МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ

# КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

**на тему** «Програмний додаток підтримки діяльності компанії з ремонту комп'ютерної техніки»

за спеціальністю 122 «Комп'ютерні науки»

**Виконавець роботи:** студент групи ІТ-71 Проценко Максим Олегович

**Кваліфікаційна робота бакалавра**
**захищена на засіданні ЕК**
**з оцінкою** **_____ «__»____2021р.**

Науковий керівник: _____ доц. Ващенко С.М.
(підпис)

Голова комісії _____ Шифрін Д.М._____
(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань

Студент _____
(підпис)

**Суми-2021**

**Сумський державний університет**
**Факультет** електроніки та інформаційних технологій
**Кафедра** комп'ютерних наук
**Секція** інформаційних технологій проектування
**Спеціальність** – 122 «Комп'ютерні науки та інформаційні технології»

**ЗАТВЕРДЖУЮ**

Зав. секції ІТП

_____В.В. Шендрик

«___»_____2021 р.

**ЗАВДАННЯ**

**НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ**

*Проценко Максим Олегович*

**1 Тема роботи** *Програмний додаток підтримки діяльності компанії з ремонту комп'ютерної техніки*

**керівник роботи** *доц. Ващенко С.М.*,

затверджені наказом по університету від *« 14» квітня 2021 р. № 0181-VI*

**2 Строк подання студентом роботи** *« 07 » червня 2021р.*

**3 Вхідні дані до роботи** *технічне завдання на розробку програмного додатку підтримки діяльності компанії з ремонту комп'ютерної техніки*

**4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)** *аналіз предметної області, проектування програмного додатку, розробка програмного додатку*

**5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)** *Актуальність роботи. Постановка задачі. Аналіз предметної області. Структурно-функціональна модель.Діаграма варіантів використання. Схема бази даних. Демонстрація роботи. Висновки*

**6. Консультанти розділів роботи:**

| **Розділ** | **Консультант** | **Підпис, дата** | |
|---|---|---|---|
| | | **Завдання видав** | **Завдання прийняв** |
| | | | |

**7.Дата видачі завдання**_____

## КАЛЕНДАРНИЙ ПЛАН

| № п/п | Назва етапів кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Визначення бажаних напрямків роботи | 18.01.21-29.01.21 | |
| 2 | Обговорення роботи з керівником | 01.02.21-12.02.21 | |
| 3 | Затвердження теми | 13.02.21-18.02.21 | |
| 4 | Обговорення часу, бюджету і технологій, які будуть використовуватись із замовником | 19.02.21-04.03.21 | |
| 5 | Проектування додатку | 05.03.21-22.05.21 | |
| 6 | Опис результатів виконання | 23.05.21-29.05.21 | |
| 7 | Оформлення документації | 30.05.21-03.06.21 | |
| 8 | Представлення проекту замовнику | 03.06.21-15.06.21 | |

**Студент** _____ Проценко М.О.

**Керівник роботи** _____ доц. Ващенко С.М.

# РЕФЕРАТ

Тема роботи «Програмний додаток  підтримки діяльності компанії  з ремонту комп'ютерної техніки»

Пояснювальна записка складається зі вступу, трьох основних розділів, висновку, списку використаних джерел із 15 найменувань та трьох додатків. Загальний обсяг пояснювальної записки складає 101 сторінку, в тому числі 57 сторінок основного тексту, 2 сторінки списку використаних джерел, 44 сторінок додатків.

Перший розділ містить визачення актуальності проблеми, огляд всіх продуктів-аналогів, а також – формування завдання на проектування програмного додатку.

Другой розділ складається з моделювання додатку. В ньому було проведено структурно-функціональне моделювання, створено всі необхідні UML-моделі та проведено моделювання бази даних.

Третій розділ описує архітектуру додатку, процес створення бази даних, вікон додатку, логіки програми, а також – описує процес роботи користувача в програмі.

В результаті виконання даної роботи було створено додаток для сервісу з ремонту ПК, який буде допомагати робітникам вдало структурувати та зберігати всю необхідну інформацію щодо замовлень.

Ключові слова: ДЕСКТОПНИЙ ДОДАТОК, WINDOWS PRESENTATION FOUNDATION, MYSQL, ВІДДАЛЕНИЙ СЕРВЕР, PDF, ITEXTSHARP, MATERIALDESIGN, C#.

**ЗМІСТ**

# ВСТУП

В будь-якому виду бізнесу найважливіше за все – контроль. Якщо говорити конкретно про сферу ремонту комп'ютерної техніки, дуже важливо не загубитись серед незліченої кількість замовлень, деякі з яких можуть виконуватись по кілька місяців. В такому випадку користуватись паперовим записником не дуже зручно.

