ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської кваліфікаційної роботи

на ступінь вищої освіти бакалавр

на тему: «Розробка сервісу налаштувань прав користувачів у корпоративних додатках мовою C#»

Виконав: студент 4 курсу, групи ПД– 42

спеціальності

121 Інженерія програмного забезпечення

(шифр і назва спеціальності)

Кириллов М.В.

(прізвище та ініціали)

Керівник Трінтіна Н.А.

(прізвище та ініціали)

Рецензент_____

(прізвище та ініціали)

Нормоконтроль _____

(прізвище та ініціали)

Київ – 2022

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти - «Бакалавр»

Спеціальність - 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри

Інженерії програмного

забезпечення

_____О.В. Негоденко

«____»__ _____2022 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Кириллов Микола Вячеславович

(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка сервісу налаштувань прав користувачів у корпоративних додатках мовою C#»

Керівник роботи Трінтіна Наталія Альбертівна , кандидат технічних наук

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

затверджені наказом вищого навчального закладу від "18"лютого 2022року №22.

2. Строк подання студентом роботи 03.06.2022

3. Вихідні дані до роботи:

3.1 Офіційна документація Microsoft.

3.2 Microsoft SQL Server Management Studio.

3.3 Visual Studio.

3.4 DevExpress.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

4.1 Аналіз актуальності та проблематики розроблюваного додатку.

4.2 Аналіз та вибір інструментів для реалізації додатку.

4.3 Проектування додатку.

4.4 Висновки

5. Перелік графічного матеріалу

5.1 Титульний слайд
5.2 Мета, об’єкт, предмет та наукова новизна дослідження
5.3 Актуальність роботи
5.4 Аналоги
5.5 Технічні завдання
5.6 Програмні засоби реалізації
5.7 Зв’язок елементів додатку
5.8 UML діаграма бази даних мобільного додатку
5.9 Апробація результатів дослідження
5.10 Висновки
5.11 Кінцевий слайд

6. Дата видачі завдання 11.04.2022

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	11.04-14.04	Виконано
2	Вивчення та аналіз задачі	15.04-17.04	Виконано
3	Розробка структури додатку	18.04-21.04	Виконано
4	Розробка дизайну та графічних елементів	22.04-25.04	Виконано
5	Програмна реалізація системи	26.04-05.05	Виконано
6	Налагодження програми	05.05-07.05	Виконано
7	Вступ,висновки,реферат	07.05-10.05	Виконано
8	Розробка обов’язкових демонстраційних матеріалів	11.05-15.05	Виконано
9	Попередній захист роботи	16.05-01.06	Виконано
10	Здача роботи	03.06	

Студент _____ Кириллов М.В.
(підпис) (прізвище та ініціали)

Керівник роботи_____ Трінтіна Н.А.

РЕФЕРАТ

Об'єкт дослідження – процес надання/обмеження прав користувачів у корпоративних ІТ-програмах.

Предмет дослідження – додаток, де надаються/ обмежуються права користувачів у корпоративних ІТ-програмах.

Мета дослідження – покращення процесу надання/обмеження прав користувачів у корпоративних ІТ-програмах.

Методи дослідження – У дипломній роботі був проведений аналіз існуючих аналогів, переваг та недоліків програмних інструментів для додатка.

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

Додаток реалізований за допомогою мови програмування C# та SQL з використанням DevExpress.

Зміст

ВСТУП

Оцінка сучасного стану об'єкта розробки. Сьогодні кожна IT – компанія має програмістів, які надають/обмежують права користувачів на доступ до того чи іншого документу.

Об'єкт дослідження – процес надання/обмеження прав користувачів у корпоративних IT-програмах.

Предмет дослідження – додаток, де надаються/ обмежуються права користувачів у корпоративних IT-програмах.

Мета дослідження – покращення процесу надання/обмеження прав користувачів у корпоративних IT-програмах.

Наукова новизна проекту – створення унікального додатку для надання/обмеження прав користувачів у корпоративних IT-програмах.

У дипломному проекті був проведений аналіз додатків-аналогів, та виявлені переваги та недоліки.

Додаток розроблений на мові програмування C# з використанням бібліотеки DevExpress у середовищі розробки Visual Studio 2017. Інтерфейс проектувався у Visual Studio 2017. Елементи інтерфейсу – елементи WinForms та бібліотеки DevExpress. Існуючий механізм управління правами доступу в SQL Server Management Studio – незручний, тому актуальним є додаток, в якому реалізований значно зручніший механізм порівняно з існуючими аналогами.

1.ТЕОРЕТИЧНА ЧАСТИНА

1.1 Загальні уявлення про права користувачів SQL Server

При зберіганні даних в будь-якій СУБД одним з головних завдань користувача є забезпечення безпеки цих даних. Проблема безпеки особливо гостро стоїть у реляційних базах даних, оскільки інтерактивний SQL дозволяє легко отримати доступ до них. Вимоги до системи безпеки в типовій реляційній базі даних досить
різноманітні:

• Доступ до даних окремої таблиці повинен бути дозволений одним користувачам та заборонено іншим.

• Одним користувачам має бути дозволено змінювати дані в деякій таблиці, а іншим – здійснювати лише вибірку даних із неї.

• До ряду таблиць доступ повинен здійснюватись лише до окремих сербців.

• Певним користувачам повинно бути заборонено звернення до якоїсь таблиці за допомогою інтерактивного SQL, але дозволено користуватися прикладними програмами, що змінюють цю таблицю.

Керування привілеями для окремого користувача може бути досить виснажливим. В результаті до стандарту SQL було додано концепцію ролей.
Нагадаємо, що роль — це просто іменований набір привілеїв.
Ролі SQL Server дозволяють групувати імена користувачів разом і керувати дозволами на рівні сервера. Вони відіграють центральну роль у безпеці SQL Server. SQL Server має два типи ролей:

- **Виправлені ролі сервера** , які вбудовані в SQL Server і не дозволяють змінювати дозволи або ролі, визначені користувачем. Нижче ми розглянемо основні фіксовані ролі сервера.
- **Визначені користувачем ролі** , які можна налаштувати відповідно до вимог безпеки вашої організації.

При управлінні базою даних SQL Server важливо дотримуватися правила найменших привілеїв, яке гарантує, що користувачі отримають доступ лише до необхідних даних. Мета — надати користувачам доступ до даних, необхідних для нормальної роботи, але нічого крім цього. Ви можете обмежити та контролювати доступ користувачів, налаштувавши ролі сервера. SQL Server надає три типи ролей, які можна використовувати для обмеження доступу до даних у вашій базі даних: ролі на рівні сервера, ролі на рівні бази даних і ролі на рівні програми.

Ролі на рівні сервера допомагають керувати дозволами для всього екземпляра SQL Server. Фіксовані ролі сервера дозволяють учасникам додавати інших користувачів до тієї ж ролі, але це не стосується ролей сервера, визначених користувачем. SQL Server надає такі фіксовані ролі сервера, починаючи з найменш привілейованих ролей:

- **public** — роль за замовчуванням для принципалів сервера, які не мають певних дозволів на об'єкт, що захищається. Призначайте публічні дозволи лише об’єктам, які можуть бути доступними для всіх користувачів. Ви не можете відкликати публічний дозвіл для жодної ролі сервера.
- **dbcreator —** може змінювати, створювати, скидати або відновлювати бази даних.
- **diskadmin —** може керувати дисковими файлами.
- **bulkadmin —може** виконувати BULK INSERT
- **setupadmin —** може додавати/видаляти зв’язані сервери та запускати Transact-SQL
- **processadmin** —може завершити запущені процеси в екземплярі сервера SQL.
- **securityadmin** — може адмініструвати логіни, може скидати паролі входу на сервер SQL, а також надавати, забороняти або анулювати дозволи на рівні сервера або на рівні бази даних
- **serveradmin —** може змінити конфігурацію сервера та вимкнути його

- **sysadmin** — може виконувати всі дії сервера.

Як і ролі на рівні сервера, у SQL Server є фіксовані ролі на рівні бази даних, і ви можете створювати додаткові ролі, налаштовуючи їх за допомогою операторів GRANT, DENY і REVOKE.

Фіксовані ролі існують незалежно для кожної бази даних у вашому екземплярі SQL Server. Ролі сервера **власника** баз даних дозволяється керувати членством фіксованих ролей бази даних.

Microsoft SQL Server надає такі фіксовані ролі бази даних:

- **db_owner** — дозволяє виконувати всі дії з обслуговування та налаштування бази даних, а також скидати базу даних
- **db_securityadmin —може** змінювати власне членство в ролях і керувати дозволами. Уважно стежте за цією роллю, оскільки вона має можливість підвищувати привілеї.
- **db_accessadmin —** може додавати/вилучати доступ до бази даних для груп і логінів Windows, а також логінів SQL Server
- **db_backupoperator —може** виконувати резервне копіювання бази даних
- **db_ddladmin —може** виконувати команди мови визначення даних (DDL) .
- **db_datawriter —** може додавати, змінювати або видаляти будь-які дані таблиці користувача.
- **db_datareader** —обмежений читанням даних із таблиць користувача
- **db_denydatawriter** — заборонено додавати, змінювати або видаляти дані таблиці користувача
- **db_denydatareader** —не може прочитати будь-які дані в таблиці користувача

Ролі програми полегшують роботу програм із виділеними дозволами, схожими на користувача.

1.2 Методи вирішення проблем, зв’язаних з правами користувачів

Забезпечення захисту даних від несанкціонованого використання з самого початку розглядалося як одна з основних вимог до систем управління базами даних. Побудова будь-якої моделі захисту починається з поняття принципала (англійський термін — principal). Принципал має право дозволяти доступ до інформаційної системі іншим особам, які у результаті стають користувачами цієї інформаційної системи. Тільки користувачі мають можливість роботи із системою. Щоб розмежувати можливості різних користувачів, вводяться поняття об'єкта і дії. Дії можуть бути як пов'язані з об'єктом (наприклад, методи об'єкта) або з класом об'єктів, так і не пов'язані. На цьому рівні абстракції зв'язок між діями та об'єктами не має значення.
Зазвичай від імені принципала діє адміністратор, який для доступу до програмної системи (наприклад, операційної системи або системи управління базами даних) використовує особливий ідентифікатор користувача, який називається суперкористувачем.

Суперкористувач може виконувати будь-які дії та має необмежений доступ до всіх об'єктів. Будь-який інший користувач може виконувати тільки ті дії і тільки над тими об'єктами, на які йому надано право. Для входу до бази даних користувач повинен мати обліковий запис користувача Windows або реєстраційне ім'я входу в SQL Server. Для подальшого доступу та роботи з певною базою даних користувач також повинен мати обліковий запис користувача бази даних. Для роботи з кожною окремою базою даних потрібно мати обліковий запис користувача саме для цієї бази даних. Ролі програми дозволяють примусово забезпечувати безпеку для певної програми. Іншими словами, ролі програми дозволяють додатку взяти на себе відповідальність за автентифікацію користувача, замість того, щоб це робила система баз даних. Наприклад, якщо службовці компанії можуть змінювати дані про співробітників тільки за допомогою будь-якої програми (а не за допомогою інструкцій мови Transact-SQL або будь-якого іншого засобу), для цієї

програми можна створити роль програми.

Ролі програм істотно відрізняються від усіх інших типів ролей. По перше, ролі додатків не мають членів, оскільки вони використовують лише додатки, і тому їм немає необхідності надавати дозволи безпосередньо користувачам. По-друге, для активації ролі програми потрібен пароль.
Коли програма активує для сеансу роль програми, цей сеанс втрачає
всі дозволи, які застосовуються до імен введення, облікових записів
користувачів, груп користувачів або ролям у всіх базах даних. Так як ці ролі застосовуються тільки до бази даних, в якій вони знаходяться, сеанс може отримати доступ до іншої бази даних лише за допомогою дозволів, наданих користувачеві guest бази даних, до якої потрібен доступ. Тому, якщо база даних не має користувача guest, сеанс не може отримати доступу до цієї бази даних.

Ідентифікація користувачів

Кожному користувачеві в реляційній базі даних присвоюється ідентифікатор - коротке ім'я, що однозначно визначає користувача для програмного забезпечення СУБД. Ці ідентифікатори є основою системи безпеки. Кожна інструкція SQL виконується в СУБД від імені конкретного користувача. Від його ідентифікатора залежить, чи буде дозволено або заборонено виконання конкретної інструкції. У промисловій базі даних ідентифікатор користувача призначається адміністратором. У базі даних на персональному комп'ютері може бути лише один ідентифікатор користувача, який позначає користувача, який створив базу даних і є її власником.

Аутентифікація користувачів

У більшості комерційних реляційних СУБД ідентифікатор користувача створюється для кожного сеансу (сесії) зв'язку з базою даних. В

інтерактивному режимі сеанс починається, коли користувач запускає інтерактивну програму формування запитів, і продовжується до тих пір, поки користувач не вийде з програми або не введе команду зміни користувача. У додатку, що використовує програмний SQL, сеанс починається, коли програма підключається до СУБД, і закінчується після завершення програми. Протягом сеансу всі інструкції SQL асоціюються з ідентифікатором користувача, встановленим для цього сеансу. Однак у сучасних прикладних системах можливо також, що програма встановлює кілька з'єднань з базою даних і вибирає одне з них для передачі інструкцій SQL.

Як правило, на початку сеансу необхідно ввести як ідентифікатор користувача, так і пов'язаний з ним пароль. Пароль служить для підтвердження того, що користувач дійсно має право працювати під введеним ідентифікатором.

Групи користувачів

У великих виробничих базах даних часто є групи користувачів зі схожими завданнями. Наприклад, у навчальній базі даних три особи у відділі обробки замовлень утворюють природну групу користувачів; дві людини у фінансовому відділі утворюють іншу природну групу. В межах кожної групи всі користувачі р-ботають з однаковими даними і повинні мати ідентичні привілеї.

Відповідно до стандарту ANSI/ISO, з групами користувачів можна надійти одним із трьох способів:

• Кожному члену групи можна призначити той самий ідентифікатор користувача. Це спрощує керування системою безпеки, оскільки дозволяє встановити привілеї доступу до даних один раз (у зв'язку з тим, що ідентифікатор користувача один).

Однак у цьому випадку людей, які спільно використовують один ідентифікатор користувача, не можна буде розрізнити ні на дисплеї системного оператора, ні у звітах СУБД.

• Всім членам групи можна надати різні ідентифікатори користувача. Це дозволить вам диференціювати користувачів у звітах
СУБД і встановлюватиме надалі різні привілеї для окремих користувачів. Однак привілеї доведеться встановлювати для кожного користувача індивідуально, що може бути стомлюючим і збільшує ймовірність виникнення помилок.

• У тих, найбільш сучасних, СУБД, які помержують цю можливість, можна створити роль, яка містить необхідні привілеї. Роль є іменовану колекцію привілеїв. Можна назначити кожному користувачеві його власний ідентифікатор і пов'язати з ним певну роль. Очевидно, що це найкращий вибір, оскільки можна розрізняти користувачів, не ускладнюючи при цьому адміністрування.

Привілеї

Безліч дій, які користувач має право виконувати над об'єктом бази даних, називається привілеями користувача щодо даного об'єкту. У стандарті SQL 1 для таблиць та уявлень визначено чотири привілеї:

• Привілей SELECT дозволяє витягувати дані з таблиці або подання. Маючи цей привілей, можна задавати ім'я таблиці або подання у реченні FROM інструкції SELECT або підзапиту.

• Привілей INSERT дозволяє вставляти нові записи в таблицю або уявлення. Маючи цей привілей, можна задавати ім'я таблиці або подання у реченні INTO інструкції INSERT.

• Привілей DELETE дозволяє видаляти записи з таблиці або подання. Маючи цей привілей, можна задавати ім'я таблиці або подання у реченні FROM інструкції DELETE.

