

data.Add("place", (string.IsNullOrEmpty(tbRegion.Text) ? "%" : tbRegion.Text) +

(string.IsNullOrEmpty(tbPlace.Text) ? "%" : tbPlace.Text) +

(string.IsNullOrEmpty(tbStreet.Text) ? "%" : tbStreet.Text) +

(string.IsNullOrEmpty(tbHouse.Text) ? "%" : tbHouse.Text) +

(string.IsNullOrEmpty(tbAppartment.Text) ? "%" : tbAppartment.Text));

return data;

}

private void CreateColumnsInDataGrids()

{

dataGridPassport.Columns.Add("colPassport", "Паспорт");

dataGridPassport.Columns.Add("colFirstName", "Ім\'я");

dataGridPassport.Columns.Add("colLastName", "Прізвище");

dataGridPassport.Columns.Add("colPatronimic", "По Батькові");

dataGridPassport.Columns.Add("colSex", "Стать");

dataGridPassport.Columns.Add("colPassportIssue", "Дата видачі");

dataGridPassport.Columns.Add("colPassportValidity", "Дійсний до");

dataGridPassport.Columns.Add("colPassportAgency", "Орган, що видав");

dataGridPassport.Columns.Add("colBirth", "Свід. про нар.");

dataGridPassport.Columns.Add("colBirthDate", "Дата народження");

dataGridPassport.Columns.Add("colBirtPlace", "Місце народження");

dataGridPassport.Columns.Add("colFatherPassport", "Паспорт Батька");

dataGridPassport.Columns.Add("colMotherPassport", "Паспорт матері");

dataGridPassport.Columns.Add("colBirthReg", "Місце реєстрації");

dataGridPassport.Columns.Add("colRecord", "Витяг з ЄДР");

dataGridPassport.Columns.Add("colLivePlace", "Місце проживання");

dataGridPassport.Columns.Add("colIssuePlace", "Місце отримання");

dataGridPassport.Columns.Add("colRegDate", "Дата занесення до реєстру");

dataGridPassport.Columns.Add("colReference", "Номер витягу");

dataGridPassport.Columns.Add("colStatements", "Відомості, що запитуються");

dataGridPassport.Columns.Add("colTaxpayer", "КПП");

dataGridPassport.Columns.Add("colHusband", "Чоловік");

dataGridPassport.Columns.Add("colWife", "Дружина");

dataGridPassport.Columns.Add("colAct", "Акт");

dataGridCar.Columns.Add("colRegId","Реєстраційний номер");

dataGridCar.Columns.Add("colIssueYear", "Рік випуску");

dataGridCar.Columns.Add("colRegDate", "Дата реєстрації");

dataGridCar.Columns.Add("colPassport", "Паспорт володаря");

dataGridCar.Columns.Add("colValidityDate", "Дійсний до");

dataGridProperty.Columns.Add("colDocId", "Номер документу");

dataGridProperty.Columns.Add("colRegDate", "Дата реєстрації");

dataGridProperty.Columns.Add("colAgency", "Агенство");

dataGridProperty.Columns.Add("colRegistrar", "Парспарт реєстратора");

dataGridProperty.Columns.Add("colOwnershipReason", "Причина володіння");

dataGridProperty.Columns.Add("colIssueReason", "Причина видачі");

dataGridProperty.Columns.Add("colOwnershipForm", "Форма власності");

dataGridProperty.Columns.Add("colShare", "Доля");

dataGridProperty.Columns.Add("colOwnerPassport", "Паспорт володаря");

dataGridProperty.Columns.Add("colPropertyObject", "");

dataGridProperty.Columns.Add("colObjectId", "Номер об\'єкту");

dataGridProperty.Columns.Add("colAdress", "Адреса");

dataGridProperty.Columns.Add("colCadastralId", "Кадастровий номер");

dataGridProperty.Columns.Add("colObjectRegDate", "Дата рєстрації об\'єкту");

dataGridDeath.Columns.Add("colPassport", "Паспорт нащадка");

dataGridDeath.Columns.Add("colFirstName", "Ім\'я");

dataGridDeath.Columns.Add("colSecondName", "Прізвище");

dataGridDeath.Columns.Add("colPatronimic", "По Батькові");

dataGridDeath.Columns.Add("colSex", "Стать");

dataGridDeath.Columns.Add("colBirthId", "Свідоцтво про нар.");

dataGridDeath.Columns.Add("colBirthDate", "Дата нар.");

dataGridDeath.Columns.Add("colFather", "Паспорт Батька");

dataGridDeath.Columns.Add("colMother", "Паспорт Матері");

dataGridDeath.Columns.Add("colBirthPlace", "Місце народження");

dataGridDeath.Columns.Add("colRegPlace", "Місце реєстрації");

}

private void ShowColumnsByCheckBoxes()

{

bool passportVisibility = chbPassport.Checked,

birthCertificateVisibility = chbBirth.Checked,

liveCertificateVisibility = chbLive.Checked,

taxpayerCertificateVisibility = chbTaxpayer.Checked,

marriageCertificateVisibility = chbMarriage.Checked;

dataGridPassport.Columns["colPassport"].Visible = passportVisibility;

dataGridPassport.Columns["colFirstName"].Visible = passportVisibility;

dataGridPassport.Columns["colLastName"].Visible = passportVisibility;

dataGridPassport.Columns["colPatronimic"].Visible = passportVisibility;

dataGridPassport.Columns["colSex"].Visible = passportVisibility;

dataGridPassport.Columns["colPassportIssue"].Visible = passportVisibility;

dataGridPassport.Columns["colPassportValidity"].Visible = passportVisibility;

dataGridPassport.Columns["colPassportAgency"].Visible = passportVisibility;

dataGridPassport.Columns["colBirth"].Visible = birthCertificateVisibility;

dataGridPassport.Columns["colBirthDate"].Visible = birthCertificateVisibility;

dataGridPassport.Columns["colBirtPlace"].Visible = birthCertificateVisibility;

dataGridPassport.Columns["colFatherPassport"].Visible = birthCertificateVisibility;

dataGridPassport.Columns["colMotherPassport"].Visible = birthCertificateVisibility;

dataGridPassport.Columns["colBirthReg"].Visible = birthCertificateVisibility;

dataGridPassport.Columns["colRecord"].Visible = liveCertificateVisibility;

dataGridPassport.Columns["colLivePlace"].Visible = liveCertificateVisibility;

dataGridPassport.Columns["colIssuePlace"].Visible = liveCertificateVisibility;

dataGridPassport.Columns["colRegDate"].Visible = liveCertificateVisibility;

dataGridPassport.Columns["colReference"].Visible = liveCertificateVisibility;

dataGridPassport.Columns["colStatements"].Visible = liveCertificateVisibility;

dataGridPassport.Columns["colTaxpayer"].Visible = taxpayerCertificateVisibility;

dataGridPassport.Columns["colHusband"].Visible = marriageCertificateVisibility;

dataGridPassport.Columns["colWife"].Visible = marriageCertificateVisibility;

dataGridPassport.Columns["colAct"].Visible = marriageCertificateVisibility;

}

private void chbPassport\_CheckedChanged(object sender, EventArgs e)

{

ShowColumnsByCheckBoxes();

}

private void chbBirth\_CheckedChanged(object sender, EventArgs e)

{

ShowColumnsByCheckBoxes();

}

private void chbLive\_CheckedChanged(object sender, EventArgs e)

{

ShowColumnsByCheckBoxes();

}

private void chbTaxpayer\_CheckedChanged(object sender, EventArgs e)

{

ShowColumnsByCheckBoxes();

}

private void chbMarriage\_CheckedChanged(object sender, EventArgs e)

{

ShowColumnsByCheckBoxes();

}

private void btnClear\_Click(object sender, EventArgs e)

{

tbAppartment.Clear();

tbDeathId.Clear();

tbDocId.Clear();

tbFirstName.Clear();

tbHouse.Clear();

tbLastName.Clear();

tbPatronimic.Clear();

tbPlace.Clear();

tbpropertyDocId.Clear();

tbRegion.Clear();

tbStreet.Clear();

tbTransportRegId.Clear();