В наш час інформаційні системи дуже активно використовуються, щоб забезпечити зручний збір інформації, а також можна було з мінімальним затратами часу упорядковувати та управляти нею. Подібні системи потрібні майже в кожному, навіть найменшому магазині, сервісі, навчальному закладу тощо. Саме тому питання розробки подібної системи для сервісного центру по ремонту комп'ютерної техніки є актуальною [6].

Об'єктом даної роботи є процес систематизації інформації щодо замовлень в сервісі з ремонту комп'ютерної техніки. Сам по собі процес доволі простий – необхідно лише зберегти необхідні дані щодо замовлень, але, при цьому, потрібно зробити це в максимально зручному форматі для того, щоб користувач в будь-який момент мав доступ до інформації.

Предметом роботи є один із можливих варіантів вирішення даної проблеми – програми з графічним інтерфейсом та елементами, які можуть допомогти користувачу виконувати свою роботу, не відволікаючись на систематизацію та керування інформацією.[7]

Мета кваліфікаційної роботи – створити програму з графічним інтерфейсом, яка буде виконувати всі необхідні операції з інформацією про замовлення, що виконуються в майстерні, та матиме можливість синхронізувати дані між декількома комп'ютерами.

Основні задачі роботи:

- проаналізувати існуючі аналоги та сформувати вимоги до програмного продукту;

- провести модулювання процесу організації роботи з інформацією в сервісному центрі та роботи з майбутнім програмним забезпеченням;
- змоделювати та реалізувати базу даних для збереження інформації;
- виконати програмну реалізацію та оформити потрібну документацію.

За результатами проведеної роботи було опубліковано тези доповіді на конференції «ІМА-2021» [5].

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Огляд останніх досліджень і публікацій

В будь-якому сервісному центрі, найскладніше завдання – розібратися із інформацією щодо замовлень. Наприклад, в магазині замовника даного проекту «Мелт», вся необхідна інформація щодо замовлень записується на стікерах під час прийому замовлення. На них вдається помістити лише номер телефону клієнта і тип неполадки. Решта персональних даних щодо замовника (таких як ім’я та фамілія) просто ігноруються. Таким чином, зовсім не ведеться архів замовлень.

Проблеми виникають, коли потрібно розпочинати ремонт або видавати замовлення клієнту. Дуже часто частина інформації просто розмазується по стікеру, і навіть номер телефону прочитати неможливо. Також, деякі з майстрів мають нерозбірливий почерк, що дуже заважає і сповільнює виконання обов’язків. Також, після виконання замовлення потрібно подзвонити клієнту і проінформувати про результати ремонту. Інколи, клієнти залишають не свій особистий номер, і виникають недоречні ситуації, коли майстер не знає, хто саме йому потрібен.

Майже всі великі сервісні центри мають своє програмне забезпечення для керування замовленнями. У випадку СЦ «Мелт», замовлень не так багато, але при цьому вони стабільно існують і потребують систематизації. То ж, в такому випадку набагато дешевше буде створити свій, доволі простий програмний продукт для керування замовленнями, зі всіма необхідними функціями, при цьому не платити кожного місяця доволі великі кошти за підписку на різноманітні сервіси.

## 1.2 Аналіз програмних продуктів, які підходять для запису та систематизації інформації про замовлення

Було знайдено доволі багато програм-аналогів, які підходять для систематизації та збереження інформації щодо замовлень. Серед них:

- Microsoft Excel,
- Google Tables,
- Microsoft ToDo,
- Рем онлайн.

Перший із знайдених варіантів - Microsoft Excel (рис. 1.1). Програма має зручний формат таблиці, що підходить для систематизації, а за допомогою вбудованих функцій можна автоматизувати деякі процеси, що стосуються виконання замовлень. До мінусів MS Excel можна віднести те, що для створення необхідного функціоналу потрібно буде довго ознайомлюватись з можливостями цього редактору, при цьому не вдасться досягти ідеальної форми запису, яка підходить для нашої мети. Також, ця програма досить дорога і через неї буде дуже важко організувати одночасний доступ до даних із декількох пристроїв. Приклад робочого вікна MS Excel зображено на рисунку 1.1:[8]