• Привілей UPDATE дозволяє модифікувати записи в таблиці або поданні. Маючи цей привілей, можна задавати таблицю або представлення в інструкції UPDATE як цільової таблиці. Привілей

UPDATE може бути обмежена окремими стовпцями таблиці або
уявлення, дозволяючи цим оновлювати лише ці стовпці і забороняючи оновлювати інші.

1.3 Актуальність

У сучасних умовах діяльність будь-якої організації пов'язана з оперуванням великим обсягом інформації, доступ до якої має широке коло осіб. Наслідком збільшеного останнім часом значення інформації стали високі вимоги до конфіденційності, цілісності та доступності даних.

Системи управління базами даних (СУБД), особливо реляційні СУБД та експертні системи (ЕС), стали домінуючим інструментом у сфері зберігання, обробки та подання даних. Будь-який збій у роботі СУБД (ЕС), що супроводжується втратою, хоч і тимчасової, доступу до даних, негайно відбивається на конкурентної спроможності підприємства. Тому захист даних від несанкціонованого доступу, від несанкціонованої модифікації або просто від їх руйнування є одним із пріоритетних завдань при проектуванні будь-якої інформаційної системи.

Зараз більше уваги приділяється безпеці баз даних, ніж у минулому, оскільки обсяг даних, що зберігаються в корпоративній базі даних, збільшується, і люди все більше залежать від корпоративних даних для прийняття рішень, управління обслуговуванням клієнтів, управління ланцюжками постачання тощо. Будь-яка втрата чи недоступність корпоративних даних паралізує сьогоднішню організацію та серйозно вплине на її продуктивність. Тепер недоступність бази даних навіть на кілька хвилин може призвести до серйозних втрат для організації.

Швидкість створення даних перевищує встановлене сховище. У 2020 році International Data Corporation (IDC) повідомила, що було створено або скопійовано 64,2 ЗБ даних. Дослідники пов'язують це різке зростання зі світовим попитом на цифрові послуги протягом року. Не всі дані, створені в

2020 році, були збережені, але IDC припускає, що існує достатньо доказів, що зберігання більшої кількості даних може принести користь підприємствам. Доступ до конфеденційної інформації повинен бути лише у тих користувачів, які мають відповідні права на роботу з даною інформацією.

Інакше з'являється така проблема, як виток інформації, коли користувачі отриають доступ до тих чи інших документів, не маючи на те жодних прав.

Проблема витоку інформації є досить розповсюдженою.

Рис 1

22 листопада 2019 аналітичний центр компанії InfoWatch опублікував результати глобального дослідження витоків конфіденційної інформації в першому півріччі 2019 року. У цей період аналітиками було зареєстровано 1276 випадків витоків конфіденційної інформації, з яких 55,6% відбулися внаслідок внутрішніх порушень, а 44,4% через зовнішній вплив. Сукупна кількість скомпрометованих записів даних перевищила показник першого півріччя 2018 року більш ніж у 3,6 рази і склала 8,74 млрд записів.

Рис 2

Сергій Хайрук, провідний аналітик ГК InfoWatch, відзначає, що повідомлення про великі інциденти, пов'язані з витоком інформації обмеженого доступу, у досліджуваному періоді з'являлися у ЗМІ практично щодня. Причому від порушень страждають компанії різного масштабу. Найчастіше йдеться про такі світові бренди, як Airbus, Apple, Facebook, Samsung. У порівнянні з аналогічним періодом минулого року зростання витоків конфіденційної інформації склало 22%, а збитки в ряді випадків можна оцінити в сотні мільйонів доларів. За даними експертів, за 2019 рік у світі зафіксовано 218 витоків конфіденційної інформації у фінансовому секторі, що на 7,9% більше, ніж на рік раніше. Така кількість інцидентів становить 8,7% від усіх витоків, зареєстрованих у світі у 2019 році.

6 квітня 2022 року компанія ГК InfoWatch опублікувала дослідження витоків інформації обмеженого доступу, зареєстрованих у 2021 році у всьому світі. Усього за аналізований період у відкритих джерелах було 1729 випадків витоку конфіденційної інформації з компаній та держорганів, що на 28,1% менше, ніж за аналогічний період 2020 року (2406 випадків).

Скомпрометованими виявилися 8,42 млрд записів персональних (ПДН) та платіжних даних, що на 28,8% менше, ніж у 2020 році, коли кількість записів, що втекли, становила понад 11,82 млрд (за уточненими даними). Кількість записів також виявилася меншою, ніж в аномальному за цим показником 2019 (15,1 млрд), але більше, ніж у 2018 (7,24 млрд).

Рис 3

Рис 4

Отже, розробка корпоративного додатку для налаштування прав користувачів є досить актуальною, зважаючи на проблему витоку інформації. Він дозволить співробітникам надавати відповідні права користувачам на доступ до відповідних документів.

1.4 Аналіз наявних додатків

На даний момент налаштовувати права користувачів можливо лише в SQL Server Management Studio.

Додавання члена до визначеної ролі сервера

1. У Провіднику об'єктів розгорніть сервер, для якого потрібно змінити фіксовану роль сервера.
2. Розгорніть папку **Безпеки** .
3. Розгортання папки **"Ролі сервера**"
4. Клацніть правою кнопкою миші роль, яку потрібно змінити, і виберіть пункт **Властивості**.
5. У діалоговому вікні **Властивості ролі сервера -** *Ім'я ролі* сервера на сторінці **Учасники** натисніть кнопку **Додати**.
6. У діалоговому вікні **Вибір входу або ролі сервера** в розділі **Введіть імена об'єктів для вибору (приклади)** введіть роль входу або сервера, щоб додати до цієї ролі сервера.
7. Натисніть**ОК**.

Додавання учасника до ролі бази даних, визначеної користувачем

1. У Провіднику об'єктів розгорніть сервер, для якого потрібно змінити визначену користувачем роль бази даних.
2. Розгорніть папку **Бази даних** .
3. Розгорніть базу даних, в якій потрібно змінити визначену користувачем роль бази даних.
4. Розгорніть папку **Безпеки** .
5. Розгорніть папку **"Ролі"** .
6. Розгорніть папку **Ролі сервера** .
7. Клацніть правою кнопкою миші роль, яку потрібно змінити, і виберіть пункт **Властивості**.

8. У діалоговому вікні **Властивості ролей бази даних - база даних**_*RoleName* на сторінці **Загальні** натисніть кнопку **Додати**.
9. У діалоговому вікні **Вибір користувача бази даних або ролі** в розділі **Введіть імена об'єктів для вибору (приклади)** введіть роль входу або бази даних, щоб додати до цієї ролі бази даних. *Database_role_name*.
10. Натисніть кнопку **OK**.

Налаштовуючи ролі у даному середовищі розробки, я дійшов висновку, що даний механізм керування ролями є незручним, коли у вас на сервері зберігаються багато баз даних з великою кількістю таблиць.

1.5 Складання технічного завдання

Під час дослідження було вирішено створити додаток на ПК. По-перше,спочатку, ми повинні обрати мову програмування, за допомогою якої буде написано дану програму. Додаток можна написати за допомогою платформи .NET.

Платформа .NET Framework – це технологія, яка підтримує створення та виконання веб-служб та програм Windows.

Фреймворк .NET є потужною платформою для створення додатків. Можна виділити такі основні риси:

- **Підтримка декількох мов**. Основою платформи є спільна мова виконання (CLR), так що .NET підтримує таку мову, як C#, а також різні діалекти інших мов, які прив'язані до .NET, такі як Delphi.NET. При складанні код на будь-якій з цих мов складається в збірку спільною мовою CIL (Common Intermediate Language) - свого роду складання платформи .NET. Тому за певних умов ми можемо зробити окремі модулі однієї програми окремими мовами.

- **Різноманітні** технології. Спільна мова виконання (CLR) і бібліотека основного класу є основою цілого стека технологій ADO.NET, які розробники можуть використовувати для створення додатків. Форми.

ADO.NET є технологією роботи з даними, яка заснована на платформі .NET Framework. Ця технологія представляє нам набір класів, через які ми можемо надсилати запити до баз даних, встановлювати підключення, отримувати відповідь від бази даних та здійснювати низку інших операцій. Функціонал ADO.NET побудований таким чином, щоб надати розробникам уніфікований інтерфейс для роботи з різними СУБД.

Windows Forms — це технологія інтерфейсу для .NET, що представляє собою набір керованих бібліотек, які спрощують виконання стандартних завдань, таких як читання з файлової системи і запис в неї.Платформа розробки Windows Forms підтримує широкий набір функцій для розробки програм, включаючи елементи керування, графіку, прив'язку даних та введення користувача. Характерною рисою Windows Forms є використання візуального конструктора з функцією перетягування Visual Studio для спрощення створення програм Windows Forms. У Windows Forms можна розробляти графічно складні програми, які легко розгортати, оновлювати, і з якими зручно працювати як в автономному режимі, так і в мережі. Програми Windows Forms можуть отримати доступ до локального обладнання та файлової системи комп'ютера, на якому працює програма.

Мені потрібно, щоб обрана мова програмування підходила до моїх вимог.

Вимоги до додатка:

- авторизація користувача
- наявність фунції перегляду даних
- наявність фунції додавання даних
- наявність фунції модифікації даних
- наявність фунції видалення даних

- можливість надання прав користувачу на перегляд/додавання/модифікацію/видалення даних

Аналізуючи вимоги до розробки даного додатку, було вирішено його створювати за допомогою платформи .NET Framework. Найкращим варіантом виявилася мова програмування C# з використанням WinForms та ADO.NET.

2. ОПИС ПРОГРАМНИХ ЗАСОБІВ

2.1 Мова програмування C# та .NET

C# - мова програмування, що поєднує об'єктно-орієнтовані та аспектно-орієнтовані концепції. Розроблена в 1998-2001 роках групою інженерів під керівництвом Андерс Хейлсберг в компанії Microsoft як основна мова розробки додатків для платформи Microsoft .NET. C# відноситься до широко відомого сімейства мов C, і здасться добре знайомим будь-кому, хто працював з C, C++, Java або JavaScript.

Дана мова програмування є досить розповсюдженою при створенні комп’ютерних та веб-додатків. Дана мова має величезну кількість користувачів, і як наслідок, користувачі можуть, одне одному, допомогти з труднощами, які трапляються при розробці. Постійно створюється нове програмне забезпечення та інструменти, внаслідок чого якість мови неухильно зростає. За допомогою об'єктно-орієнтованого підходу ми можемо легко модифікувати великі за обсягом програми. Ця мова програмування підтримує такі принципи об'єктно-орієнтованого програмування, як інкапсуляція, поліморфізм та успадкування. З виходом нових версій до функціоналу C# додаються такі можливості, як асинхронність, багатопоточність, лямбда вирази, обробка виключень та багато іншого.

За допомогою C# також можливо створити повнофункціональний веб-сайт, який, за наявності об'єктно-орієнтованого підходу, буде легко обслуговуватись.

C# є складовою компонентою, без якої неможливо було б створити безліч існуючих ігор. Він поєднується з ігровим рушієм Unity, який є найпопулярнішим інструментом для розробки відеоігор. Такому успіху у створенні різноманітних ігор C# сприяє об’єктно-орієнтований підхід, інтерфейси, автоматичне керування оперативною пам'яттю комп'ютера за допомогою збирання сміття.

Переваги C#:

- C# - це об'єктно-орієнтована, проста, але потужна мова програмування, яка дозволяє розробникам створювати багаті програми.
- C# відноситься до мов скомпільованого типу, тому має всі переваги таких мов.
- C# об'єднує найкращі ідеї сучасних мов програмування Java, C++, Visual Basic тощо.
- Завдяки широкому різноманіттю синтаксичних конструкцій і можливості роботи з платформою .Net, C# дозволяє розробляти програмні рішення швидше, ніж будь-яка інша мова.
- Більшої надійності вдалося досягти завдяки роботі машини CLR, адже на відміну від інших компіляторів CLR, розроблений додаток працює на віртуальному процесорі. Тому в разі будь-яких помилок це не позначиться на роботі інших програм в системі, але це також означає, що для запуску програми потрібен додатковий час. Відповідно, програми, написані на мові програмування C#, надійніші, але менш швидкі (ніж ті ж програми, написані на C++).
- підтримка переважної більшості продуктів Microsoft.
- індивідуальні розробники та невеликі компанії мають здатність безкоштовно користуватися багатьма інструментами, наприклад Azure, Mac Pro, Windows Server, Visual Studio.

Технології платформи .NET, наприклад Windows Forms, WPF тісно пов’язані між собою, але не треба розглядати їх як одне й теж.

Середовище .NET Framework має на меті підтримку створення і виконання сильно розподілених компонентних додатків. Він надає можливість використовувати створені API-інтерфейси для створення комп’ютерних додатків як WPF, для роботи з базами даних як ADO.NET, та інших. Додатки створенні за допомогою .Net є кросплатформними програмами.

Платформа .NET Framework – це відповідь компанії Microsoft на іншу

платформу, Java, що мала велику популярність у той час, коли була створена компанією Sun Microsystems.

Засновник корпорації Microsoft Білл Гейтс визнав, що найкращий винахід його компанії – це .NET. Це висловлювання не було даремним, бо за допомогою потужної платформи .NET Framework створюються чудові додатки.

Переваги .NET:

- в цій програмі легко переробити вже створені додатки;
- досить великий вибір корисних інструментів, які спрощують створення додатків і скорочують час їх розробки;
- наявність кросплатформної функції.

2.2 WinForms

Windows Forms - це платформа інтерфейсу користувача для створення класичних додатків Windows. Вона забезпечує один з найефективніших способів створення класичних програм за допомогою візуального конструктора в Visual Studio. Такі функції, як розміщення візуальних елементів керування шляхом перетягування, полегшують створення класичних додатків. У Windows Forms можна розробляти графічно складні програми, які легко розгортати, оновлювати, і з якими зручно працювати як в автономному режимі, так і в мережі. Програми Windows Forms можуть отримати доступ до локального обладнання та файлової системи комп'ютера, на якому працює програма.

У Windows Forms передбачено безліч елементів керування, які можна додавати до форм. Наприклад, елементи керування можуть відображати текстові поля, кнопки, списки, перемикачі і навіть веб-сторінки. Якщо передбачені елементи керування не підходять для ваших цілей, у Windows

Forms можна створювати власні елементи керування за допомогою класу UserControl.

У Windows Forms є багатофункціональні елементи керування інтерфейсу користувача, що дозволяють емулювати функції таких складних програм, як Microsoft Office. За допомогою елементів керування ToolStrip та MenuStrip ви можете створювати панелі інструментів та меню, які містять текст та зображення, відображають підменю та розміщують інші елементи керування, такі як текстові поля та поля зі списками. У багатьох програмах потрібно відображати дані з бази даних, файлу XML або JSON, веб-служби або іншого джерела даних. Windows Forms надає гнучкий елемент керування з ім'ям DataGridView для відображення таких табличних даних у традиційному форматі рядків і стовпців так, що кожен фрагмент даних займає свій власний осередок. За допомогою DataGridView можна також налаштувати зовнішній вигляд окремих осередків, зафіксувати рядки і стовпці на своєму місці, а також забезпечити відображення складних елементів управління всередині осередків.

У Windows Forms можна легко підключатися до джерел даних через мережу. Компонент BindingSource представляє підключення до джерела даних та містить методи для прив'язки даних до елементів управління, переходу до попереднього або наступного запису, редагування записів та збереження змін у вихідному джерелі. Елемент управління BindingNavigator надає простий інтерфейс на основі компонента BindingSource для переходу між записами.

Для роботи з Winforms потрібно добре знати C# та компоненти WinForms. Навіть якщо розробник може досконало освоїти, наприклад C#, він усе одно може зробити проект невдалим, якщо він не знає принципів розробки кожної платформи.

2.3 DevExpress

DevExpress - це всеосяжний набір інструментів звітності для платформи

.NET, що дає розробникам можливість задіяти у своїх додатках дані з будь-яких джерел завдяки підтримці об'єктів даних Visual Studio.