}

}

}

## А.9 Клас PersonalForm

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Unified\_State\_Register

{

public partial class PersonalForm : Form

{

private MainForm \_mForm;

private Database \_DB;

private User \_user;

private ReportPassportForm \_passportReport;

private ReportTaxpayerCertificateForm \_taxpayerReport;

private ReportReferenceFromUSRForm \_referenceFromUSRReport;

private ReportBirthCertificateForm \_birthCertificateReport;

private ReportMarriageCertificateForm \_marriageCertificateReport;

private List<ReportTransportCertificateForm> \_transportCertificateReports;

private List<ReportPropertyOwnershipForm> \_propertyOwnershipReports;

public PersonalForm(MainForm mForm, Database DB, User user)

{

InitializeComponent();

\_mForm = mForm;

\_DB = DB;

\_user = user;

}

private void btnPassport\_Click(object sender, EventArgs e)

{

try

{

\_passportReport = new ReportPassportForm(\_DB.ReadPassportForReport(\_user.PassportId));

\_passportReport.Show();

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

return;

}

}

private void btnTaxpayer\_Click(object sender, EventArgs e)

{

try

{

\_taxpayerReport = new ReportTaxpayerCertificateForm(\_DB.ReadTaxpayerCertificateForReport(\_user.PassportId));

\_taxpayerReport.Show();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

return;

}

}

private void btnReference\_Click(object sender, EventArgs e)

{

try

{

\_referenceFromUSRReport = new ReportReferenceFromUSRForm(\_DB.ReadReferenceFromUSRForReport(\_user.PassportId));

\_referenceFromUSRReport.Show();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

return;

}

}

private void btnBirthCertificate\_Click(object sender, EventArgs e)

{

try

{

\_birthCertificateReport = new ReportBirthCertificateForm(\_DB.ReadBirthCertificateForReport(\_user.PassportId));

\_birthCertificateReport.Show();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

return;

}

}

private void btnMarriageCertificate\_Click(object sender, EventArgs e)

{

try

{

\_marriageCertificateReport = new ReportMarriageCertificateForm(\_DB.ReadMarriageCertificateForReport(\_user.PassportId));

\_marriageCertificateReport.Show();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

return;

}

}

private void btnTransportCertificate\_Click(object sender, EventArgs e)

{

try

{

List<string[]> values = \_DB.ReadTransportCertificateForReport(\_user.PassportId);

foreach (var value in values)

{

ReportTransportCertificateForm form = new ReportTransportCertificateForm(value);

form.Show();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

return;

}

}

private void btnPropertyRightCertificate\_Click(object sender, EventArgs e)

{

try

{

List<string[]> values = \_DB.ReadPropertyRightCertificateForReport(\_user.PassportId);

foreach (var value in values)

{

ReportPropertyOwnershipForm form = new ReportPropertyOwnershipForm(value);

form.Show();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

return;

}

}

private void btnSignOut\_Click(object sender, EventArgs e)

{

\_mForm.makeVisible();

}

private void PersonalForm\_Load(object sender, EventArgs e)

{

}

}

}

## А.10 Клас ReportPassportForm

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Unified\_State\_Register

{

public partial class ReportPassportForm : Form

{

private List<string> \_passportData;

public ReportPassportForm(List<string> passportData)

{

InitializeComponent();

\_passportData = passportData;

}

private void ReportPassportForm\_Load(object sender, EventArgs e)

{

this.reportPassport.ProcessingMode = Microsoft.Reporting.WinForms.ProcessingMode.Local;

Microsoft.Reporting.WinForms.ReportParameter[] parameters = new Microsoft.Reporting.WinForms.ReportParameter[]

{

new Microsoft.Reporting.WinForms.ReportParameter ("PassportId", \_passportData[0]),

new Microsoft.Reporting.WinForms.ReportParameter ("FirstName", \_passportData[1]),

new Microsoft.Reporting.WinForms.ReportParameter ("LastName", \_passportData[2]),

new Microsoft.Reporting.WinForms.ReportParameter ("Patronimic", \_passportData[3]),

new Microsoft.Reporting.WinForms.ReportParameter ("Sex", \_passportData[4]),

new Microsoft.Reporting.WinForms.ReportParameter ("IssueDate", Convert.ToDateTime(\_passportData[5]).ToString("dd-MM-yyyy")),

new Microsoft.Reporting.WinForms.ReportParameter ("ValidityDate", Convert.ToDateTime(\_passportData[6]).ToString("dd-MM-yyyy")),

new Microsoft.Reporting.WinForms.ReportParameter ("Agency", \_passportData[7]),

new Microsoft.Reporting.WinForms.ReportParameter ("TaxpayerId", \_passportData[8]),

new Microsoft.Reporting.WinForms.ReportParameter ("RecordId", \_passportData[9]),

new Microsoft.Reporting.WinForms.ReportParameter ("BirthDate", Convert.ToDateTime(\_passportData[10]).ToString("dd-MM-yyyy")),

new Microsoft.Reporting.WinForms.ReportParameter ("BirthPlace", \_passportData[11])

};

this.reportPassport.LocalReport.SetParameters(parameters);

this.reportPassport.RefreshReport();

}

}

}

## А.11 Клас ReportBirthCertificateForm

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Unified\_State\_Register

{

public partial class ReportBirthCertificateForm : Form

{

private List<string> \_birthCertificateData;

public ReportBirthCertificateForm(List<string> birthCertificateData)

{

InitializeComponent();

\_birthCertificateData = birthCertificateData;

}

private void ReportBirthCertificateForm\_Load(object sender, EventArgs e)

{

this.reportBirthCertificate.ProcessingMode = Microsoft.Reporting.WinForms.ProcessingMode.Local;

Microsoft.Reporting.WinForms.ReportParameter[] parameters = new Microsoft.Reporting.WinForms.ReportParameter[]

{

new Microsoft.Reporting.WinForms.ReportParameter ("Name", \_birthCertificateData[0]),

new Microsoft.Reporting.WinForms.ReportParameter ("CertificateId", \_birthCertificateData[1]),

new Microsoft.Reporting.WinForms.ReportParameter ("BirthDate", Convert.ToDateTime(\_birthCertificateData[2]).ToString("dd-MM-yyyy")),

new Microsoft.Reporting.WinForms.ReportParameter ("BirthPlace", \_birthCertificateData[3]),

new Microsoft.Reporting.WinForms.ReportParameter ("RegPlace", \_birthCertificateData[4]),

new Microsoft.Reporting.WinForms.ReportParameter ("Father", \_birthCertificateData[5]),

new Microsoft.Reporting.WinForms.ReportParameter ("Mother",\_birthCertificateData[6])

};

this.reportBirthCertificate.LocalReport.SetParameters(parameters);

this.reportBirthCertificate.RefreshReport();

}

}

}

## А.12 Клас ReportTaxpeyerCertificateForm

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Unified\_State\_Register

{

public partial class ReportTaxpayerCertificateForm : Form

{

private List<string> \_taxpayerData;

public ReportTaxpayerCertificateForm(List<string> taxpayerData)

{

InitializeComponent();

\_taxpayerData = taxpayerData;

}

private void ReportTaxpayerCertificateForm\_Load(object sender, EventArgs e)

{

this.reportTaxpayerCertificate.ProcessingMode = Microsoft.Reporting.WinForms.ProcessingMode.Local;

Microsoft.Reporting.WinForms.ReportParameter[] parameters = new Microsoft.Reporting.WinForms.ReportParameter[]

{

new Microsoft.Reporting.WinForms.ReportParameter ("Name", \_taxpayerData[0]),

new Microsoft.Reporting.WinForms.ReportParameter ("BirthDate", Convert.ToDateTime(\_taxpayerData[1]).ToString("dd-MM-yyyy")),

new Microsoft.Reporting.WinForms.ReportParameter ("TaxpayerId", \_taxpayerData[2]),

new Microsoft.Reporting.WinForms.ReportParameter ("RegDate", Convert.ToDateTime(\_taxpayerData[3]).ToString("dd-MM-yyyy")),

new Microsoft.Reporting.WinForms.ReportParameter ("Agency", \_taxpayerData[4]),

new Microsoft.Reporting.WinForms.ReportParameter ("FormId", \_taxpayerData[5])

};

this.reportTaxpayerCertificate.LocalReport.SetParameters(parameters);

this.reportTaxpayerCertificate.RefreshReport();