Компоненти DevExpress Reporting Subscription:

- Елементи керування звітами для WinForms. Представлено XtraReports Suite для WinForms (перегляд і створення звітів).
- Елементи керування звітом для WPF. Представлені XtraReports Suite для WPF з підтримкою середовища розробки Visual Studio, вони призначені для легкого створення / розповсюдження звітів.
- Елементи керування звітами для ASP.NET веб-форм.
- Елементи керування звітами для ASP.NET MVC.
- Елементи керування звітом для ASP.NET Core.

2.4 T-SQL

Transact-SQL (T-SQL) є основною мовою, яка використовується для маніпулювання та обробки даних у Microsoft SQL Server. T-SQL є основною мовою управління та обробки даних у реляційній системі управління базами даних Microsoft (RDBMS) – SQL Server, як при установці на сайті клієнта, так і в «хмарній» моделі (база даних Microsoft Windows Azure SQL). T-SQL є діалектом стандартної мови SQL, який, в свою чергу, є одночасно стандартом Міжнародної організації зі стандартів (ISO) і Американського національного інституту стандартів (ANSI).

Ви зможете використовувати структуру оператора SELECT і використовувати для виконання завдань, зв’язаних з маніпуляцією даних, наприклад, фільтрацією та сортуванням. T-SQL має міцні математичні основи. Вам не потрібно бути математиком, щоб написати хороший SQL-код (хоча це не завадить), але з чітким розумінням основних понять мови ви можете краще зрозуміти саму мову. Основні елементи мови виглядають однаково. Однак кожен постачальник вирішує, які функції реалізувати, а які не реалізувати. Крім того, стандарт дозволяє вибирати певні аспекти при монтажі. Кожен

постачальник також зазвичай впроваджує розширення до стандарту, якщо вони вважають, що він не охоплює деякі важливі функціональні можливості. Дотримання стандарту при написанні коду вважається найкращим рішенням. Це робить ваш код більш портативним, а ваші знання стають більш портативними, тому що вам не складе труднощів почати працювати на нових платформах. Якщо діалект, на якому ви пишете, підтримує як стандартні, так і нестандартні способи виконання будь-яких операцій, завжди слід вибирати стандартний спосіб. наприклад, T-SQL підтримує кілька функцій для перетворення вихідного значення на кінцевий (цільовий) тип даних. Якщо ви хочете використовувати аргумент стилю.

Ще одним прикладом вибору стандартної форми є завершення операторів T-SQL. Згідно зі стандартним SQL, оператори повинні закінчуватися крапкою з комою. В даний час T-SQL не вимагає цього для всіх операторів, тільки в тих випадках, коли можливі неоднозначні елементи коду, наприклад, в реченні WITH загального виразу таблиці (CTE). Необхідно дотримуватися стандарту і закінчувати всі твердження крапкою з комою, навіть якщо це не потрібно поточній версії. В основі стандартної мови SQL лежить реляційна модель, яка є математичною моделлю для управління і обробки даних Насправді у мові T-SQL більший акцент зроблено на теорію мультимножин, ніж теорію множин. Мультимножина у багатьох відношеннях дуже схожа на множину, але дозволяє появу дублікатів. T-SQL надає достатньо інструментів, які дозволяють при бажанні слідувати реляційної теорії. Наприклад, дана мова має оператор DISTINCT, який дозволяє видалити дублікати.

```
SELECT DISTINCT country
FROM HR.Employees;
```

І ось результат відредагованого запиту:

```
Country
---------------
UK
USA
```

Інший найважливіший аспект множини — порядок елементів — не має значення. Тому в теорії рядки в таблиці йдуть в довільному порядку. Потім,

якщо ви запитуєте таблицю і явно не вказуєте, що ви хочете повернути рядки в певному порядку, результат вважається реляційним. Таким чином, ви не очікуєте отримати результат з будь-яким конкретним порядком рядків, незалежно від того, що ви знаєте про фізичне представлення даних, наприклад, коли дані індексуються. Якщо ви все-таки хочете ввести певний порядок представлення рядків в результаті, вам потрібно додати речення ORDER BY в запит наступним чином:

```
SELECT empid, lastname
FROM HR.Employees
ORDER BY empid;
```

У цьому випадку результат не реляційний - це те, що стандартний SQL називає курсором. Порядок рядків у результатах запиту гарантується атрибутом empid. Результатом цього запиту є

```
empid   lastname
------  -------------
5       Buck
8       Cameron
1       Davis
9       Dolgopyatova
2       Funk
7       King
3       Lew
4       Peled
6       Suurs
```

Заголовок зв'язку — це набір атрибутів, які потрібно визначити за іменем та іменем типу. Немає порядку для цих атрибутів. На відміну від цього, T-SQL відстежує порядкові позиції стовпців на основі порядку, в якому вони відображаються у визначенні таблиці. Під час створення запиту за допомогою SELECT*, ви гарантуєте, що результат запиту поверне стовпці в порядку визначення. T-SQL також дозволяє посилатися на порядкові позиції стовпців результату в реченні ORDER BY наступним чином:

```
SELECT empid, lastname
FROM HR.Employees
ORDER BY empid;
```

T-SQL дозволяє запиту повертати кілька стовпців результату з однаковим ім'ям. T-SQL має як фізичну сторону, так і логічну сторону. Логічною стороною є концептуальна інтерпретація запиту, яка пояснює, який

правильний результат запиту. Фізична сторона - обробка запиту движком баз даних (Обробник баз даних). Фізична обробка повинна дати результат, визначений логічною обробкою запиту. Оптимізація може змінити послідовність логічних етапів обробки запиту або видалити їх взагалі, але тільки в тому випадку, якщо результат залишається тим, що визначається логічною обробкою запиту. T-SQL, який базується на стандартному SQL, є декларативною мовою, схожою на англійську. У цій мові "декларативний" означає, що ви визначаєте, що вам потрібно, на відміну від імперативних мов, які також визначають, як досягти того, що вам потрібно. Стандартний SQL описує логічну інтерпретацію декларативного запиту (частина "що"), тоді як з'ясування того, як фізично обробити запит (частина "як") відповідає на обробник баз даних.

Як і в мовах програмування, SQL має різні типи даних для зберігання змінних:

- Числа — для хранения числовых переменных (bit, int, tinyint, smallint, bigint, numeric, decimal, money, smallmoney, float, real).
- Дати — для зберігання дати й часу (дата й час (дата й час, малий час, дата, дата, дата й час2, дата йостеп).
- Символи — для зберігання даних символів (char, nchar, varchar, nvarchar).
- Двійковий — для зберігання двійкових даних (двійкових, варбінарних, зображень).
- Великотомні типи даних для зберігання великих двійкових даних (текст, ntext, зображення).
- Special - покажчики (курсор), 16-байтове шістнадцяткове число, яке використовується для GUID (uniqueidentifier), штамп зміни рядка (мітка часу), rowversion (rowversion), таблиці (таблиця).

Для використання російських символів (а не кодування ASCII) використовуються типи даних з приставкою «n» (nchar, nvarchar, ntext), які кодують символи з двома байтами. Іншими словами, для роботи з Юнікодом використовуються типи даних з «n» (від слова national). Струнні константи з Юнікодом також пишуться з "n" на початку.

Для даних змінної довжини використовуються типи даних з приставкою «var». Типи даних без префікса «var» мають фіксовану довжину простору пам'яті, невикористана частина якого заповнюється пробілами або нулями.

Ідентифікатори - це спеціальні символи, які використовуються зі змінними для ідентифікації їх типу або групування слів у змінну. Типи ідентифікаторів:

- @ - ідентифікатор локальної змінної (змінної користувача).
- @@ — ідентифікатор глобальної змінної (вбудованої).
- # - ідентифікатор локальної таблиці або процедури.
- ## - це ідентифікатор глобальної таблиці або процедури.
- [ ] — ідентифікатор для групування слів у змінну (робота як стандартна » »).

Директиви сценарію - це конкретні команди, які використовуються тільки в MS SQL. Ці команди допомагають серверу визначити правила роботи зі сценарієм і транзакціями. Типові представники: GO - інформує утиліти SQL Server про закінчення партії операторів Transact-SQL, EXEC (або EXECUTE) - виконує процедуру або скалярну функцію.

Коментарі використовуються для створення пояснень для блоків сценаріїв, а також для тимчасового відключення команд при налагодженні сценарію. Коментарі можуть бути як рядковими, так і блоковими:

`--` — Лінійний коментар виключає з виконання лише один рядок, якому передують два мінуси.

`/* */` — Блокова коментатор виключає з виконання весь блок команд, укладених у вказану конструкцію.

Оператори - це спеціальні команди, призначені для виконання простих операцій над змінними:

- Арифметичні оператори: "*" - множення, "/" - розділити, "%" - залишок ділення, "+" - додати, "-" - відняти, "()" - дужки.
- Оператори порівняння: "=" дорівнює, ">" більше, "<" менше, "> =" більше або дорівнює, "< =" менше або дорівнює, "<>" ("!=") не дорівнює, між (замість ">=", "<=").
- Оператори підключення: "+" — з'єднання (об'єднання) рядків.
- Логічні оператори: "AND" — і , "OR" - або , "NOT" - ні.
- Оператори з наборами: "IN".

Специфікація Transact-SQL значно розширює стандартні можливості SQL з вбудованими функціями:

1. Агрегатні функції - це функції, які працюють з колекціями значень і виробляють єдине значення. Типові представники: AVG - середнє значення стовпця, SUM - сума стовпця, MAX - максимальне значення стовпця, MIN - мінімальне значення стовпця, COUNT - кількість елементів стовпця.
2. Скалярні функції - це функції, які повертають єдине значення, працюють з скалярними даними або взагалі не вводять. Типові представники: DATEDIFF - різниця між датами, ABS - числовий модуль, DB_NAME - назва бази даних, USER_NAME - ім'я поточного користувача, LEFT - частина рядка зліва.
3. Функції вказівника — це функції, які використовуються як посилання на інші дані. Типові представники: OPENXML - покажчик на джерело

даних у вигляді структури XML, OPENQUERY - покажчик на джерело даних в якості іншого запиту.

Вираз - це комбінація символів і операторів, яка отримує скалярне значення як вхідне, а на виході дає інше значення або виконує якусь дію. У Transact-SQL вирази поділяються на 3 типи: DDL, DCL і DML.

1. DDL (Мова визначення даних) - використовується для створення об'єктів в базі даних. Основними представниками цього класу є: CREATE — створення об'єктів, ALTER — зміна об'єктів, DROP — видалення об'єктів.
2. DCL (Мова керування даними) - призначений для присвоєння прав об'єктам бази даних. Основними представниками цього класу є: ГРАНТ - дозвіл на об'єкт, DENY - заборона на об'єкт, REVOKE - скасування дозволів і заборон на об'єкт.
3. DML (Мова маніпулювання даними) – використовується для запиту та зміни даних. Основними представниками цього класу є: SELECT — отримання даних, INSERT — вставка даних, UPDATE — зміна даних, DELETE — видалення даних.

У Transact-SQL існують спеціальні команди, які дозволяють контролювати потік виконання сценарію, перериваючи його або направляючи в потрібну гілку.

1. Блок групування - це структура, яка об'єднує список виразів в один логічний блок (BEGIN ... КІНЕЦЬ).
2. Блок умов - це структура, яка перевіряє виконання певної умови (IF ... ELSE).
3. Блок циклу - це структура, яка організовує повторення виконання логічного блоку (WHILE ... ПЕРЕРВА... ПРОДОВЖИТИ).
4. Transition - це команда, яка переходить потік виконання сценарію на вказану мітку (GOTO).

5. Delay - це команда, яка затримує виконання сценарію (WAITFOR).
6. Виклик помилки - це команда, яка генерує помилку виконання сценарію (RAISERROR).

2.5 ADO.NET

ADO.NET — це набір класів, які надають послуги доступу до даних для програмістів .NET Framework. ADO.NET надає багатий набір компонентів для створення розподілених програм для обміну даними. Це невід'ємна частина .NET Framework, що забезпечує доступ до реляційних даних, даних XML та програм. ADO.NET підтримує різноманітні потреби в розробці, включаючи створення клієнтських баз даних і бізнес-об'єктів середнього рівня, які використовуються програмами, інструментами, мовами або Інтернет-браузерами.

Основу інтерфейсу взаємодії з базами даних у ADO.NET представляє обмежене коло об'єктів: Connection, Command, DataReader, DataSet та DataAdapter. За допомогою об'єкта Connection відбувається встановлення підключення до джерела даних. Об'єкт Command дозволяє виконувати операції з даними БД. Об'єкт DataReader зчитує дані, отримані в результаті запиту. Об'єкт DataSet призначений для зберігання даних із БД та дозволяє працювати з ними незалежно від БД. І об'єкт DataAdapter є посередником між DataSet та джерелом даних. Головним чином через ці об'єкти і йтиме робота з базою даних. Однак, щоб використовувати один і той же набір об'єктів для різних джерел даних, необхідний відповідний провайдер даних. Власне через провайдер даних в ADO.NET здійснюється взаємодія з базою даних. Причому для кожного джерела даних в ADO.NET може бути свій провайдер, який і визначає конкретну реалізацію вищевказаних класів.

За замовчуваннямADO.NET має таких вбудованих постачальників:

- Провайдер для MS SQL Server

- Постачальник баз даних OLE (надає доступ до деяких старих версій MS SQL Server, а також до баз даних Access, DB2, MySQL та Oracle)
- Постачальник odbc (постачальник для тих джерел даних, для яких немає власних постачальників)
- Постачальник для Oracle
- Постачальник entityClient. Постачальник даних для технології сутності Framework ORM
- Постачальник для SQL Server Compact 4.0

Основні простори імен, що використовуються в ADO.NET:

- System.Data: визначає класи, інтерфейси, делегати, що реалізують архітектуру ADO.NET
- System.Data.Common: містить класи, загальні для всіх провайдерів ADO.NET
- System.Data.Design: визначає класи, які використовуються для створення власних наборів даних
- System.Data.Odbc: визначає функціональність провайдера даних для ODBC
- System.Data.OleDb: визначає функціональність провайдера даних для OLE DB
- System.Data.Sql: зберігає класи, які підтримують специфічну для SQL Server функціональність
- System.Data.OracleClient: визначає функціональність провайдера для баз даних Oracle
- System.Data.SqlClient: визначає функціональність провайдера для баз даних MS SQL Server
- System.Data.SqlServerCe: визначає функціональність провайдера для SQL Server Compact 4.0
- System.Data.SqlTypes: містить класи для типів даних MS SQL Server

- Microsoft.SqlServer.Server: зберігає компоненти для взаємодії SQL Server та середовища CLR

Схематично архітектуру ADO.NET можна представити так:

Рис 5

Функціонально класи ADO.NET можна розбити на два рівні: підключений та відключений. Кожен провайдер даних .NET реалізує свої версії об'єктів Connection, Command, DataReader, DataAdapter та інших, що складають підключений рівень. Тобто за допомогою них встановлюється підключення до БД та виконується з нею взаємодія. Як правило, реалізації цих об'єктів для кожного конкретного провайдера у своїй назві мають префікс, який свідчить про провайдер:

Object	SQL Server	OLE DB	ODBC
Connection	SqlConnection	OleDbConnection	OdbcConnection
Command	SqlCommand	OleDbCommand	OdbcCommand
Data reader	SqlDataReader	OleDbDataReader	OdbcDataReader
Data adapter	SqlDataAdapter	OleDbDataAdapter	OdbcDataAdapter

Рис 6

Інші класи, такі як DataSet, DataTable, DataRow, DataColumn та ряд інших складають відключений рівень, тому що після вилучення даних у DataSet ми можемо працювати з цими даними незалежно від того, встановлено підключення чи ні. Тобто після отримання даних із БД програма може бути відключена від джерела даних.