}

}

}

## А.13 Клас ReportReferenceFromUSRForm

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Unified\_State\_Register

{

public partial class ReportReferenceFromUSRForm : Form

{

private List<string> \_referenceFromUSRData;

public ReportReferenceFromUSRForm(List<string> referenceFromUSRData)

{

InitializeComponent();

\_referenceFromUSRData = referenceFromUSRData;

}

private void ReportReferenceFromUSRForm\_Load(object sender, EventArgs e)

{

this.reportReferenceFromUSR.ProcessingMode = Microsoft.Reporting.WinForms.ProcessingMode.Local;

Microsoft.Reporting.WinForms.ReportParameter[] parameters = new Microsoft.Reporting.WinForms.ReportParameter[]

{

new Microsoft.Reporting.WinForms.ReportParameter ("Name", \_referenceFromUSRData[0]),

new Microsoft.Reporting.WinForms.ReportParameter ("BirthDate", Convert.ToDateTime(\_referenceFromUSRData[1]).ToString("dd-MM-yyyy")),

new Microsoft.Reporting.WinForms.ReportParameter ("BirthPlace", \_referenceFromUSRData[2]),

new Microsoft.Reporting.WinForms.ReportParameter ("RecordId", \_referenceFromUSRData[3]),

new Microsoft.Reporting.WinForms.ReportParameter ("ReferenceId", \_referenceFromUSRData[4]),

new Microsoft.Reporting.WinForms.ReportParameter ("RegistrationPlace",\_referenceFromUSRData[5]),

new Microsoft.Reporting.WinForms.ReportParameter ("IssuePlace", \_referenceFromUSRData[6]),

new Microsoft.Reporting.WinForms.ReportParameter ("RegDate", Convert.ToDateTime(\_referenceFromUSRData[7]).ToString("dd-MM-yyyy")),

new Microsoft.Reporting.WinForms.ReportParameter ("ReqStatements", \_referenceFromUSRData[8]),

};

this.reportReferenceFromUSR.LocalReport.SetParameters(parameters);

this.reportReferenceFromUSR.RefreshReport();

}

}

}

## А.14 Клас ReportMarriageCertificateForm

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Unified\_State\_Register

{

public partial class ReportMarriageCertificateForm : Form

{

private List<string> \_marriageCertificateData;

public ReportMarriageCertificateForm(List<string> marriageCertificateData)

{

InitializeComponent();

\_marriageCertificateData = marriageCertificateData;

}

private void ReportMarriageCertificateForm\_Load(object sender, EventArgs e)

{

this.reportMarriageCertificate.ProcessingMode = Microsoft.Reporting.WinForms.ProcessingMode.Local;

Microsoft.Reporting.WinForms.ReportParameter[] parameters = new Microsoft.Reporting.WinForms.ReportParameter[]

{

new Microsoft.Reporting.WinForms.ReportParameter ("RegDate", Convert.ToDateTime(\_marriageCertificateData[0]).ToString("dd-MM-yyyy")),

new Microsoft.Reporting.WinForms.ReportParameter ("ActId", \_marriageCertificateData[1]),

new Microsoft.Reporting.WinForms.ReportParameter ("HusbandBefore", \_marriageCertificateData[2]),

new Microsoft.Reporting.WinForms.ReportParameter ("HusbandAfter", \_marriageCertificateData[3]),

new Microsoft.Reporting.WinForms.ReportParameter ("WifeBefore", \_marriageCertificateData[4]),

new Microsoft.Reporting.WinForms.ReportParameter ("WifeAfter", \_marriageCertificateData[5]),

new Microsoft.Reporting.WinForms.ReportParameter ("RegPlace", \_marriageCertificateData[6]),

new Microsoft.Reporting.WinForms.ReportParameter ("Agency", \_marriageCertificateData[7]),

new Microsoft.Reporting.WinForms.ReportParameter ("HusbandName", \_marriageCertificateData[8]),

new Microsoft.Reporting.WinForms.ReportParameter ("WifeName", \_marriageCertificateData[9]),

new Microsoft.Reporting.WinForms.ReportParameter ("HusbandBirthDate", Convert.ToDateTime(\_marriageCertificateData[10]).ToString("dd-MM-yyyy")),

new Microsoft.Reporting.WinForms.ReportParameter ("WifeBirthDate", Convert.ToDateTime(\_marriageCertificateData[11]).ToString("dd-MM-yyyy"))

};

this.reportMarriageCertificate.LocalReport.SetParameters(parameters);

this.reportMarriageCertificate.RefreshReport();

}

}

}

## А.15 Клас ReportTransportCertificateForm

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Unified\_State\_Register

{

public partial class ReportTransportCertificateForm : Form

{

private string[] \_transportCertificateData;

public ReportTransportCertificateForm(string[] transportCertificateData)

{

InitializeComponent();

\_transportCertificateData = transportCertificateData;

}

private void ReportTransportCertificateForm\_Load(object sender, EventArgs e)

{

this.reportTransportCertificate.ProcessingMode = Microsoft.Reporting.WinForms.ProcessingMode.Local;

Microsoft.Reporting.WinForms.ReportParameter[] parameters = new Microsoft.Reporting.WinForms.ReportParameter[]

{

new Microsoft.Reporting.WinForms.ReportParameter ("RegId", \_transportCertificateData[0]),

new Microsoft.Reporting.WinForms.ReportParameter ("IssueYear", \_transportCertificateData[1]),

new Microsoft.Reporting.WinForms.ReportParameter ("RegDate", Convert.ToDateTime(\_transportCertificateData[2]).ToString("dd-MM-yyyy")),

new Microsoft.Reporting.WinForms.ReportParameter ("ValDate", Convert.ToDateTime(\_transportCertificateData[3]).ToString("dd-MM-yyyy")),

new Microsoft.Reporting.WinForms.ReportParameter ("Name", \_transportCertificateData[4]),

new Microsoft.Reporting.WinForms.ReportParameter ("RegLive",\_transportCertificateData[5]),

};

this.reportTransportCertificate.LocalReport.SetParameters(parameters);

this.reportTransportCertificate.RefreshReport();

}

}

}

## А.16 Клас ReportPropertyOwnershipForm

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Unified\_State\_Register

{

public partial class ReportPropertyOwnershipForm : Form

{

private string[] \_propertyOwnershipData;

public ReportPropertyOwnershipForm(string[] propertyOwnershipData)

{

InitializeComponent();

\_propertyOwnershipData = propertyOwnershipData;

}

private void ReportPropertyOwnershipForm\_Load(object sender, EventArgs e)

{

this.reportPropertyOwnership.ProcessingMode = Microsoft.Reporting.WinForms.ProcessingMode.Local;

Microsoft.Reporting.WinForms.ReportParameter[] parameters = new Microsoft.Reporting.WinForms.ReportParameter[]

{

new Microsoft.Reporting.WinForms.ReportParameter ("DocId", \_propertyOwnershipData[0]),

new Microsoft.Reporting.WinForms.ReportParameter ("RegDate", Convert.ToDateTime(\_propertyOwnershipData[1]).ToString("dd-MM-yyyy")),

new Microsoft.Reporting.WinForms.ReportParameter ("Agency", \_propertyOwnershipData[2]),

new Microsoft.Reporting.WinForms.ReportParameter ("OwnershipReason", \_propertyOwnershipData[3]),

new Microsoft.Reporting.WinForms.ReportParameter ("IssueReason", \_propertyOwnershipData[4]),

new Microsoft.Reporting.WinForms.ReportParameter ("OwnershipForm", \_propertyOwnershipData[5]),

new Microsoft.Reporting.WinForms.ReportParameter ("Share", \_propertyOwnershipData[6]),

new Microsoft.Reporting.WinForms.ReportParameter ("PropertyObject", \_propertyOwnershipData[7]),

new Microsoft.Reporting.WinForms.ReportParameter ("ObjectId", \_propertyOwnershipData[8]),

new Microsoft.Reporting.WinForms.ReportParameter ("Adress", \_propertyOwnershipData[9]),

new Microsoft.Reporting.WinForms.ReportParameter ("CadastralId", \_propertyOwnershipData[10]),

new Microsoft.Reporting.WinForms.ReportParameter ("ObjectRegDate",Convert.ToDateTime(\_propertyOwnershipData[11]).ToString("dd-MM-yyyy")),

new Microsoft.Reporting.WinForms.ReportParameter ("RegistrarName", \_propertyOwnershipData[12]),

new Microsoft.Reporting.WinForms.ReportParameter ("OwnerName", \_propertyOwnershipData[13]),

new Microsoft.Reporting.WinForms.ReportParameter ("OwnerTaxpayerId", \_propertyOwnershipData[14])