2.6 SQL Server

Microsoft SQL Server - система управління реляційними базами даних (РСУБД), розроблена корпорацією Microsoft. Основна мова запитів — Transact-SQL, створений спільно Microsoft і Sybase. Transact-SQL є реалізацією стандарту ANSI/ISO із структурованої мови запитів (SQL) з розширеннями. Використовується до роботи з базами даних розміром від персональних до великих баз даних масштабу підприємства; конкурує з іншими СУБД у цьому сегменті ринку.

SQL Server характеризується такими особливостями як:

- Продуктивність. SQL Server працює дуже швидко.
- Надійність та безпека. SQL Server надає шифрування даних.
- Простота. З цієї СУБД щодо легко працювати та вести адміністрування.

Центральним аспектом у MS SQL Server, як і будь-який СУБД, є база даних. База даних представляє сховище даних, організованих певним способом. Для організації бази даних MS SQL Server використовує реляційну модель. Реляційна модель передбачає зберігання даних у вигляді таблиць, кожна з яких складається з рядків та стовпців. Кожен рядок зберігає окремий об'єкт, а стовпці розміщуються атрибути цього об'єкта. Для взаємодії з базою даних використовується мова SQL (Structured Query Language). Клієнт (наприклад, зовнішня програма) надсилає запит мовою SQL за допомогою спеціального API. СУБД належним чином інтерпретує та виконує запит, а потім надсилає

клієнту результат виконання. Виділяються два різновиди мови SQL: PL-SQL та T-SQL. PL-SQL використовується в таких СУБД як Oracle та MySQL. T-SQL (Transact-SQL) застосовується до SQL Server.Залежно від завдання, яке виконує команда T-SQL, він може належати до одного з таких типів:

- **DDL** (Data Definition Language/Мова визначення даних). До цього типу відносяться різні команди, які створюють базу даних, таблиці, індекси, процедури, що зберігаються і т.д. Загалом визначають дані.

  Зокрема, до цього типу ми можемо віднести такі команди:

  - **CREATE** : створює об'єкти бази даних (саму базу даних, таблиці, індекси тощо)
  - **ALTER** : змінює об'єкти бази даних
  - **DROP** : видаляє об'єкти бази даних
  - **TRUNCATE** : видаляє всі дані з таблиць
- **DML** (Data Manipulation Language/Мова маніпуляції даними). До цього типу відносять команди з вибору даних, їх оновлення, додавання, видалення - загалом всі команди, з допомогою якими ми можемо управляти даними.

  До цього типу належать такі команди:

  - **SELECT** : отримує дані з БД
  - **UPDATE** : оновлює дані
  - **INSERT** : додає нові дані
  - **DELETE** : видаляє дані
- **DCL** (Data Control Language/Мова керування доступу до даних). До цього типу відносять команди, які керують правами доступу до даних. Зокрема, це такі команди:
  - **GRANT** : надає права на доступ до даних
  - **REVOKE** : відкликає права на доступ до даних

SQL Server підтримує віддзеркалення та кластеризацію баз даних. Кластер сервера SQL – це сукупність однаково конфігурованих серверів; така схема допомагає розподілити робоче навантаження між кількома серверами. Усі сервери мають одне віртуальне ім'я, і дані розподіляються за IP-адресами машин кластера протягом робочого циклу. Також у разі відмови або збою на одному із серверів кластера доступне автоматичне перенесення навантаження на інший сервер.

SQL Server підтримує надмірне дублювання даних за трьома сценаріями:

- Знімок: Здійснює «знімок» бази даних, який сервер надсилає одержувачам.
- Історія змін: Всі зміни бази даних безперервно передаються користувачам.
- Синхронізація з іншими серверами: Бази даних кількох серверів синхронізуються між собою. Зміни всіх баз даних відбуваються незалежно друг від друга кожному сервері, а під час синхронізації відбувається звірка даних. Цей тип дублювання передбачає можливість вирішення протиріч між БД.

У SQL Server вбудована підтримка .NET Framework. Завдяки цьому процедури БД, що зберігаються, можуть бути написані будь-якою мовою платформи .NET, використовуючи повний набір бібліотек, доступних для .NET Framework, включаючи Common Type System (система поводження з типами даних у Microsoft .NET Framework). Однак, на відміну від інших процесів, .NET Framework, будучи базовою системою для SQL Server, виділяє додаткову пам'ять та вибудовує засоби управління SQL Server замість того, щоб використовувати вбудовані засоби Windows. Це підвищує продуктивність порівняно із загальними алгоритмами Windows, оскільки алгоритми розподілу ресурсів спеціально налаштовані для використання у структурах SQL Server.

3. ЗАГАЛЬНІ ВІДОМОСТІ ДОДАТКУ

3.1 Розробка концепції

Аналізуючи актуальність додатку, для найефективнішого розповсюдження було прийняте рішення його розробки на ПК, наступним етапом стане проголошення концепції.

Цільова аудиторія – співробітники компанії, бо вони працюють з документами.

Головна ідея цього додатку також полягає в зручному механізмі надання прав користувачам на перегляд/редагування даних у документах.

Короткий опис застосування – цей додаток націлений на зручне управління правами користувачів. За його допомогою ви маєте можливість надавати чи не давати доступ користувачам до деяких документів або забороняти редагувати відповідні документи. Також ви маєте можливість додавати нових користувачів з відповідними правами. Одним із найважливіших функцій цього додатку є можливість досить легко і швидко налаштовувати права доступу в незалежності від об'єму баз даних.

Головні вимоги додатку:

- Форма авторизації
- зручне меню
- можливість перегляду документів
- зручний інтерфейс

Унікальність додатку:

- унікальний механізм налаштування прав доступу користувачів
- єдиний додаток для налаштування прав доступу користувачів

3.2 Проектування структури та створення додатку

Для розробки проекту додатку на ПК, будо проаналізовано багато інструментів, як наслідок було вирішено використовувати Visual Studio.

Перш за все нам потрібно визначитись із основними елементами інтерфейсу, щоб потім про них не забути.

Головні елементи інтерфейсу додатку:

1. Форма.
2. Логотип додатку.
3. Текстове поле для логіну.
4. Текстове поле для паролю.
5. Кнопка авторизації.
6. Кнопка додання.
7. Кнопка відмови.
8. Кнопка видалення.
9. Кнопка модифікації.
10. Кнопка оновлення.
11. Меню.
12. Кнопка списку додатків.
13. Кнопка списку ролей.
14. Кнопка списку користувачів.
15. Кнопка списку груп сервісів.
16. Кнопка списку сервісів.
17. Кнопка налаштування користувача на роль.
18. Кнопка налаштування користувача на додаток.
19. Кнопка налаштування сервісу на роль.
20. Кнопка повернення.
21. GridControl – таблиця даних.

Визначившись із головними елементами інтерфейсу можна починати працювати у Visual Studio. Для початку роботи з цим інструментом, потрібно запустити її та створити новий проект.

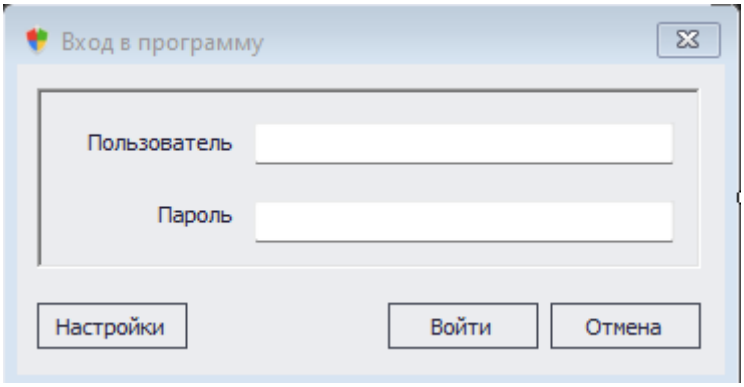


Рис 7

На початковій формі має бути можливість увійти користувачу що вже реєструвався

При натисканні на кнопку входу, повинні переходити на нове вікно, де ми й будемо вводити свій логін та пароль, тож нам потрібно поєднати нову форму з початковою. Це можна реалізувати за допомогою даного коду.

```
private void btnOK_Click(object sender, EventArgs e)
{
    SqlConnectionStringBuilder s = new SqlConnectionStringBuilder();
    s.InitialCatalog = Properties.Settings.Default.InitialCatalog;
    s.DataSource = Properties.Settings.Default.DataSource;
    s.UserID = AppUserName.Text;
    s.Password = AppUserPassword.Text;
    s.MultipleActiveResultSets = true;   // разрешить больше 1 запроса к серверу в рамках 1 конекта

    // Создание подключения
    using (SqlConnection cn = new SqlConnection())
    {
        cn.ConnectionString = s.ConnectionString;
        try
        {
            cn.Open();
            // Ошибки нет - подключение успешно
            // Сохраним имя для следующего входа в программу
            Properties.Settings.Default.AppUserName = AppUserName.Text;
            Properties.Settings.Default.Save();

            // Сохраним параметры подключения для использования в других формах
            DataForms.AppUserName = AppUserName.Text;
            DataForms.AppUserPassword = AppUserPassword.Text;
            DataForms.ConnectionString = Properties.Settings.Default.SQL_Connect_Prod;
            DataForms.IDApp = Properties.Settings.Default.IDApp;
            //Прячем парольную форму и открываем основную

            if (Check_User_App(cn, AppUserName.Text, DataForms.IDApp) == 0)
            {
                Hide();
                MainForm f = new MainForm();

                f.ShowDialog();
                Close();
            }
        }
        catch (SqlException ex)
        {
            MessageBox.Show(ex.Message);
        }
        finally
        {
            cn.Close();
        }
    };
}
```

Створення головної сторінки – це найвідповідальніша частина, бо інтерфейс повинен бути інтуїтивно зрозумілим для нового користувача, щоб він одразу не вийшов із додатку.

Головна сторінка повинна бути багатофункціональною, але у той же час простою, із графічної точки зору. Оскільки це перша сторінка, не враховуючи автентифікацію, то ми повинні створити кнопки для швидкого переходу між основними можливостями додатку.

Основним елементом інтерфейсу головної сторінки буде меню, яке складається з 8 кнопок:

- Кнопка списку додатків.

- Кнопка списку ролей.
- Кнопка списку користувачів.
- Кнопка списку груп сервісів.
- Кнопка списку сервісів.
- Кнопка налаштування користувача на роль.
- Кнопка налаштування користувача на додаток.
- Кнопка налаштування сервісу на роль.

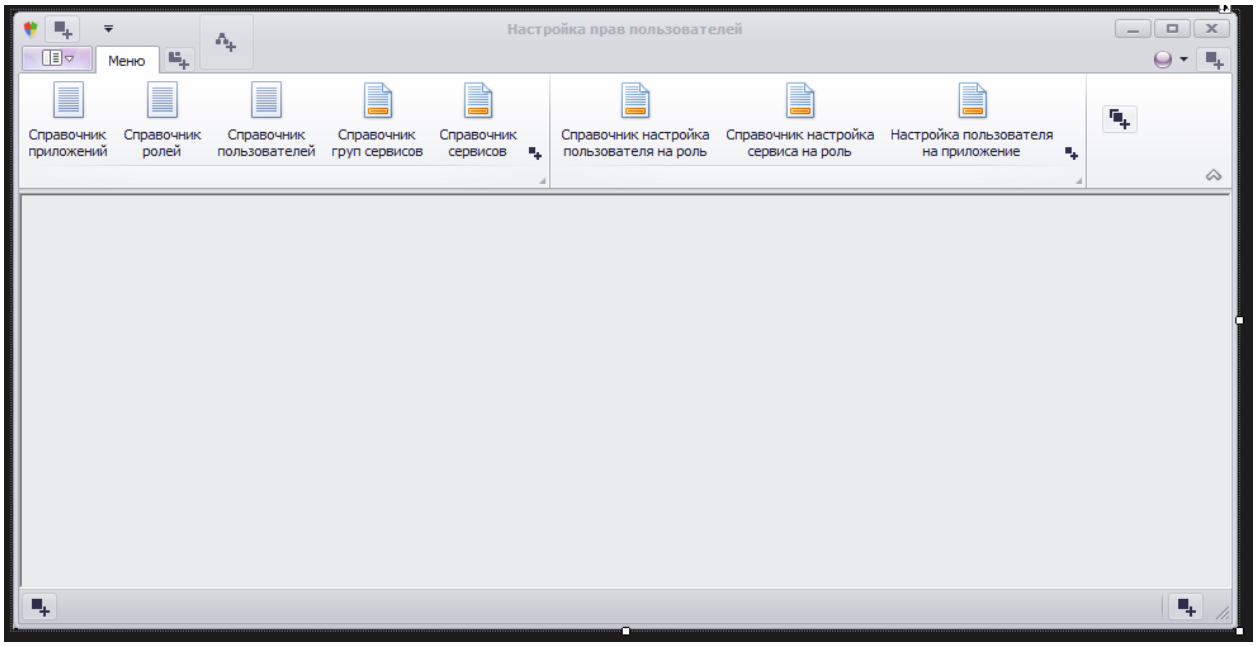


Рис 8

Для перегляду списку додатків потрібно натиснути кнопку список додатків.

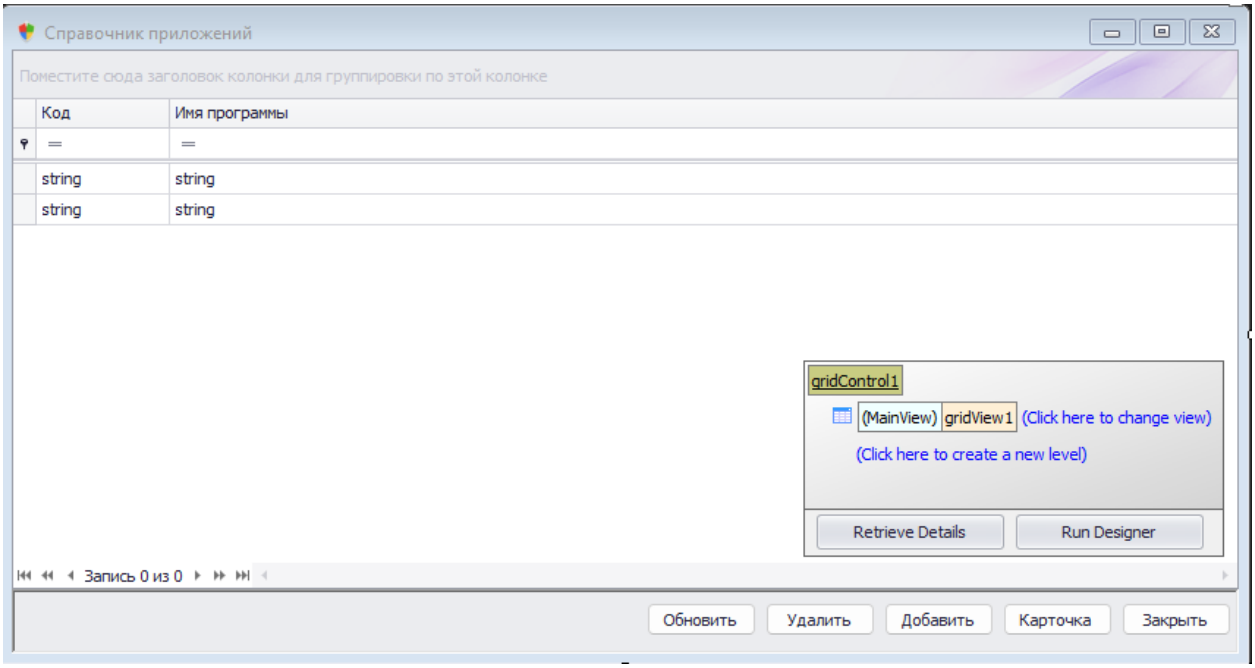


Рис 9

Якщо користувач має права на додавання, то натиснувши кнопку Додати, відкриється форма з текстовим полем, куди вам потрібно вписати ім’я нового додатку.

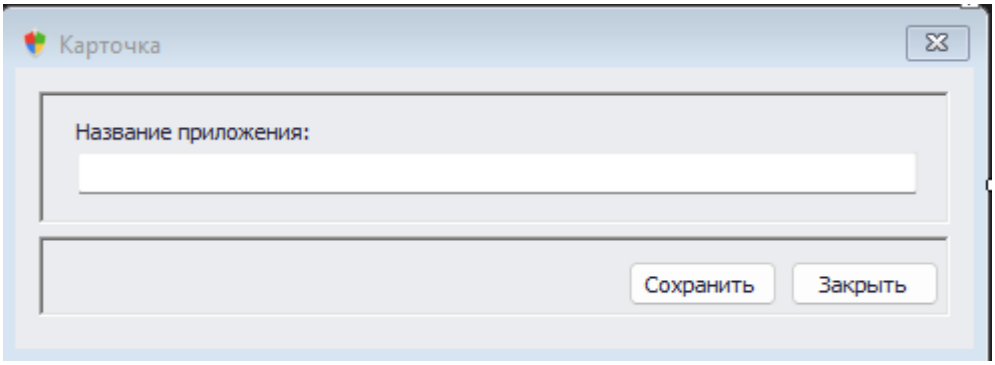


Рис 10

Якщо користувач хоче переглянути список ролей, то потрібно натиснути кнопку список ролей.

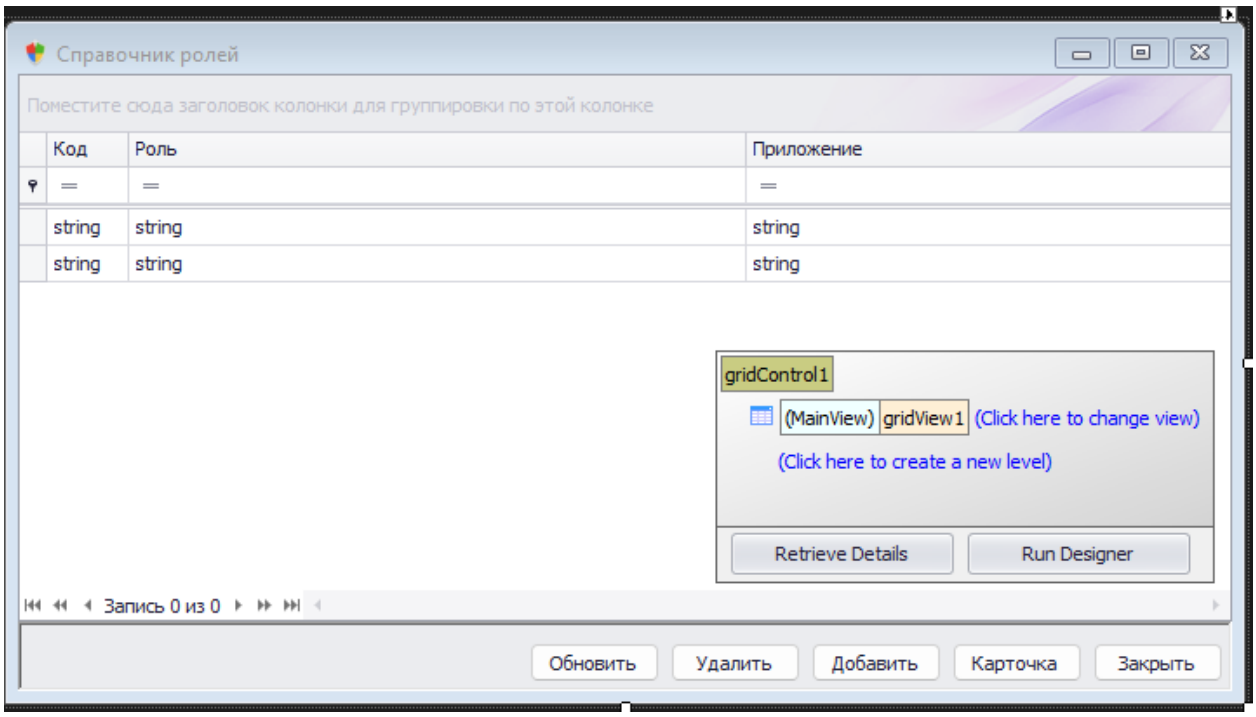


Рис 11

Для додавання нової ролі потрібно натиснути кнопку Добавить, відкриється форма з текстовим полем, куди потрібно вписати ім’я нової ролі та поле з випадаючим списком, де потрібно обрати додаток для цієї ролі.

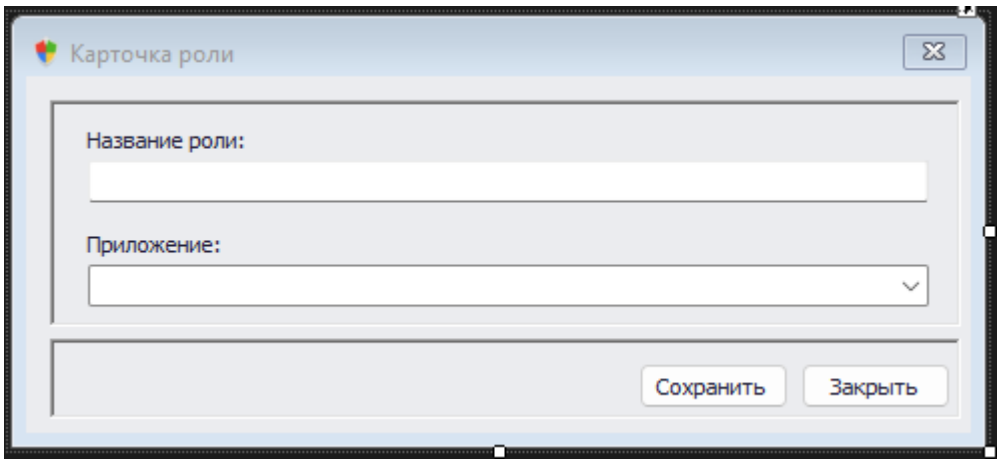


Рис 12

Для перегляду списку користувачів потрібно натиснути на кнопку Список пользователей.

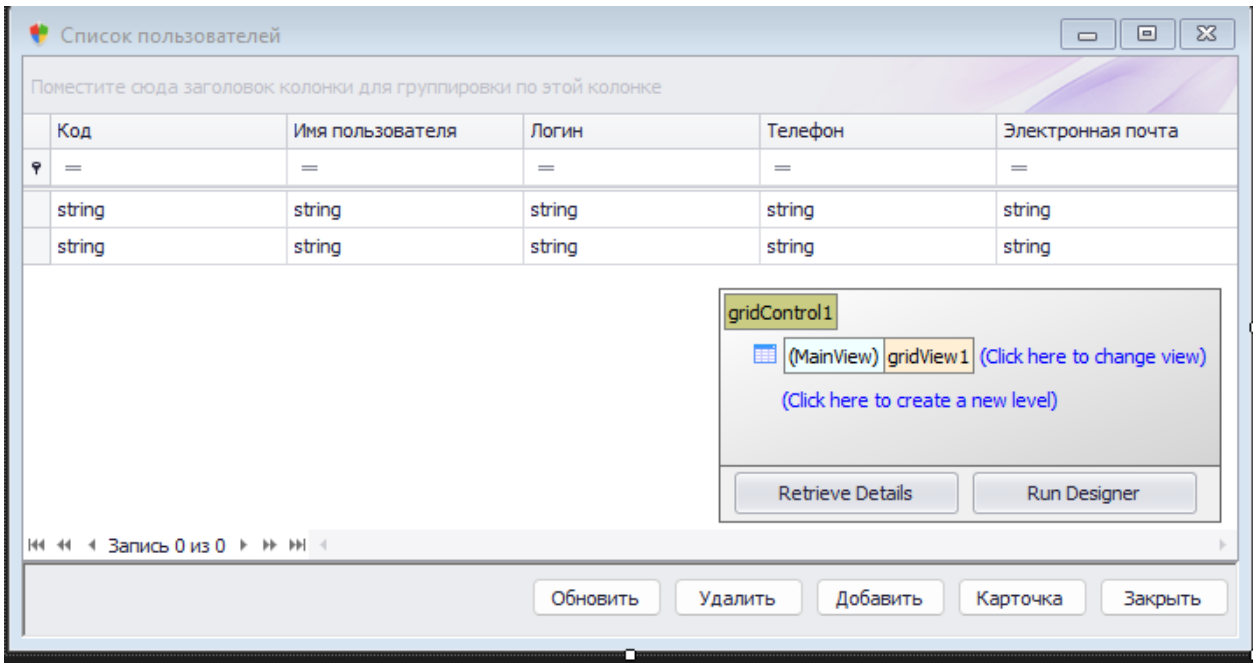


Рис 13

Для додавання нового користувача потрібно натиснути на кнопку Добавить, відкриється форма з чотирма текстовими полями, де потрібно вказати ім’я користувача, логін, телефон та електронну пошту.

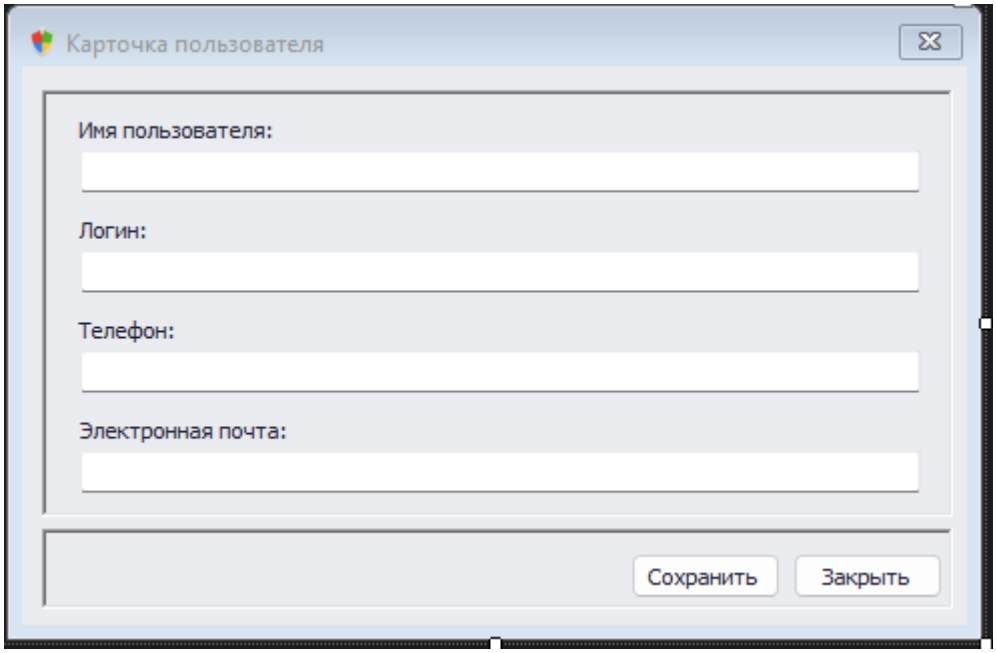


Рис 14

Для перегляду списку груп сервісів потрібно натиснути кнопку Список груп сервисов.

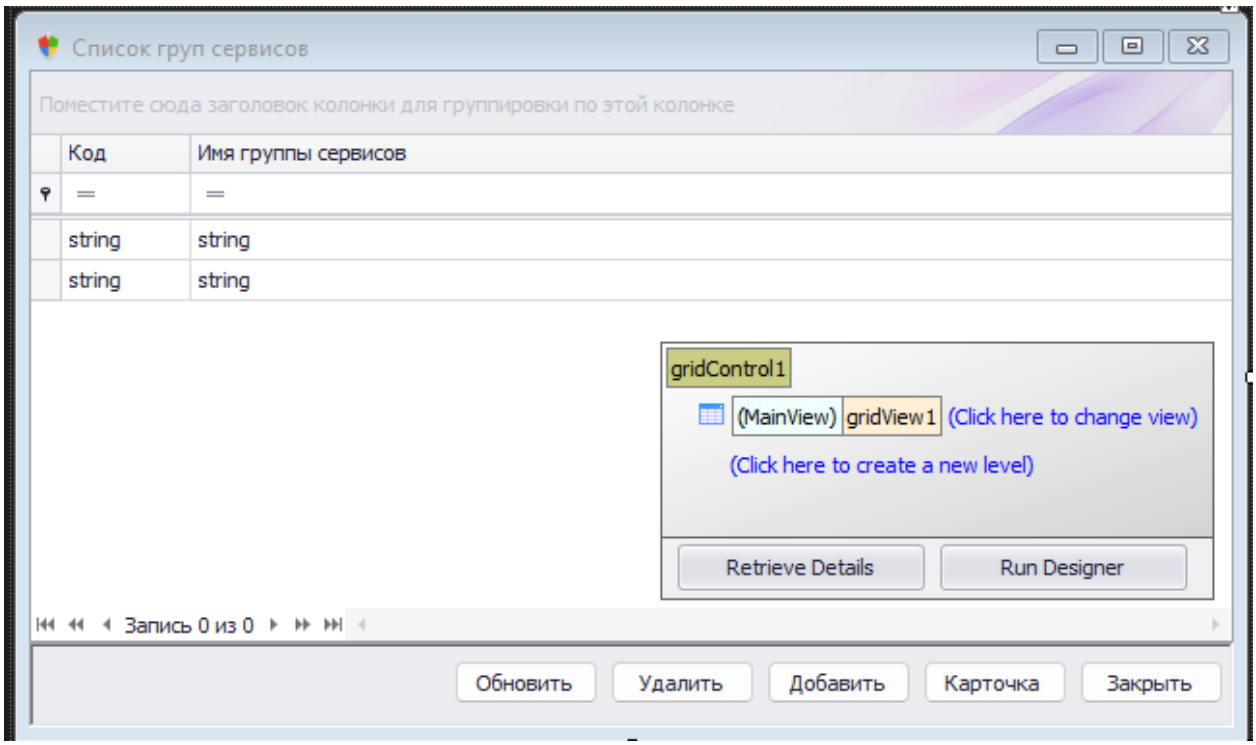


Рис 15

Для додавання нової групи сервісів потрібно натиснути кнопку додати, відкриється форма з текстовим полем, де потрібно вказати ім’я нової групи сервісів.

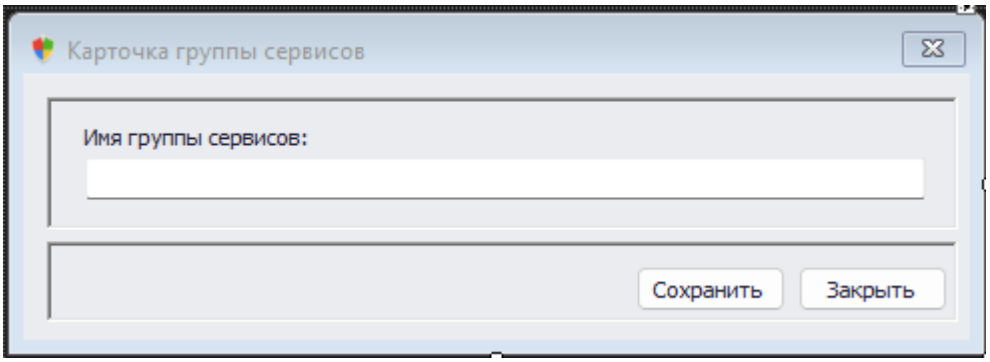


Рис 16

Для перегляду сервісів потрібно натиснути кнопку Список сервисов.