};

this.reportPropertyOwnership.LocalReport.SetParameters(parameters);

this.reportPropertyOwnership.RefreshReport();

}

}

}

# ДОДАТОК Б КОД ВБУДОВАНИХ ПРОЦЕДУР ТА ФУНКЦІЙ

## Б.1 Процедура sign\_up\_user

drop procedure if exists sign\_up\_user ;

delimiter $$

create procedure sign\_up\_user(in username varchar(12), in user\_password varchar(32), in permission char(1), in passport\_id char(9))

begin

declare exception condition for sqlstate '45000';

if is\_username\_exists(username)

then signal exception set message\_text = "Ім'я користувача вже зайнято";

end if;

if is\_user\_passport\_exists(passport\_id)

then signal exception set message\_text = "На цей номер паспорту вже зареєстровано користувача";

end if;

if not is\_passport\_exists(passport\_id)

then signal exception set message\_text = "Не вірний номер паспорту";

end if;

insert into users (User\_Name, User\_Password, User\_Permission, Passport\_Id)

values (username, user\_password, permission, passport\_id);

end$$

delimiter ;

## Б.2 Процедура sign\_in\_user

drop procedure if exists sign\_in\_user;

delimiter $$

create procedure sign\_in\_user(in username varchar(12), in user\_password varchar(32), out permission char(1), out passport\_id char(9))

begin

declare existance bool;

declare exception condition for sqlstate '45000';

set existance = is\_username\_exists(username);

if not existance

then signal exception set message\_text = "Невірне ім'я користувача";

end if;

set existance = is\_password\_correct(username, user\_password);

if not existance

then signal exception set message\_text = "Не вірний пароль";

end if;

select users.User\_Permission into permission from users where user\_password = users.User\_Password and username = users.User\_Name;

select users.Passport\_Id into passport\_id from users where user\_password = users.User\_Password and username = users.User\_Name;

end$$

delimiter ;

## Б.3 Процедура record\_birth

drop procedure if exists record\_birth;

delimiter $$

create procedure record\_birth(in passport\_id char(9), in first\_name varchar(30), in last\_name varchar(30), in patronimic varchar(30), in sex char(1), birth\_cert\_id char(9),

in date\_of\_birth date, in passport\_f char(9), in passport\_m char(9), in place\_of\_birth varchar(255), in place\_of\_reg varchar(255))

begin

declare exception condition for sqlstate '45000';

if is\_passport\_exists(passport\_id)

then signal exception set message\_text = "Цей номер паспорту вже зайнято";

end if;

if is\_birth\_certificate\_exists(birth\_cert\_id)

then signal exception set message\_text = "Цей номер свідоцтва про народження вже зареєстровано";

end if;

insert into birth\_certificate (Birth\_Certificate\_Id, Date\_of\_Birth, Passport\_Id\_Father, Passport\_Id\_Mother, Place\_of\_Birth, Place\_of\_Registration)

values (birth\_cert\_id, date\_of\_birth, passport\_f, passport\_m, place\_of\_birth, place\_of\_reg);

insert into passport (Passport\_Id, First\_Name, Last\_Name, Patronymic, Sex, Birth\_Certificate\_Id)

values (passport\_id, first\_name, last\_name, patronimic, sex, birth\_cert\_id);

end$$

delimiter ;

## Б.4 Процедура record\_reference\_from\_usr

drop procedure if exists record\_reference\_from\_usr;

delimiter $$

create procedure record\_reference\_from\_usr(in passport char(9), in record char(13), in reference\_id char(14), in place\_of\_reg varchar(255), in place\_of\_issue varchar(255), in requested\_statements varchar(255))

begin

declare exception condition for sqlstate '45000';

if not is\_passport\_exists(passport)

then signal exception set message\_text = "Невірний номер паспорту";

end if;

if is\_reference\_from\_usr\_exists(record)

then call archivate\_reference\_from\_usr(record);

end if;

insert into reference\_from\_usr (Record\_Id, Reference\_Id, Place\_of\_Issue, Registration\_Date, Requested\_Statements, Place\_of\_Registration)

values (record, reference\_id, place\_of\_issue, curdate(), requested\_statements, place\_of\_reg);

if is\_passport\_has\_reference\_from\_usr(passport, record)

then call archivate\_reference\_from\_usr(record);

end if;

SET FOREIGN\_KEY\_CHECKS=0;

update passport

set Record\_Id = record where Passport\_Id = passport;

SET FOREIGN\_KEY\_CHECKS=1;

end$$

delimiter ;

## Б.5 Процедура record\_taxpayer\_certificate

drop procedure if exists record\_taxpayer\_certificate;

delimiter $$

create procedure record\_taxpayer\_certificate(in passport\_id char(9), in taxpayer\_id char(10), in fiscal\_service\_agency varchar(255), in form\_id char(10))

begin

declare exception condition for sqlstate '45000';

if not is\_passport\_exists(passport\_id)

then signal exception set message\_text = "Невірний номер паспорту";

end if;

if is\_taxpayer\_certificate\_exists(taxpayer\_id)

then signal exception set message\_text = "КПП з таким номером вже існує";

end if;

if is\_passport\_has\_taxpayer\_id(passport\_id, taxpayer\_id)

then signal exception set message\_text = "Цей пасспорт вже має КПП";

end if;

insert into taxpayer\_certificate (Taxpayer\_Id, Registration\_Date, Fiscal\_Service\_Agency, Form\_Id)

values (taxpayer\_id, curdate(), fiscal\_service\_agency, form\_id);

update passport

set passport.Taxpayer\_Id = taxpayer\_id where passport.Passport\_Id = passport\_id;

end$$

delimiter ;

## Б.6 Процедура record\_passport

drop procedure if exists record\_passport;

delimiter $$

create procedure record\_passport(in passport\_id char(9), in first\_name varchar(30), in last\_name varchar(30), in patronimic varchar(30), in sex char(1),

in date\_of\_issue date, in agency\_id char(4), in taxpayer\_id char(10), in birth\_cert\_id char(9), in record\_id char(13))

begin

declare exception condition for sqlstate '45000';

declare validity\_date date default null;

declare birth\_date date default null;

if not is\_passport\_exists(passport\_id)

then signal exception set message\_text = "Невірний номер паспорту";

end if;

if not is\_taxpayer\_certificate\_exists(taxpayer\_id)

then signal exception set message\_text = "КПП з таким номером не існує";

end if;

if not is\_reference\_from\_usr\_exists(record\_id)

then signal exception set message\_text = "Виятгу з ЄДР не існує";

end if;

if not is\_birth\_certificate\_exists(birth\_cert\_id)

then signal exception set message\_text = "Цього свідоцтва про народження не існує";

end if;

if not is\_passport\_has\_taxpayer\_id(passport\_id, taxpayer\_id)

then signal exception set message\_text = "Цей пасспорт не має КПП, або має інше";

end if;

if not is\_passport\_has\_reference\_from\_usr(passport\_id, record\_id)

then signal exception set message\_text = "Цей пасспорт не має витягу або має інший";

end if;

if not is\_passport\_has\_birth\_certificate(passport\_id, birth\_cert\_id)

then signal exception set message\_text = "Цей паспорт не має свідоцтва про народження або має інше";

end if;

call archivate\_passport(passport\_id);

select Date\_of\_Birth into birth\_date from birth\_certificate where birth\_certificate.Birth\_Certificate\_Id = birth\_cert\_id;

if year(date\_of\_issue) - year(birth\_date) < 17

then set validity\_date = date\_add(date\_of\_issue, interval 4 year);

else

set validity\_date = date\_add(date\_of\_issue, interval 10 year);

end if;

insert into passport(Passport\_Id, First\_Name, Last\_Name, Patronymic, Sex, Date\_of\_Issue, Validity\_Date, Agency\_Id, Taxpayer\_Id, Birth\_Certificate\_Id, Record\_Id)

values (passport\_id, first\_name, last\_name, patronimic, sex, date\_of\_issue, validity\_date, agency\_id, taxpayer\_id, birth\_cert\_id, record\_id);

end$$

delimiter ;

## Б.7 Процедура record\_marriage

drop procedure if exists record\_marriage;

delimiter $$

create procedure record\_marriage(in passport\_h char(9), in passport\_w char(9), in reg\_place varchar(255), in agency varchar(255), in act\_id int)

begin

declare exception condition for sqlstate '45000';

declare husband\_last\_name varchar(30) default null;

declare wife\_last\_name varchar(30) default null;

if not is\_passport\_exists(passport\_h) or not is\_passport\_exists(passport\_w)

then signal exception set message\_text = "Невірний номер паспорту";

end if;

select passport.Last\_Name into husband\_last\_name from passport where passport.Passport\_Id = passport\_h;

select passport.Last\_Name into wife\_last\_name from passport where passport.Passport\_Id = passport\_w;

insert into marriage\_certificate (First\_Passport\_Id, Second\_Passport\_Id, Registration\_Date, Act\_Id, First\_Lastname\_Before, First\_Lastname\_After, Second\_Lastname\_Before, Second\_Lastname\_After, Registration\_Place, Agency)

values (passport\_h, passport\_w, curdate(), act\_id, husband\_last\_name, husband\_last\_name, wife\_last\_name, husband\_last\_name, reg\_place, agency);

end$$

delimiter ;

## Б.8 Процедура record\_transport\_registration\_certificate

drop procedure if exists record\_transport\_registration\_certificate;

delimiter $$

create procedure record\_transport\_registration\_certificate(in reg\_id char(8), in issue\_year char(4), in passport\_id char(9))

begin

declare exception condition for sqlstate '45000';

if not is\_passport\_exists(passport\_id)

then signal exception set message\_text = "Невірний номер паспорту";

end if;

insert into transport\_registration\_certificate (Registration\_Id, Issue\_Year, Registration\_Date, Passport\_Id, Validity\_Date)

values (reg\_id, issue\_year, curdate(), passport\_id, date\_add(curdate(), interval 10 year));

end$$

delimiter ;

## Б.9 Процедура record\_property\_right\_certificate

drop procedure if exists record\_property\_right\_certificate;

delimiter $$

create procedure record\_property\_right\_certificate(in document\_id char(9), in agency varchar(255), in registrar\_passport\_id char(9), in ownership\_reason varchar(255), in issue\_reason varchar(255),

in ownership\_form char(1), in share\_size int, in passport\_id char(9), in property\_object varchar(255), in object\_id int, in adress varchar(255),

in cadastral\_id char(19), in object\_reg\_date date)

begin

declare exception condition for sqlstate '45000';

if not is\_passport\_exists(passport\_id)

then signal exception set message\_text = "Невірний номер паспорту";

end if;

insert into property\_right\_certificate

values (document\_id, curdate(), agency, registrar\_passport\_id, ownership\_reason, issue\_reason, ownership\_form, share\_size, passport\_id, property\_object, object\_id, adress, cadastral\_id, object\_reg\_date);

end$$

delimiter ;

## Б.10 Процедура record\_death

drop procedure if exists record\_death;

delimiter $$

create procedure record\_death(in passport\_id char(9), in document\_id char(9), in reg\_place varchar(255), in agency varchar(255))

begin

declare exception condition for sqlstate '45000';

declare birth\_cert\_id char(9) default null;

declare tax\_cert\_id char(10) default null;

declare recod\_id char(13) default null;

if not is\_passport\_exists(passport\_id)

then signal exception set message\_text = "Невірний номер паспорту";

end if;

select passport.Birth\_Certificate\_Id into birth\_cert\_id from passport where passport.Passport\_Id = passport\_id;

select passport.Taxpayer\_Id into tax\_cert\_id from passport where passport.Passport\_Id = passport\_id;

select passport.Record\_Id into recod\_id from passport where passport.Passport\_Id = passport\_id;

call archivate\_birth\_certificate(birth\_cert\_id);

call archivate\_taxpayer\_certificate(tax\_cert\_id);

call archivate\_reference\_from\_usr(recod\_id);

call archivate\_marriage(passport\_id);

call archivate\_transport\_registration\_certificate(passport\_id);

call archivate\_property\_right\_certificate(passport\_id);

call archivate\_passport(passport\_id);

set foreign\_key\_checks=0;

insert into death\_certificate

values (passport\_id, curdate(), reg\_place, agency, document\_id);

insert into arch\_unified\_passport\_id(Passport\_Id)

values(passport\_id);

set foreign\_key\_checks=1;

end$$

delimiter ;

## Б.11 Процедура archivate\_reference\_from\_usr

drop procedure if exists archivate\_reference\_from\_usr;

delimiter $$

create procedure archivate\_reference\_from\_usr(in record\_id char(13))

begin

declare exception condition for sqlstate '45000';

if is\_reference\_from\_usr\_exists(record\_id) then

set foreign\_key\_checks=0;

insert into arch\_reference\_from\_usr (Record\_Id, Reference\_Id, Place\_of\_Registration, Place\_of\_Issue, Registration\_Date, Requested\_Statements)

select reference\_from\_usr.Record\_Id, reference\_from\_usr.Reference\_Id, reference\_from\_usr.Place\_of\_Registration, reference\_from\_usr.Place\_of\_Issue, reference\_from\_usr.Registration\_Date, reference\_from\_usr.Requested\_Statements

from reference\_from\_usr where reference\_from\_usr.Record\_Id = record\_id;

update arch\_reference\_from\_usr set arch\_reference\_from\_usr.Arch\_Date = curdate() where arch\_reference\_from\_usr.Arch\_Doc\_Id = last\_insert\_id();

delete from reference\_from\_usr where reference\_from\_usr.Record\_Id = record\_id;

set foreign\_key\_checks=1;

else

signal exception set message\_text = "Такого витягу з ЄДР не існує";

end if;

end$$

delimiter ;

## Б.12 Процедура archivate\_passport

drop procedure if exists archivate\_passport;

delimiter $$

create procedure archivate\_passport(in passport\_id char(9))

begin

declare exception condition for sqlstate '45000';

declare last\_record bigint default null;

declare record\_from\_passport char(13) default null;

if not is\_passport\_exists(passport\_id)

then signal exception set message\_text = "Невірний номер паспорту";

end if;

select passport.Record\_Id into record\_from\_passport from passport where passport.Passport\_Id = passport\_id;

select arch\_reference\_from\_usr.Arch\_Doc\_Id into last\_record from arch\_reference\_from\_usr where arch\_reference\_from\_usr.Record\_Id = record\_from\_passport;

SET FOREIGN\_KEY\_CHECKS=0;

insert into arch\_passport(Passport\_Id, First\_Name, Last\_Name, Patronymic, Sex, Date\_of\_Issue, Validity\_Date, Agency\_Id, Taxpayer\_Id, Birth\_Certificate\_Id)

select Passport\_Id, First\_Name, Last\_Name, Patronymic, Sex, Date\_of\_Issue, Validity\_Date, Agency\_Id, Taxpayer\_Id, Birth\_Certificate\_Id from passport where passport.Passport\_Id = passport\_id;

update arch\_passport set arch\_passport.Record\_Id = last\_record, arch\_passport.Arch\_Date = curdate() where arch\_passport.Arch\_Doc\_Id = last\_insert\_id();

delete from passport where passport.Passport\_Id = passport\_id;

SET FOREIGN\_KEY\_CHECKS=1;

end$$

delimiter ;

## Б.13 Процедура archivate\_marriage

drop procedure if exists archivate\_marriage;

delimiter $$

create procedure archivate\_marriage(in passport char(9))

arch\_marriage:

begin

declare act int default null;

if not is\_marriage\_certificate\_exists(passport)

then leave arch\_marriage;

end if;

select marriage\_certificate.Act\_Id into act from marriage\_certificate where marriage\_certificate.First\_Passport\_Id = passport or marriage\_certificate.Second\_Passport\_Id = passport;

set foreign\_key\_checks=0;

insert into arch\_marriage\_certificate (First\_Passport\_Id, Second\_Passport\_Id, Registration\_Date, Act\_Id, First\_Lastname\_Before, First\_Lastname\_After, Second\_Lastname\_Before, Second\_Lastname\_After, Registration\_Place, Agency)

select \* from marriage\_certificate where marriage\_certificate.First\_Passport\_Id = passport or marriage\_certificate.Second\_Passport\_Id = passport;

update arch\_marriage\_certificate set arch\_marriage\_certificate.Arch\_Date = curdate() where arch\_marriage\_certificate.Act\_Id = act;

delete from marriage\_certificate where marriage\_certificate.First\_Passport\_Id = passport or marriage\_certificate.Second\_Passport\_Id = passport;

set foreign\_key\_checks=1;

end$$

delimiter ;