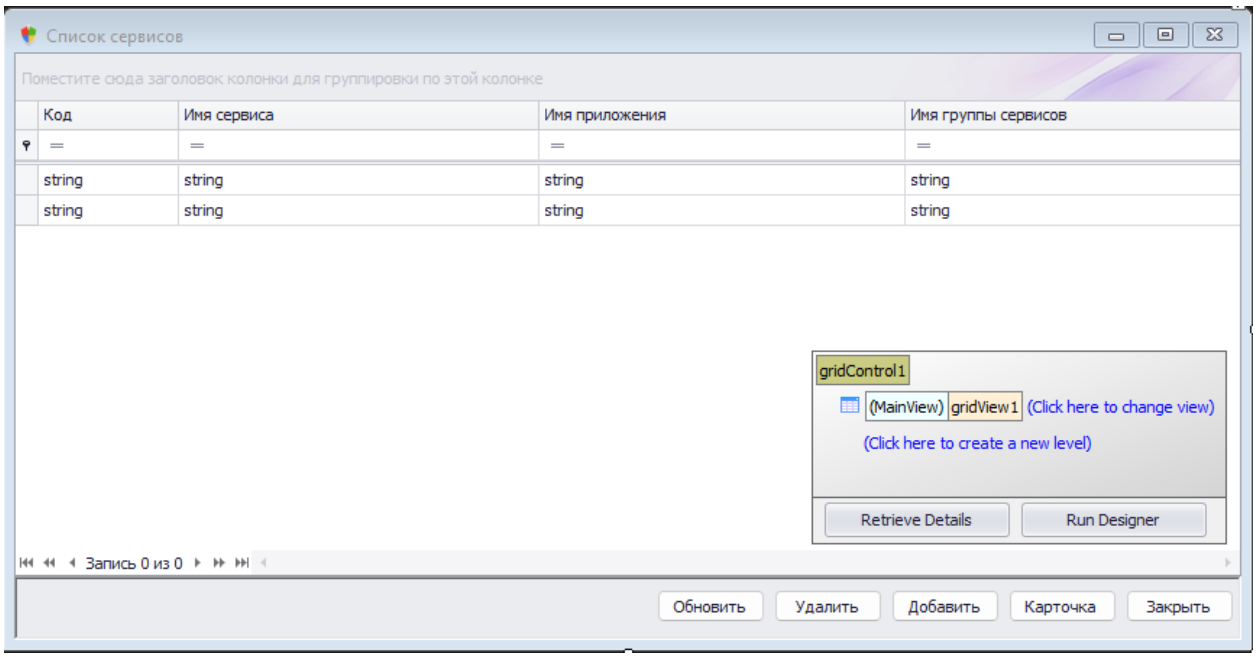


Рис 17

Для додавання нового сервісу потрібно натиснути кнопку Добавить, відкриється форма з текстовим полем, де потрібно вказати ім’я сервісу, де потрібно обрати додаток та групу сервісів.

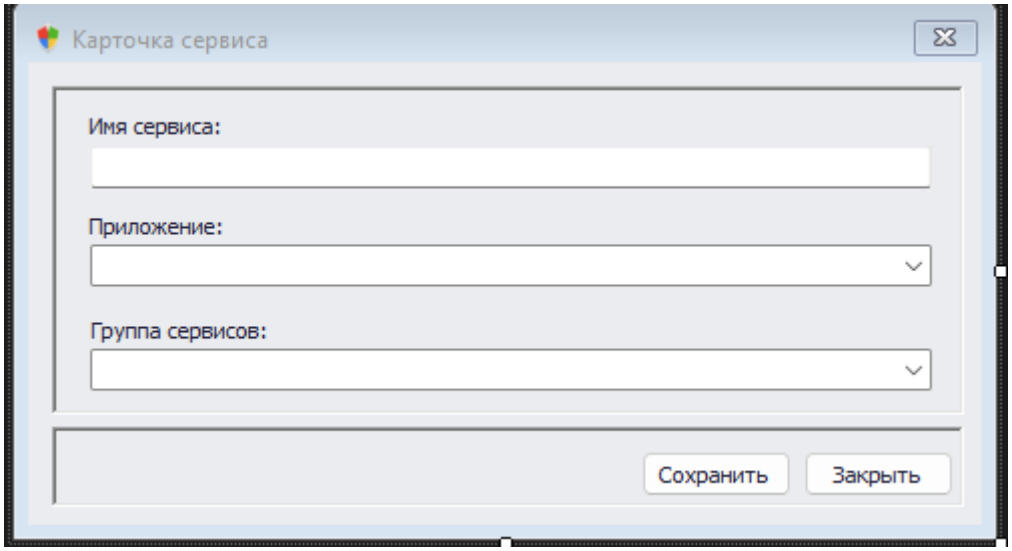


Рис 18

Для перегляду ролей користувача потрібно натиснути кнопку Настройка пользователя на роль.

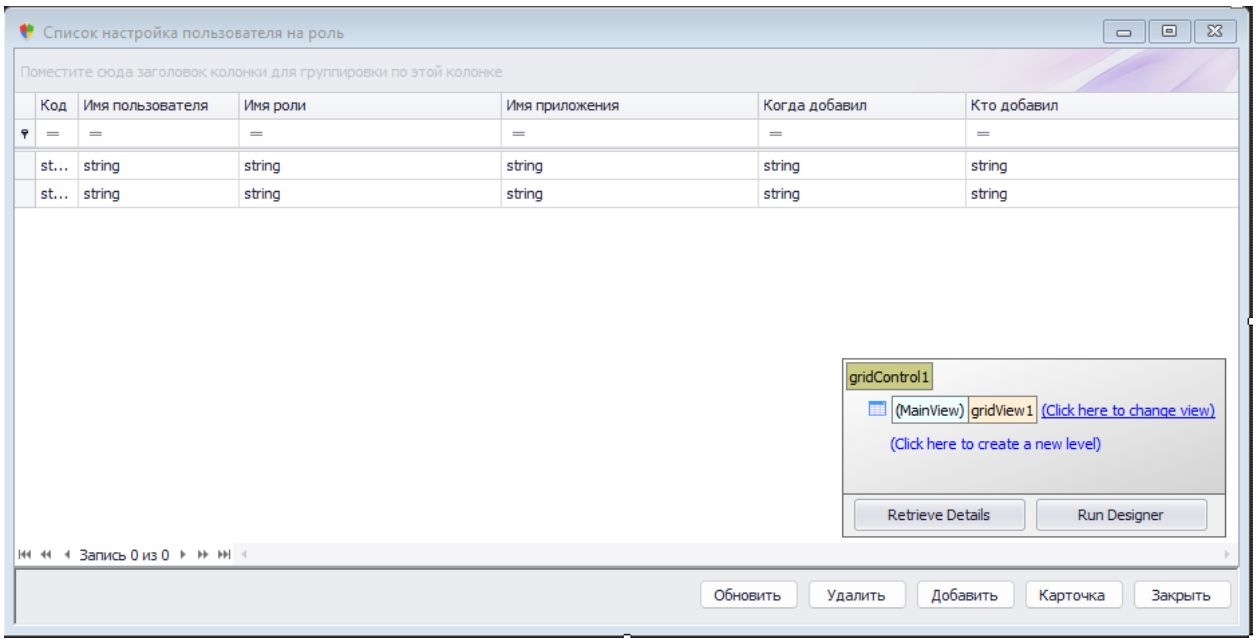


Рис 19

Для надання користувачу ролі потрібно натиснути кнопку добавить, відкриється форма з двома полями з випадаючим списком, де необхідно вказати користувача та роль, яку він матиме.

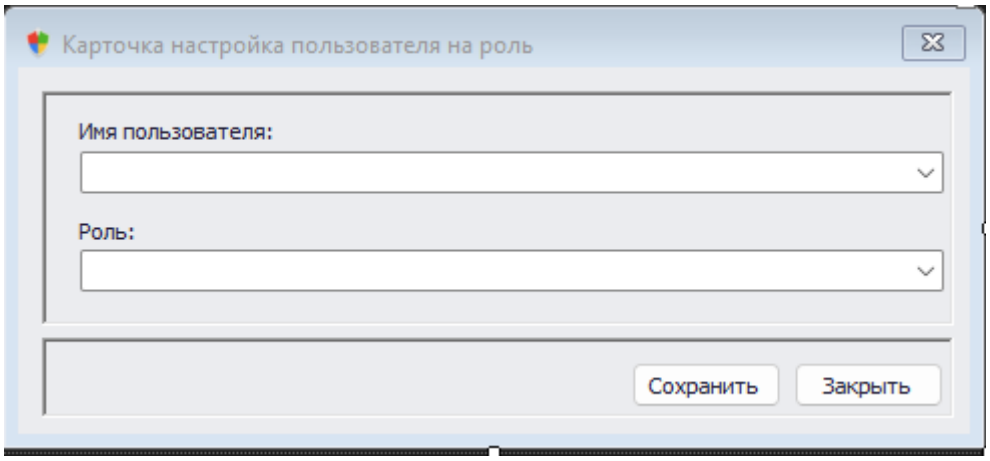


Рис 20

Для перегляду ролей та їх прав сервісу необхідно натиснути кнопку

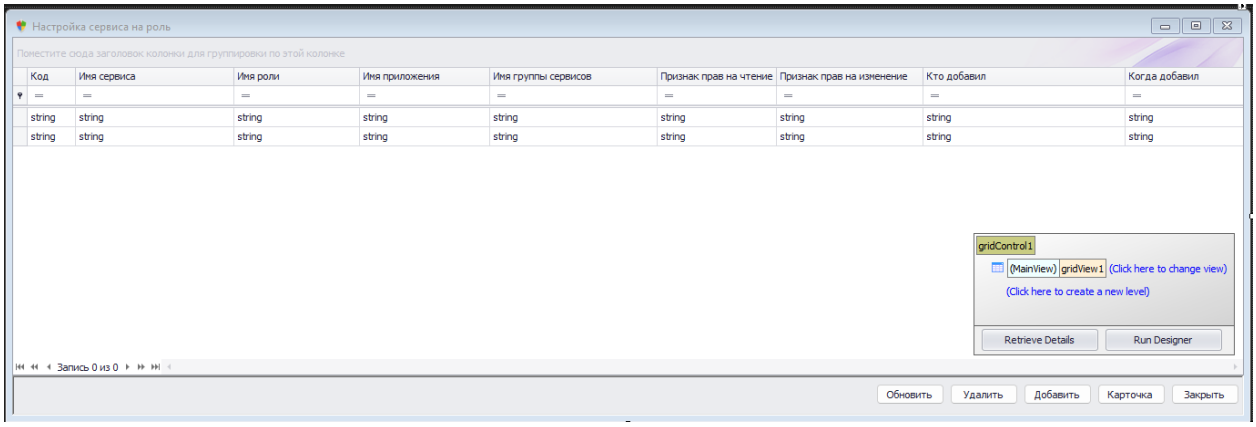


Рис 21

Для додавання для сервісу ролі з правами на перегдяд/модифікацію, потрібно натиснути кнопку Настройка сервиса на роль, відкриється форма, де потрібно вказати ім’я та роль сервісу та її права.

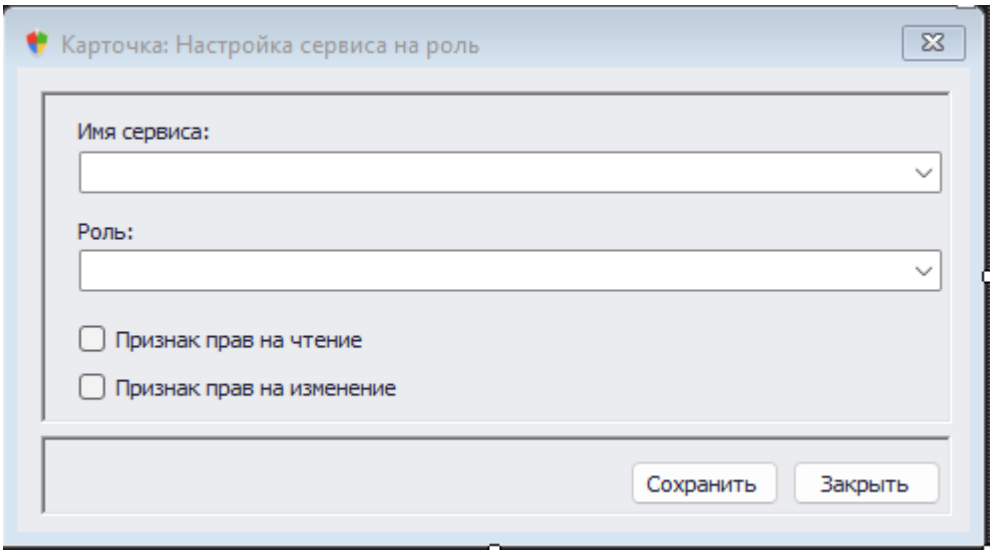


Рис 22

Для перегляду користувачів додатків потрібно натиснути кнопку Настройка пользователя на приложение.

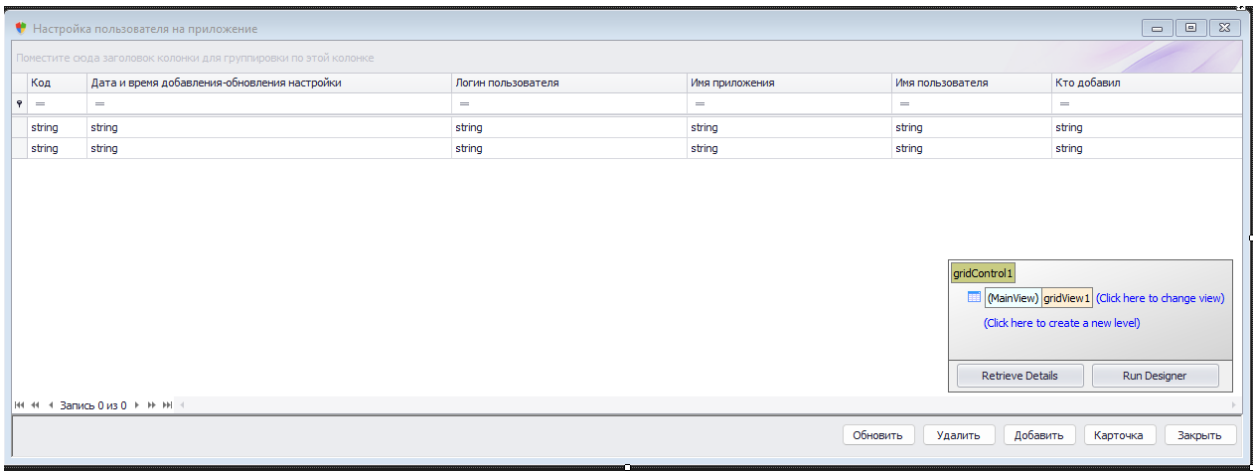


Рис 23

Для додавання нового користувача додатку потрібно натиснути кнопку Добавить, відкриється форма, де потрібно вказати користувача і додаток.