## Б.14 Процедура archivate\_transport\_registration\_certificate

drop procedure if exists archivate\_transport\_registration\_certificate;

delimiter $$

create procedure archivate\_transport\_registration\_certificate(in passport char(9))

arch\_car:

begin

if count\_of\_transport\_certificates(passport) = 0

then leave arch\_car;

end if;

set foreign\_key\_checks=0;

insert into arch\_transport\_registration\_certificate

select \* from transport\_registration\_certificate where transport\_registration\_certificate.Passport\_Id = passport;

delete from transport\_registration\_certificate where transport\_registration\_certificate.Passport\_Id = passport;

set foreign\_key\_checks=1;

end$$

delimiter ;

## Б.15 Процедура archivate\_property\_right\_certificate

drop procedure if exists archivate\_property\_right\_certificate;

delimiter $$

create procedure archivate\_property\_right\_certificate(in passport char(9))

arch\_property:

begin

if count\_of\_property\_certificates(passport) = 0

then leave arch\_property;

end if;

set foreign\_key\_checks=0;

insert into arch\_property\_right\_certificate (Document\_Id, Registration\_Date, Agency, Registrar\_Passport\_Id, Ownership\_Reason, Issue\_Reason, Ownership\_Form, Share\_Size, Owner\_Passport\_Id, Property\_Object, Object\_Id, Adress, Cadastral\_Id, Object\_Regestration\_Date)

select \* from property\_right\_certificate where property\_right\_certificate.Owner\_Passport\_Id = passport;

update arch\_property\_right\_certificate set arch\_property\_right\_certificate.Arch\_Date = curdate() where arch\_property\_right\_certificate.Owner\_Passport\_Id = passport;

delete from property\_right\_certificate where property\_right\_certificate.Owner\_Passport\_Id = passport;

set foreign\_key\_checks=1;

end$$

delimiter ;

## Б.16 Процедура archivate\_taxpayer\_certificate

drop procedure if exists archivate\_taxpayer\_certificate;

delimiter $$

create procedure archivate\_taxpayer\_certificate(in taxpayer\_id char(10))

begin

set foreign\_key\_checks=0;

insert into arch\_taxpayer\_certificate(Taxpayer\_Id, Registration\_Date, Fiscal\_Service\_Agency, Form\_Id)

select taxpayer\_certificate.Taxpayer\_Id, taxpayer\_certificate.Registration\_Date, taxpayer\_certificate.Fiscal\_Service\_Agency, taxpayer\_certificate.Form\_Id

from taxpayer\_certificate where taxpayer\_certificate.Taxpayer\_Id = taxpayer\_id;

delete from taxpayer\_certificate where taxpayer\_certificate.Taxpayer\_Id = taxpayer\_id;

set foreign\_key\_checks=1;

end$$

delimiter ;

## Б.17 Процедура archivate\_birth\_certificate

drop procedure if exists archivate\_birth\_certificate;

delimiter $$

create procedure archivate\_birth\_certificate(in birth\_cert\_id char(9))

begin

set foreign\_key\_checks=0;

insert into arch\_birth\_certificate (Birth\_Certificate\_Id, Date\_of\_Birth, Passport\_Id\_Father, Passport\_Id\_Mother, Place\_of\_Birth, Place\_of\_Registration)

select \* from birth\_certificate where birth\_certificate.Birth\_Certificate\_Id = birth\_cert\_id;

delete from birth\_certificate where birth\_certificate.Birth\_Certificate\_Id = birth\_cert\_id;

set foreign\_key\_checks=1;

end$$

delimiter ;

## Б.18 Процедура find\_car\_by\_registration\_id

drop procedure if exists find\_car\_by\_registration\_id ;

delimiter $$

create procedure find\_car\_by\_registration\_id(in reg\_id char(8))

begin

select \* from transport\_registration\_certificate where transport\_registration\_certificate.Registration\_Id like concat ("%", reg\_id, "%")

order by Registration\_Date;

end$$

delimiter ;

## Б.19 Процедура find\_property\_by\_owner\_passport\_id

drop procedure if exists find\_property\_by\_owner\_passport\_id ;

delimiter $$

create procedure find\_property\_by\_owner\_passport\_id(in passport char(9))

begin

select \* from property\_right\_certificate where property\_right\_certificate.Owner\_Passport\_Id like concat("%", passport, "%")

order by Registration\_Date;

end$$

delimiter ;

## Б.20 Процедура find\_children\_by\_dead\_parent

drop procedure if exists find\_children\_by\_dead\_parent ;

delimiter $$

create procedure find\_children\_by\_dead\_parent(in doc\_id char(9))

begin

declare passport char(9) default null;

select death\_certificate.Passport\_Id into passport from death\_certificate where death\_certificate.Document\_Id = doc\_id;

select passport\_data.Passport\_Id, passport\_data.First\_Name, passport\_data.Last\_Name, passport\_data.Patronymic, passport\_data.Sex,

passport\_data.Birth\_Certificate\_Id, passport\_data.Date\_of\_Birth, passport\_data.Passport\_Id\_Father, passport\_data.Passport\_Id\_Mother, passport\_data.Place\_of\_Birth, passport\_data.Place\_of\_Registration

from passport\_data

where passport\_data.Passport\_Id\_Father = passport or passport\_data.Passport\_Id\_Mother = passport

order by passport\_data.Date\_of\_Birth;

end$$

delimiter ;

## Б.21 Процедура find\_passport\_data

drop procedure if exists find\_passport\_data;

delimiter $$

create procedure find\_passport\_data(in passport char(9), in tax char(10), in ref char(13), in first\_name varchar(30), in last\_name varchar(30), in patronimic varchar(30), in place varchar(255))

begin

select \* from passport\_data where (passport\_data.Passport\_Id like concat("%", passport, "%")

and passport\_data.First\_Name like concat("%", first\_name, "%")

and passport\_data.Last\_Name like concat("%", last\_name, "%")

and passport\_data.Patronymic like concat("%", patronimic, "%"))

and ((passport\_data.Taxpayer\_Id like concat("%", tax, "%")) or (tax = "%" and passport\_data.Taxpayer\_Id is null))

and ((passport\_data.Record\_Id like concat("%", ref, "%")) or (ref = "%" and passport\_data.Record\_Id is null))

and ((passport\_data.Place\_of\_Registration\_Live like concat("%", place, "%")) or (place = "%%%%%" and passport\_data.Place\_of\_Registration\_Live is null));

end$$

delimiter ;

## Б.22 Процедура get\_passport

drop procedure if exists get\_passport;

delimiter $$

create procedure get\_passport(in passport char(9))

begin

select Passport\_Id, First\_Name, Last\_Name, Patronymic, Sex, Date\_of\_Issue, Validity\_Date, Agency\_Id, Taxpayer\_Id, Record\_Id, Date\_of\_Birth, Place\_of\_Birth

from passport\_data where passport\_data.Passport\_Id = passport;

end$$

delimiter ;

## Б.23 Процедура get\_taxpayer\_certificate

drop procedure if exists get\_taxpayer\_certificate;

delimiter $$

create procedure get\_taxpayer\_certificate(in passport char(9))

begin

select concat(passport.Last\_Name," ", passport.First\_Name," ", passport.Patronymic),

birth\_certificate.Date\_of\_Birth,

taxpayer\_certificate.Taxpayer\_Id, taxpayer\_certificate.Registration\_Date, taxpayer\_certificate.Fiscal\_Service\_Agency, taxpayer\_certificate.Form\_Id

from passport inner join birth\_certificate on passport.Birth\_Certificate\_Id = birth\_certificate.Birth\_Certificate\_Id

inner join taxpayer\_certificate on passport.Taxpayer\_Id = taxpayer\_certificate.Taxpayer\_Id

where passport.Passport\_Id = passport;

end$$

delimiter ;