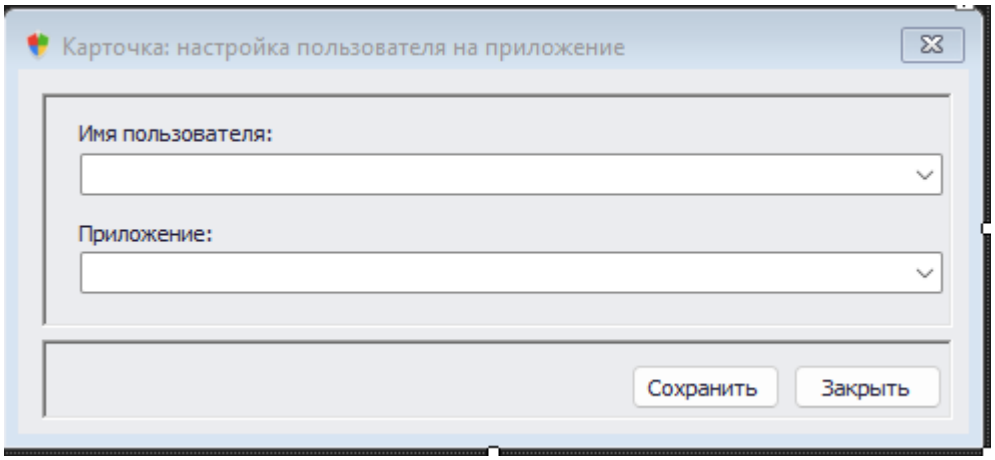
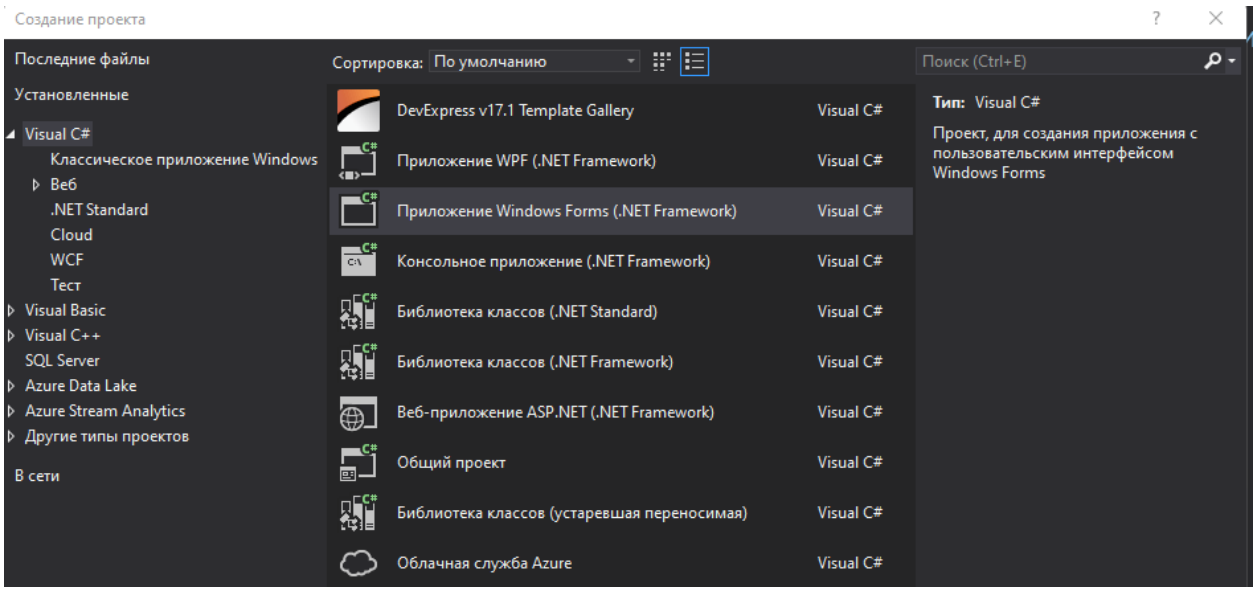


Рис 24

Після створення прототипу розпочинаємо розробку додатку з встановлення обраного середовища розробки Microsoft Visual Studio 2017 Community з офіційного сайту Microsoft.

Після завантаження, потрібно створити новий проект, у який й буде нашим додатком після завершення розробки.



Потрібно обрати Приложение Windows Forms (.NET Framework)

Також нам потрібно розробити базу даних нашого додатку.

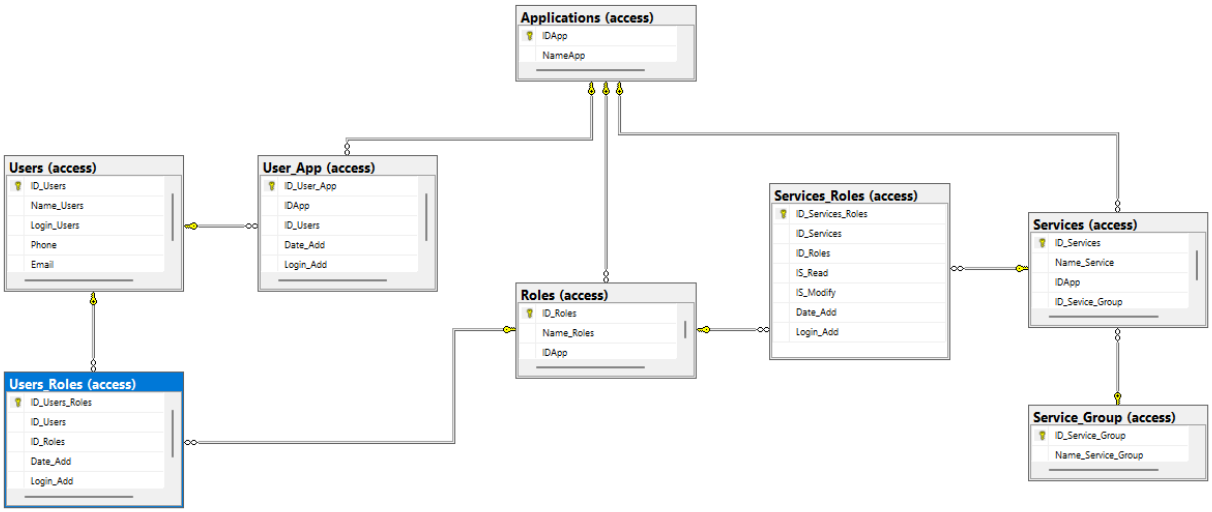


Рис 25

Далі було розроблено додаток, крок за кроком повторюючи прототип, прописуючи логіку.

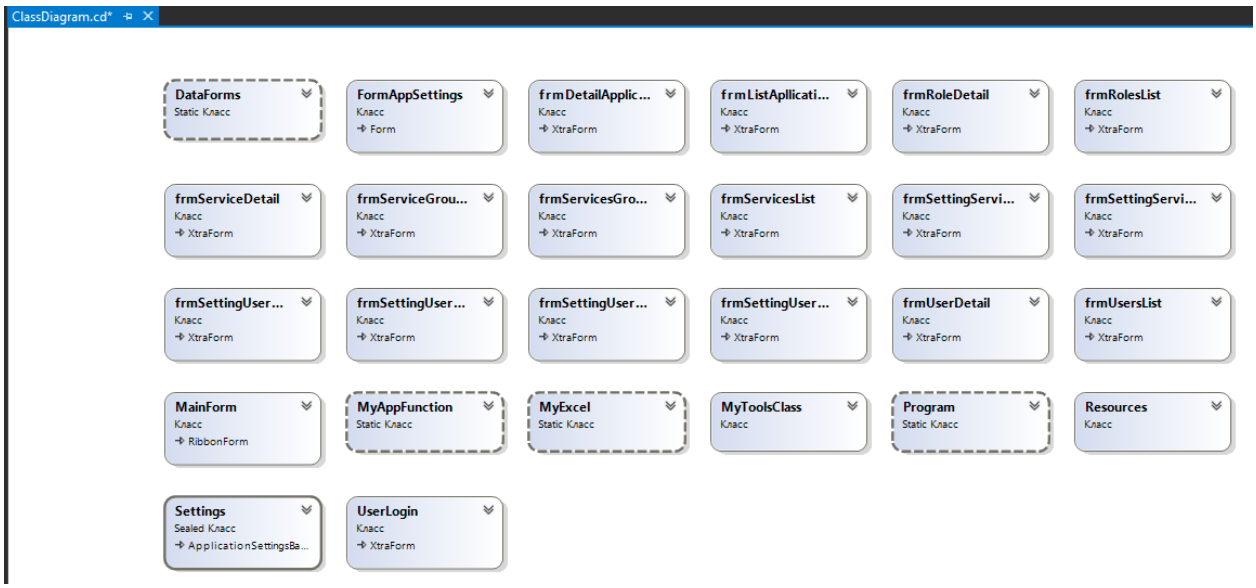

ClassDiagram.cd*
DataForms
Static Класс
FormAppSettings
Класс
Form
frmDetailApplic...
Класс
XtraForm
frmListApllicati...
Класс
XtraForm
frmRoleDetail
Класс
XtraForm
frmRolesList
Класс
XtraForm
frmServiceDetail
Класс
XtraForm
frmServiceGrou...
Класс
XtraForm
frmServicesGro...
Класс
XtraForm
frmServicesList
Класс
XtraForm
frmSettingServi...
Класс
XtraForm
frmSettingServi...
Класс
XtraForm
frmSettingUser...
Класс
XtraForm
frmSettingUser...
Класс
XtraForm
frmSettingUser...
Класс
XtraForm
frmSettingUser...
Класс
XtraForm
frmUserDetail
Класс
XtraForm
frmUsersList
Класс
XtraForm
MainForm
Класс
RibbonForm
MyAppFunction
Static Класс
MyExcel
Static Класс
MyToolsClass
Класс
Program
Static Класс
Resources
Класс
Settings
Sealed Класс
ApplicationSettingsBa...
UserLogin
Класс
XtraForm

Рис 26

ВИСНОВКИ

У результаті виконання даної дипломної роботи було розроблено додаток, за допомогою якого можна налаштовувати права користувачів на доступ до різних документів у корпоративному додатку. У роботі описується шлях створення даного додатку від ідеї, її аналізу, до її втілення у життя.

1. Під час роботи над дипломним проектом, було досліджено проблематику витоку інформації внаслідок надання доступу до конфіденційної інформації тим, у кого не має на це прав, знайдено шляхи їх вирішення, проаналізовано актуальність даної теми, та визначено її наукову новизну. Було прийнято рішення створити саме к додаток на ПК, через його розповсюдження в компаніях, найкращу відповідність вимогам мого додатку й найбільшим потенціалом успіху в майбутньому. Для покращення розуміння можливостей та функцій, які реалізуватиме мій додаток, був проведений аналіз аналогічного програмного забезпечення конкурентів.

2. Було проведено аналіз інструментів та програмних засобів реалізації, які найліпше підійдуть для створенням графічного інтерфейсу та усього функціоналу додатку. У ході аналізу було обрано використання C# як мову програмування, Microsoft Visual Studio 2017 як середовища розробки, T-SQL для розробки бази даних додатку, WinForms та бібліотеку DevExpress для створення елементів графічного дизайну, MS SQL Server Management Studio 2017 як середовища розробки бази даних додатку.

3. Визначення концепції, вимог та головних елементів інтерфейсу додатку. Далі був розроблений детальний прототип, який дозволяє переглянути увесь функціонал та графічний інтерфейс. Було створено діаграму бази даних мобільного додатку та діаграму класів додатку.

4. Розробка графічного інтерфейсу за допомогою WinForms і бібліотеки DevExpress, зв'язок з SQL Server, написання логіки додатку за

допомогою мови програмування C# у середовищі розробки Microsoft Visual Studio 2017.

Розроблений додаток має поліпшити механізм керування процесу надання прав користувачам доступу до тих чи інших документів.

Створений мобільний додаток має значні перспективи подальших досліджень. За допомогою WinForms та DevExpress можна постійно оновлювати функціонал додатку, що значно збільшить якість даного додатку, і як наслідок, полегшення процесу надання прав користувачам та поліпшення захисту інформації від витоку даних.

ПЕРЕЛІК ПОСИЛАНЬ

1. SQL Полное руководство Третье издание Джеймс Грофф Пол Вайнберг Эндрю Оппель – с. 437 – 442.

2. Душан Петкович Microsofot SQL Server 2012 Руководство для начинающих – с. 338 – 339

3. Microsoft® SQL Server® 2012. Создание запросов. Учебный курс Microsoft: Пер. с англ. / И. Бен-Ган, Д. Сарка, Р. Талмейдж. — М.: Издательство «Русская редакция», 2014. – с. 9 – 13.

4. https://ru.wikipedia.org/wiki/Transact- SQL

5. https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%A3%D1%82%D0%B5%D1%87%D0%BA%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85#:~:text=%D0%9F%D0%BE%20%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%BC%20%D1%8D%D0%BA%D1%81%D0%BF%D0%B5%D1%80%D1%82%D0%BE%D0%B2%2C%20%D0%B7%D0%B0%202019,%D0%B2%20%D0%BC%D0%B8%D1%80%D0%B5%20%D0%B2%202019%20%D0%B3%D0%BE%D0%B4%D1%83.

6. https://metanit.com/sharp/adonet/1.1.php.

7. https://docs.microsoft.com/ru-ru/dotnet/desktop/winforms/overview/?view=netdesktop-6.0.

8. https://metanit.com/sql/sqlserver/1.1.php

ДОДАТКИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО –НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра Інженерії програмного забезпечення

Розробка сервісу налаштувань прав користувачів у корпоративних додатках мовою C#

Виконавець: студент 4 курсу,
групи ПД–42
Кириллов Микола Вячеславович
Керівник роботи: Трінтіна Наталія Альбертівна, доцент кафедри ІПЗ

Київ 2022

Мета, об’єкт та предмет роботи

Мета роботи – покращення процесу надання/обмеження прав користувачів у корпоративних IT-програмах.

Об'єкт дослідження – процес надання/обмеження прав користувачів у корпоративних IT-програмах.

Предмет дослідження – додаток, де надаються/обмежуються права користувачів у корпоративних IT-програмах.

Аналіз аналогів

Переваги

- Ми налаштовуємо 1 сервіс користувача і забуваємо про проблему. У той же час цей сервіс може використовувати десятки процедур, таблиць та хлопців на різних серверах баз даних. Стандартна модель безпеки SQL вимагає налаштувати на роль або нового користувача кожен об'єкт.
- Зручний інтерфейс

Недоліки

- Користувач, який налаштовуватиме права доступу, може бути не пов'язаний з IT. З одного боку, це плюс, тому що не відволікає IT на дану роботу. З іншого боку, користувач може не правильно дати права, що вимагатиме уваги IT фахівця для виправлення ситуації.
- Ми даємо користувачеві всі права на базі, а обмеження робимо на рівні програми. З точки зору SQL Server це не дуже добре. Але з погляду тимчасових витрат підтримки сотень користувачів на десятках серверах з величезною кількістю баз даних, це рішення мінімізує налаштування прав доступу і дозволяє передати налаштування прав від IT фахівців до, наприклад, служби підтримки або досвідчених користувачів. Такі рішення застосовують у великих корпораціях, наприклад Рено, Ніссан і т.д.

Технічне завдання

- База даних компанії.
- Інтерфейс додатку.
- Розробка ПО додатку.

Спеціалізовані технології

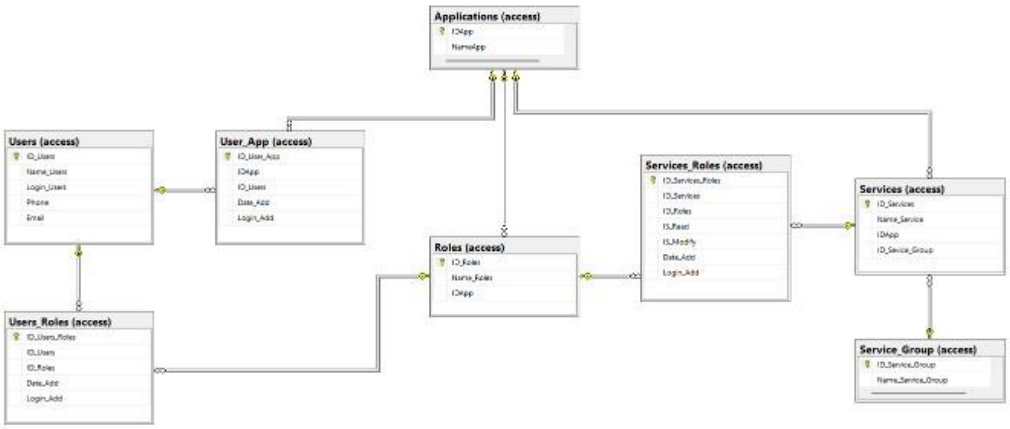
Платформа розробки додатку – Visual Studio.

Мови програмування– C#, T-SQL.