## Б.24 Процедура get\_reference\_from\_usr

drop procedure if exists get\_reference\_from\_usr;

delimiter $$

create procedure get\_reference\_from\_usr(in passport char(9))

begin

select concat(passport.Last\_Name," ", passport.First\_Name," ", passport.Patronymic, " "),

birth\_certificate.Date\_of\_Birth, birth\_certificate.Place\_of\_Birth,

reference\_from\_usr.Record\_Id, reference\_from\_usr.Reference\_Id, reference\_from\_usr.Place\_of\_Registration, reference\_from\_usr.Place\_of\_Issue, reference\_from\_usr.Registration\_Date, reference\_from\_usr.Requested\_Statements

from passport inner join birth\_certificate on passport.Birth\_Certificate\_Id = birth\_certificate.Birth\_Certificate\_Id

inner join reference\_from\_usr on passport.Record\_Id = reference\_from\_usr.Record\_Id

where passport.Passport\_Id = passport;

end$$

delimiter ;

## Б.25 Процедура get\_birth\_certificate

drop procedure if exists get\_birth\_certificate;

delimiter $$

create procedure get\_birth\_certificate(in passport char(9))

begin

declare passport\_father char(9) default null;

declare passport\_mother char(9) default null;

declare father\_name varchar(92) default null;

declare mother\_name varchar(92) default null;

select birth\_certificate.Passport\_Id\_Father, birth\_certificate.Passport\_Id\_Mother into passport\_father, passport\_mother

from passport inner join birth\_certificate on passport.Birth\_Certificate\_Id = birth\_certificate.Birth\_Certificate\_Id

where passport.Passport\_Id = passport;

if is\_passport\_exists(passport\_father)

then select concat(passport.Last\_Name," ", passport.First\_Name," ", passport.Patronymic) into father\_name from passport where passport.Passport\_Id = passport\_father;

else select concat(arch\_passport.Last\_Name," ", arch\_passport.First\_Name," ", arch\_passport.Patronymic) into father\_name

from arch\_passport

where arch\_passport.Passport\_Id = passport\_father and arch\_passport.Arch\_Doc\_Id = (select max(arch\_passport.Arch\_Doc\_Id) from arch\_passport where arch\_passport.Passport\_Id = passport\_father);

end if;

if is\_passport\_exists(passport\_mother)

then select concat(passport.Last\_Name," ", passport.First\_Name," ", passport.Patronymic) into mother\_name from passport where passport.Passport\_Id = passport\_mother;

else select concat(arch\_passport.Last\_Name," ", arch\_passport.First\_Name," ", arch\_passport.Patronymic) into mother\_name

from arch\_passport

where arch\_passport.Passport\_Id = passport\_mother and arch\_passport.Arch\_Doc\_Id = (select max(arch\_passport.Arch\_Doc\_Id) from arch\_passport where arch\_passport.Passport\_Id = passport\_mother);

end if;

select concat(passport.Last\_Name," ", passport.First\_Name," ", passport.Patronymic),

birth\_certificate.Birth\_Certificate\_Id, birth\_certificate.Date\_of\_Birth, birth\_certificate.Place\_of\_Birth, birth\_certificate.Place\_of\_Registration,

(father\_name), (mother\_name)

from passport inner join birth\_certificate on passport.Birth\_Certificate\_Id = birth\_certificate.Birth\_Certificate\_Id

where passport.Passport\_Id = passport;

end$$

delimiter ;

## Б.26 Процедура get\_marriage\_certificate

drop procedure if exists get\_marriage\_certificate;

delimiter $$

create procedure get\_marriage\_certificate(in passport char(9))

begin

declare exception condition for sqlstate '45000';

declare first\_passport char(9) default null;

declare second\_passport char(9) default null;

select passport\_data.First\_Passport\_Id, passport\_data.Second\_Passport\_Id into first\_passport, second\_passport

from passport\_data

where passport\_data.Passport\_Id = passport;

if first\_passport = null or second\_passport = null

then signal exception set message\_text = "Не перебуває у шлюбі";

end if;

select marriage\_certificate.Registration\_Date, marriage\_certificate.Act\_Id, marriage\_certificate.First\_Lastname\_Before, marriage\_certificate.First\_Lastname\_After, marriage\_certificate.Second\_Lastname\_Before, marriage\_certificate.Second\_Lastname\_After, marriage\_certificate.Registration\_Place, marriage\_certificate.Agency,

(select concat(passport\_data.Last\_Name," ", passport\_data.First\_Name," ", passport\_data.Patronymic) from passport\_data where passport\_data.Passport\_Id = first\_passport),

(select concat(passport\_data.Last\_Name," ", passport\_data.First\_Name," ", passport\_data.Patronymic) from passport\_data where passport\_data.Passport\_Id = second\_passport),

(select passport\_data.Date\_of\_Birth from passport\_data where passport\_data.Passport\_Id = first\_passport),

(select passport\_data.Date\_of\_Birth from passport\_data where passport\_data.Passport\_Id = second\_passport)

from marriage\_certificate

where marriage\_certificate.First\_Passport\_Id = first\_passport and marriage\_certificate.Second\_Passport\_Id = second\_passport;

end$$

delimiter ;

## Б.27 Процедура get\_transport\_certificate

drop procedure if exists get\_transport\_certificate;

delimiter $$

create procedure get\_transport\_certificate(in passport char(9))

begin

declare exception condition for sqlstate '45000';

if count\_of\_transport\_certificates(passport) = 0

then signal exception set message\_text = "Жодного свідоцтва реєстрації транспорту не знайдено";

end if;

select transport\_registration\_certificate.Registration\_Id, transport\_registration\_certificate.Issue\_Year, transport\_registration\_certificate.Registration\_Date, transport\_registration\_certificate.Validity\_Date,

(select concat(passport\_data.Last\_Name," ", passport\_data.First\_Name," ", passport\_data.Patronymic) from passport\_data where passport\_data.Passport\_Id = passport),

(select passport\_data.Place\_of\_Registration\_Live from passport\_data where passport\_data.Passport\_Id = passport)

from transport\_registration\_certificate

where transport\_registration\_certificate.Passport\_Id = passport;

end$$

delimiter ;

## Б.28 Процедура get\_property\_ownership\_certificate

drop procedure if exists get\_property\_ownership\_certificate;

delimiter $$

create procedure get\_property\_ownership\_certificate(in passport char(9))

begin

declare exception condition for sqlstate '45000';

declare registrar\_passport char(9) default null;

if count\_of\_property\_certificates(passport) = 0

then signal exception set message\_text = "Жодного свідоцтва реєстрації нерухомого майна не знайдено";

end if;

select property\_right\_certificate.Registrar\_Passport\_Id into registrar\_passport from property\_right\_certificate where property\_right\_certificate.Owner\_Passport\_Id = passport;

select property\_right\_certificate.Document\_Id, property\_right\_certificate.Registration\_Date, property\_right\_certificate.Agency, property\_right\_certificate.Ownership\_Reason, property\_right\_certificate.Issue\_Reason, property\_right\_certificate.Ownership\_Form, property\_right\_certificate.Share\_Size, property\_right\_certificate.Property\_Object, property\_right\_certificate.Object\_Id, property\_right\_certificate.Adress, property\_right\_certificate.Cadastral\_Id, property\_right\_certificate.Object\_Regestration\_Date,

(select concat(passport\_data.Last\_Name, " ", passport\_data.First\_Name, " ", passport\_data.Patronymic) from passport\_data where passport\_data.Passport\_Id = registrar\_passport),

(select concat(passport\_data.Last\_Name, " ", passport\_data.First\_Name, " ", passport\_data.Patronymic) from passport\_data where passport\_data.Passport\_Id = passport),

(select passport\_data.Taxpayer\_Id from passport\_data where passport\_data.Passport\_Id = passport)

from property\_right\_certificate

where property\_right\_certificate.Owner\_Passport\_Id = passport;

end$$

delimiter ;

## Б.29 Функція is\_username\_exists

drop function if exists is\_username\_exists;

delimiter $$

create function is\_username\_exists(username varchar(12)) returns bool deterministic

begin

declare existance bool default false;

declare num int default 0;

select count(users.User\_Name) into num from users where username = users.User\_Name;

if num > 0

then set existance = true;

end if;

return existance;

end$$

delimiter ;