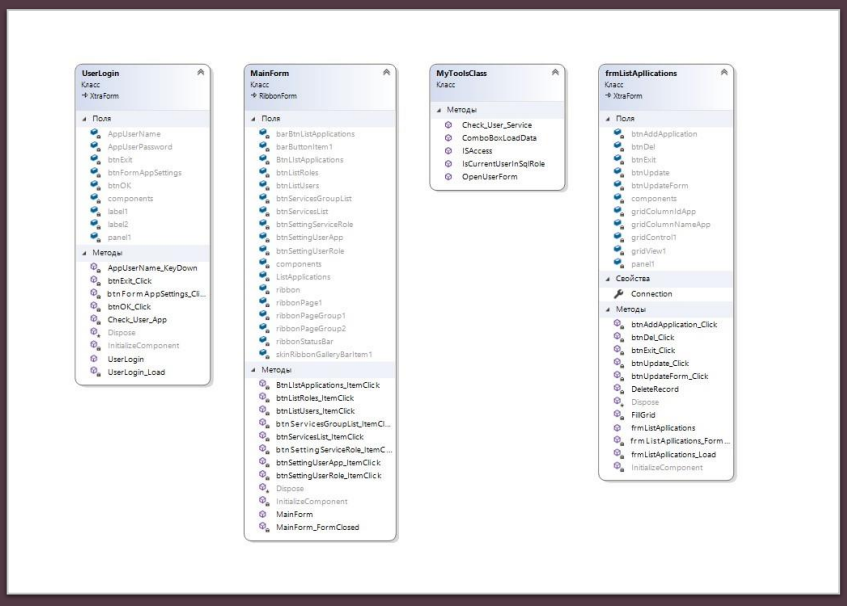
Технології– WinForms, ADO.NET

Бібіліотека – DevExpress

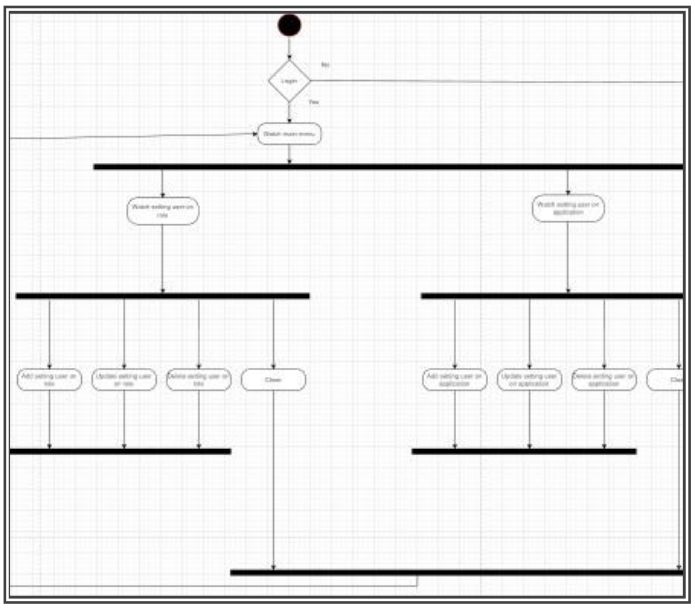
МЕТОДИ ТА КЛАСИ ПРОГРАМИ



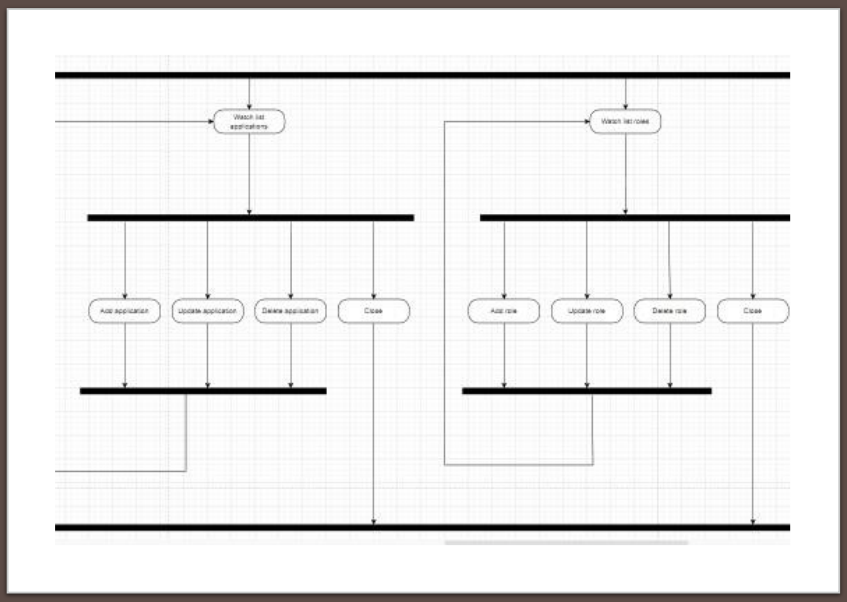
Class Diagram



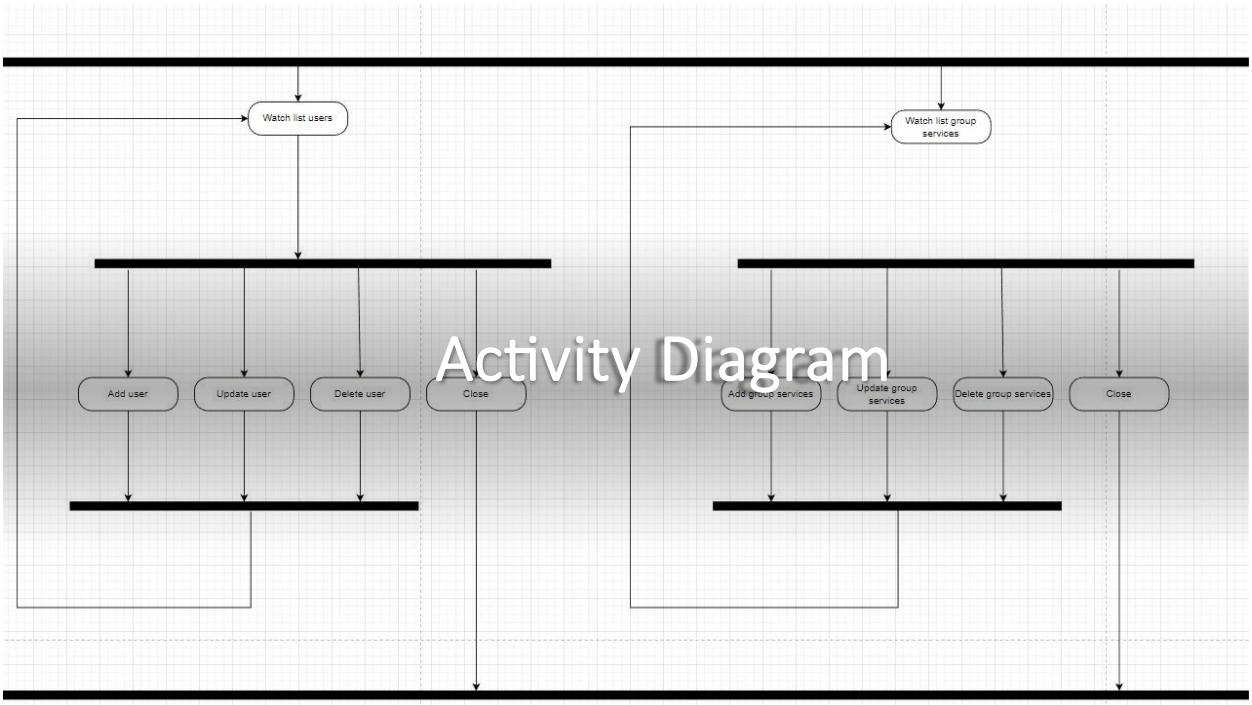
Class Diagram

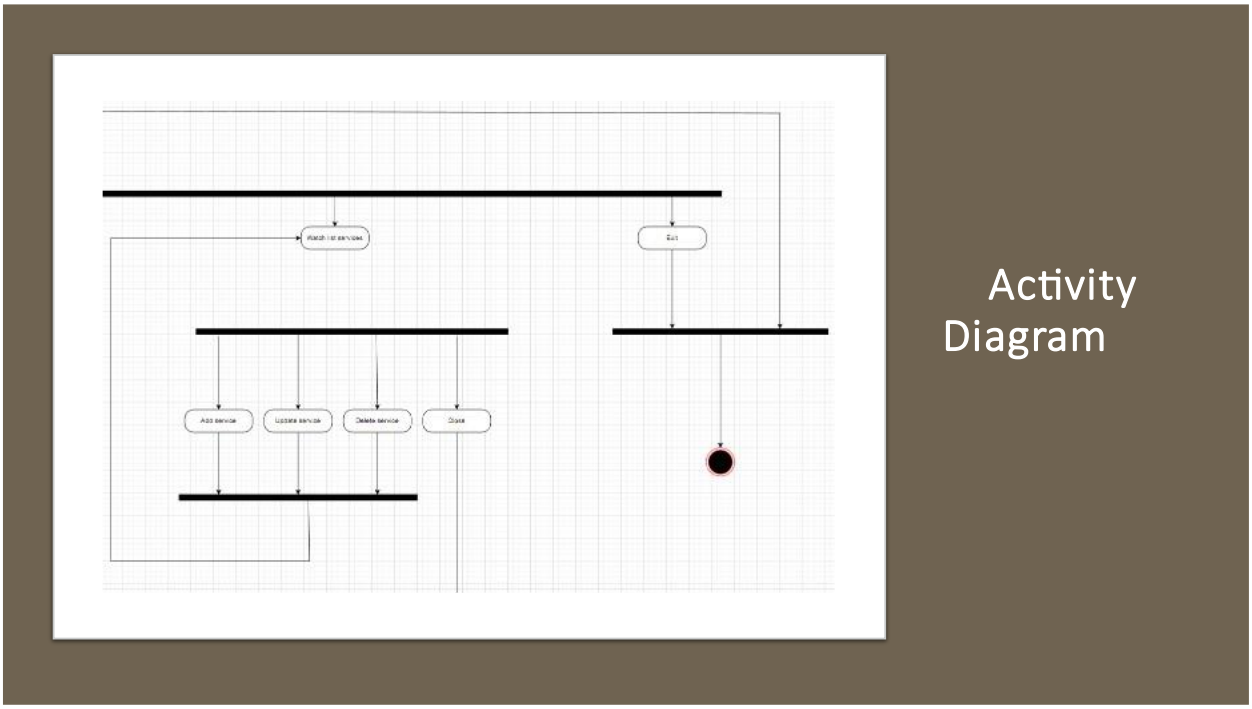


- **Activity Diagram**

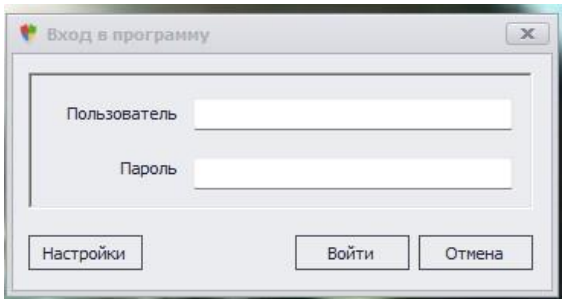


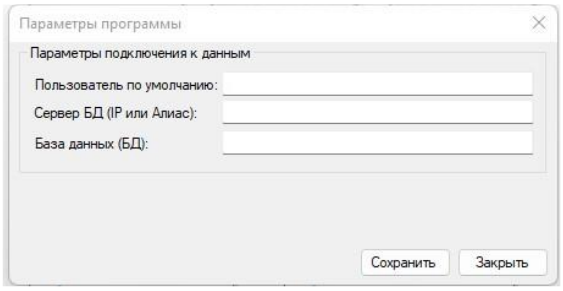
Activity Diagram

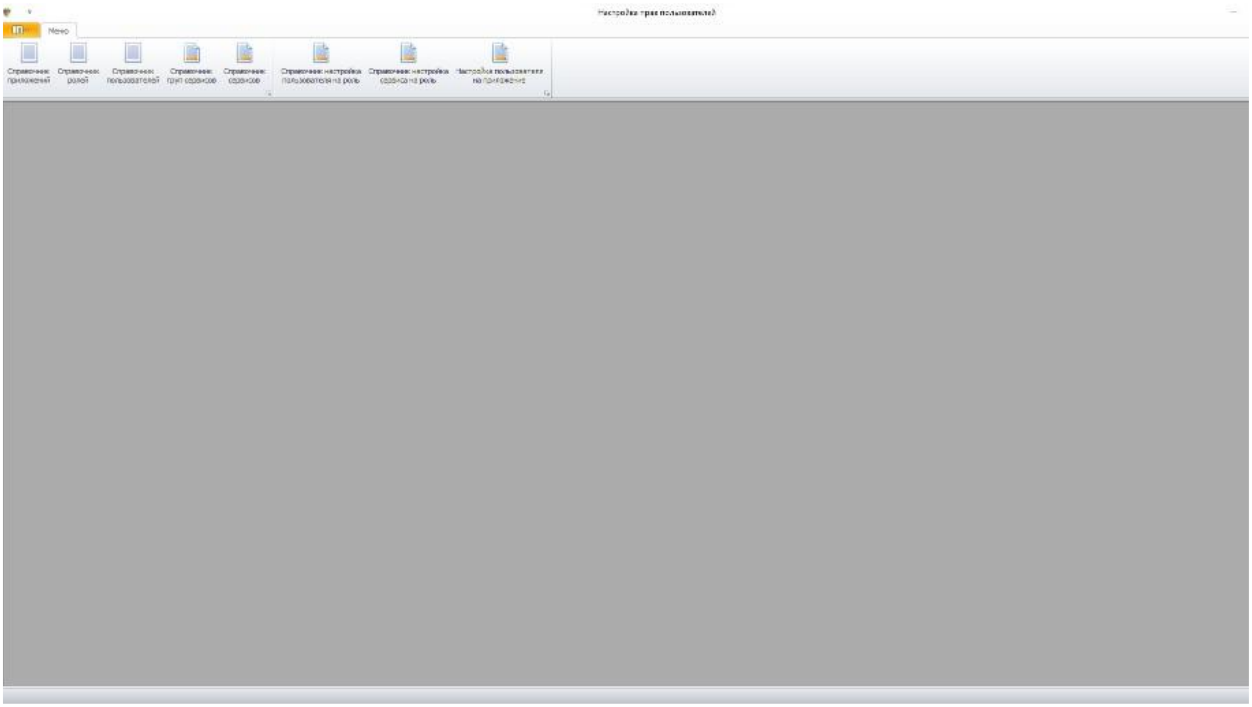

Watch list users
Add user
Update user
Delete user
Close
Watch list group services
Add group services
Update group services
Delete group services
Close
Activity Diagram


Activity Diagram

Екранні Форми







Наукова новизна та практична значимість

Наукова новизна дослідження полягає в тому, що раніше компанія не застосовувала даний механізм, будь-яка компанія може впровадити даний додаток.

Практична значимість дослідження полягає в створенні продукту, який дозволить ефективно надавати користувачам відповідні ролі та права для доступу до відповідної інформації.

Висновки

Створено бюджетний та ефективний додаток для керування ролями і правами користувачів, що підтверджую досягнення мети дипломного проекту.

Проект AccessUser має наступні переваги:

1. Зручний інтерфейс.
2. Ми налаштовуємо 1 сервіс користувача і забуваємо про проблему. У той же часцей сервіс може використовувати десятки процедур, таблиць та хлопців на різних серверах баз даних. Стандартна модель безпеки SQL вимагає налаштувати на роль або нового користувача кожен об'єкт.
3. Бюджет проекту – безкоштовний, оскільки проект виконаний внутрішніми силами.

Даний проект пройшов апробацію на НАУКОВО-ТЕХНІЧНІЙ КОНФЕРЕНЦІЇ « ЗАСТОСУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В ІНФОКОМУНІКАЦІЙНИХ ТЕХНОЛОГІЯХ».

ДЯКУЮ ЗА УВАГУ!