## Б.30 Функція is\_user\_passport\_exists

drop function if exists is\_user\_passport\_exists;

delimiter $$

create function is\_user\_passport\_exists(passport varchar(9)) returns bool deterministic

begin

declare existance bool default false;

declare num int default 0;

select count(users.Passport\_Id) into num from users where passport = users.Passport\_Id;

if num > 0

then set existance = true;

end if;

return existance;

end$$

delimiter ;

## Б.31 Функція is\_passport\_exists

drop function if exists is\_passport\_exists;

delimiter $$

create function is\_passport\_exists(passport\_id varchar(9)) returns bool deterministic

begin

declare existance bool default false;

declare num int default 0;

select count(\*) into num from passport where passport\_id = passport.Passport\_Id;

if num > 0

then set existance = true;

end if;

return existance;

end$$

delimiter ;

## Б.32 Функція is\_birth\_certificate\_exists

drop function if exists is\_birth\_certificate\_exists;

delimiter $$

create function is\_birth\_certificate\_exists(birth\_certificate\_id char(9)) returns bool deterministic

begin

declare existance bool default false;

declare num int default 0;

select count(birth\_certificate.Birth\_Certificate\_Id) into num from birth\_certificate where birth\_certificate\_id = birth\_certificate.Birth\_Certificate\_Id;

if num > 0

then set existance = true;

end if;

return existance;

end$$

delimiter ;

## Б.33 Функція is\_password\_correct

drop function if exists is\_password\_correct;

delimiter $$

create function is\_password\_correct(username varchar(12), user\_password varchar(32)) returns bool deterministic

begin

declare existance bool default false;

declare num int default 0;

select count(users.User\_Name) into num from users where username = users.User\_Name and user\_password = users.User\_Password;

if num > 0

then set existance = true;

end if;

return existance;

end$$

delimiter ;

## Б.34 Функція is\_passport\_issued

drop function if exists is\_passport\_issued;

delimiter $$

create function is\_passport\_issued(passport\_id char(9)) returns bool deterministic

begin

declare exception condition for sqlstate '45000';

declare is\_issued bool default false;

declare date\_of\_issue date default null;

declare validity\_date date default null;

declare agency\_id char(4) default null;

if not is\_passport\_exists(passport\_id)

then signal exception set message\_text = "Не вірний номер паспорту";

end if;

select passport.Date\_of\_Issue, passport.Validity\_Date, passport.Agency\_Id into date\_of\_issue, validity\_date, agency\_id

from passport where passport.Passport\_Id = passport\_id;

if date\_of\_issue = null or validity\_date = null or agency\_id = null

then set is\_issued = false;

else

set is\_issued = true;

end if;

return is\_issued;

end$$

delimiter ;

## Б.35 Функція is\_reference\_from\_usr\_exists

drop function if exists is\_reference\_from\_usr\_exists;

delimiter $$

create function is\_reference\_from\_usr\_exists(record char(13)) returns bool deterministic

begin

declare exception condition for sqlstate '45000';

declare is\_exists bool default false;

declare num int default 0;

select count(\*) into num

from reference\_from\_usr where reference\_from\_usr.Record\_Id = record;

if num > 0

then set is\_exists = true;

end if;

return is\_exists;

end$$

delimiter ;

## Б.36 Функція is\_reference\_from\_usr\_already\_archivated

drop function if exists is\_reference\_from\_usr\_already\_archivated;

delimiter $$

create function is\_reference\_from\_usr\_already\_archivated(record char(13)) returns bool deterministic

begin

declare exception condition for sqlstate '45000';

declare is\_archivated bool default false;

declare num int default 0;

select count(\*) into num

from arch\_reference\_from\_usr where arch\_reference\_from\_usr.Record\_Id = record;

if num > 0

then set is\_archivated = true;

end if;

return is\_archivated;

end$$

delimiter ;

## Б.37 Функція is\_passport\_has\_reference\_from\_usr

drop function if exists is\_passport\_has\_reference\_from\_usr;

delimiter $$

create function is\_passport\_has\_reference\_from\_usr(passport\_id char(9), record\_id char(13)) returns bool deterministic

begin

declare exception condition for sqlstate '45000';

declare is\_has bool default false;

declare reference\_field char(13) default null;

select passport.Record\_Id into reference\_field

from passport where passport.Passport\_Id = passport\_id;

if reference\_field = record\_id

then set is\_has = true;

end if;

return is\_has;

end$$

delimiter ;

## Б.38 Функція is\_taxpayer\_certificate\_exists

drop function if exists is\_taxpayer\_certificate\_exists;

delimiter $$

create function is\_taxpayer\_certificate\_exists(taxpayer\_id char(10)) returns bool deterministic

begin

declare is\_exists bool default false;

declare num int default 0;

select count(\*) into num

from taxpayer\_certificate where taxpayer\_certificate.Taxpayer\_Id = taxpayer\_id;

if num > 0

then set is\_exists = true;

end if;

return is\_exists;

end$$

delimiter ;

## Б.39 Функція is\_passport\_has\_taxpayer\_id

drop function if exists is\_passport\_has\_taxpayer\_id;

delimiter $$

create function is\_passport\_has\_taxpayer\_id(passport\_id char(9), taxpayer char(10)) returns bool deterministic

begin

declare is\_has bool default false;

declare taxpayer\_id\_field char(10) default null;

select passport.Taxpayer\_Id into taxpayer\_id\_field

from passport where passport.Passport\_Id = passport\_id;

if taxpayer\_id\_field = taxpayer

then set is\_has = true;

end if;

return is\_has;

end$$

delimiter ;

## Б.40 Функція is\_birth\_certificate\_exists

drop function if exists is\_birth\_certificate\_exists;

delimiter $$

create function is\_birth\_certificate\_exists(birth\_cert\_id char(9)) returns bool deterministic

begin

declare is\_exists bool default false;

declare num int default 0;

select count(\*) into num

from birth\_certificate where birth\_certificate.Birth\_Certificate\_Id = birth\_cert\_id;

if num > 0

then set is\_exists = true;

end if;

return is\_exists;

end$$

delimiter ;

## Б.41 Функція is\_passport\_has\_birth\_certificate

drop function if exists is\_passport\_has\_birth\_certificate;

delimiter $$

create function is\_passport\_has\_birth\_certificate(passport\_id char(9), birth\_cert\_id char(9)) returns bool deterministic

begin

declare is\_has bool default false;

declare birth\_cert\_id\_field char(9) default null;

select passport.Birth\_Certificate\_Id into birth\_cert\_id\_field

from passport where passport.Passport\_Id = passport\_id;

if birth\_cert\_id\_field = birth\_cert\_id

then set is\_has = true;

end if;

return is\_has;

end$$

delimiter ;

## Б.42 Функція is\_marriage\_certificate\_exists

drop function if exists is\_marriage\_certificate\_exists;

delimiter $$

create function is\_marriage\_certificate\_exists(passport\_id char(9)) returns bool deterministic

begin

declare is\_exists bool default false;

declare num int default 0;

declare another\_passport char(9) default null;

select count(\*) into num

from marriage\_certificate where marriage\_certificate.First\_Passport\_Id = passport\_id or marriage\_certificate.Second\_Passport\_Id = passport\_id;

if num > 0 then

set is\_exists = true;

end if;

return is\_exists;

end$$

delimiter ;

## Б.43 Функція count\_of\_property\_certificates

drop function if exists count\_of\_property\_certificates;

delimiter $$

create function count\_of\_property\_certificates(passport\_id char(9)) returns int deterministic

begin

declare num int default 0;

select count(\*) into num

from property\_right\_certificate where property\_right\_certificate.Owner\_Passport\_Id = passport\_id;

return num;

end$$

delimiter ;

## Б.44 Функція count\_of\_transport\_certificates

drop function if exists count\_of\_transport\_certificates;

delimiter $$

create function count\_of\_transport\_certificates(passport\_id char(9)) returns int deterministic

begin

declare num int default 0;

select count(\*) into num

from transport\_registration\_certificate where transport\_registration\_certificate.Passport\_Id = passport\_id;

return num;

end$$ delimiter ;