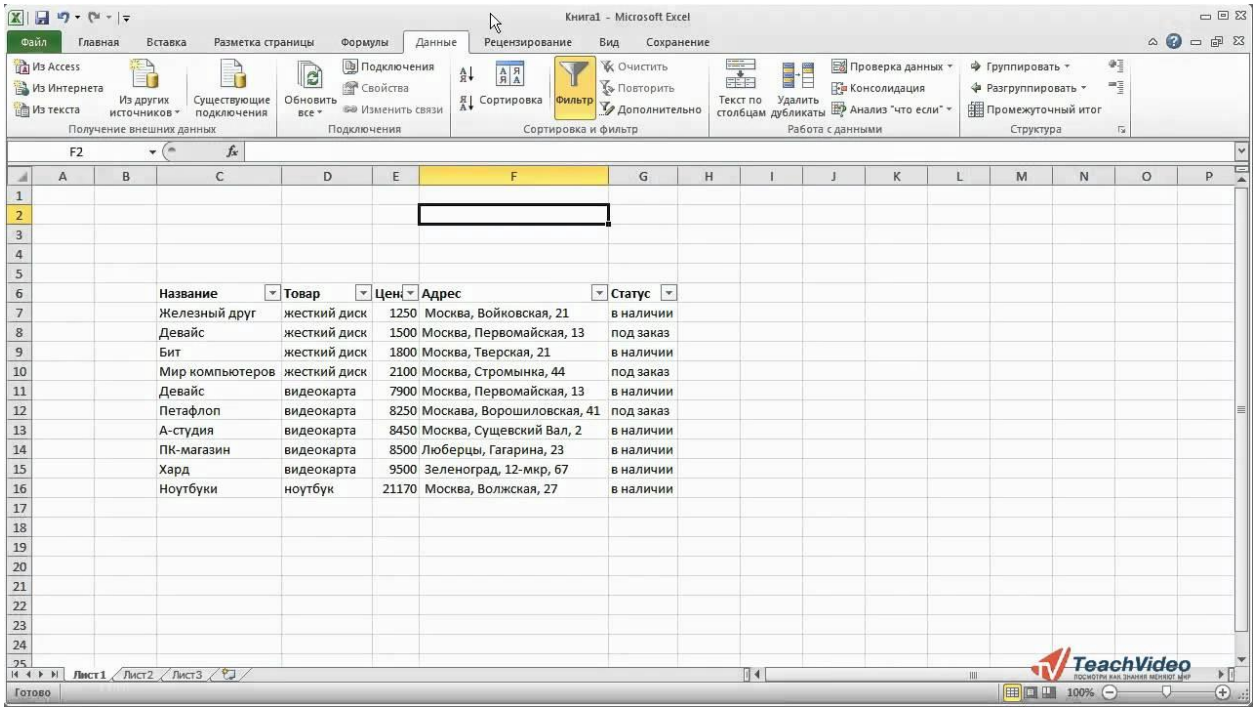



Рисунок 1.1 – Приклад використання MS Excel

Гугл Таблиці – це онлайн-варіант Excel. За рахунок можливості одночасного доступу великих груп осіб до цієї таблиці через мережу Інтернет, Google Tables являється набагато гнучкішим інструментом. У всьому іншому – сервіс має такі ж проблеми, як і MS Office. Для освоєння формул і створення необхідного функціоналу в таблиці доведеться витратити дуже багато часу, при цьому добитись максимальної зручності відображення – не вдасться. Також, з’являється нова проблема – неможливо працювати без доступу до мережі. Приклад робочого вікна Google Tables зображено на рис. 1.2 [9]:

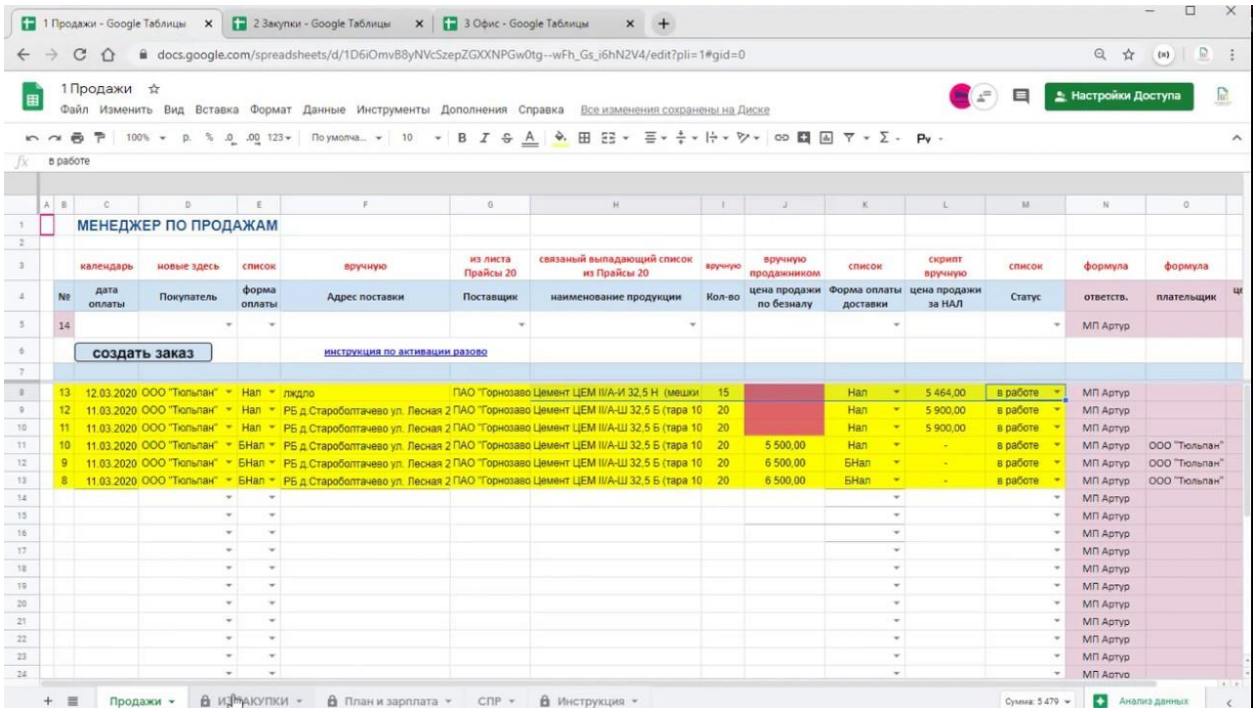



Рисунок 1.2 – Приклад використання Google tables

Microsoft ToDo – сайт і додаток для смартфонів, який дозволяє записувати свої завдання на день та отримувати нагадування щодо необхідності їх виконання. Для сервісу з ремонту ПК він може бути корисним лише в тому плані, що робітники перестануть забувати про необхідність виконання тих чи інших поручень. Приклад робочого вікна Microsoft ToDo зображено на рисунку 1.3[3].

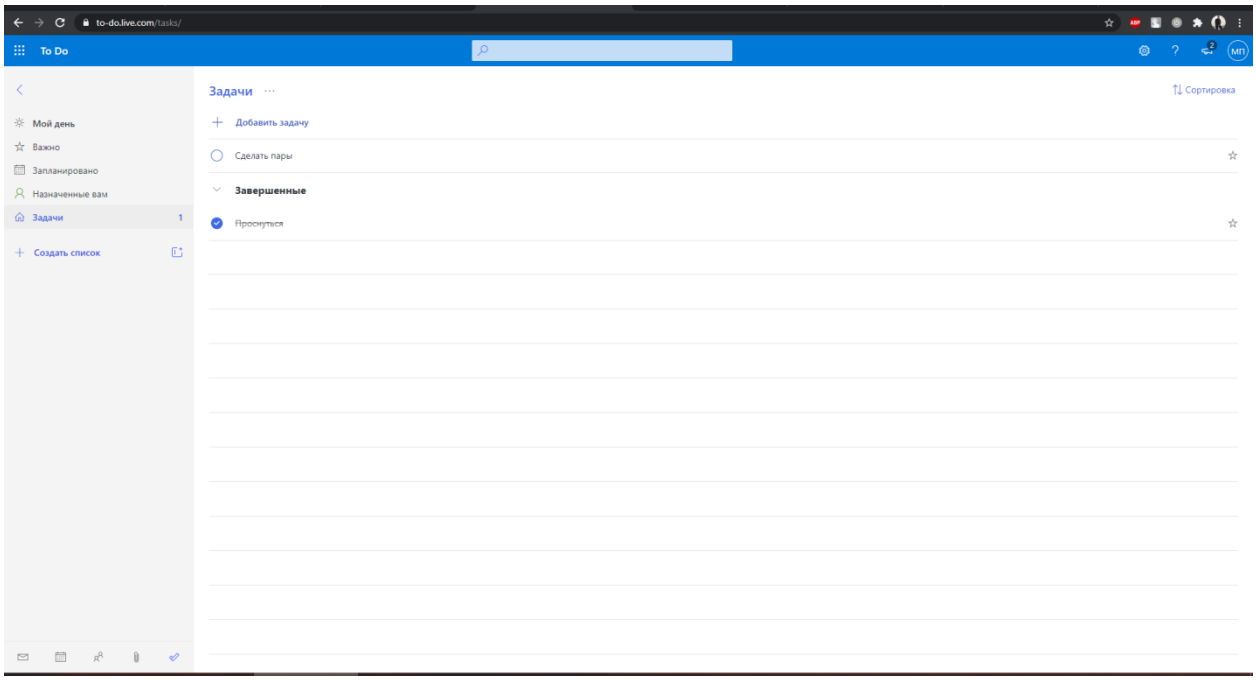



Рисунок 1.3 – Приклад використання Microsoft ToDo

РемОнлай – доволі зручний сервіс, який поєднує в собі майже всі необхідні функції. Із мінусів можна виділити лише доволі високу вартість користування продуктом і відсутність можливості збереження даних у вигляді звіту.[4]

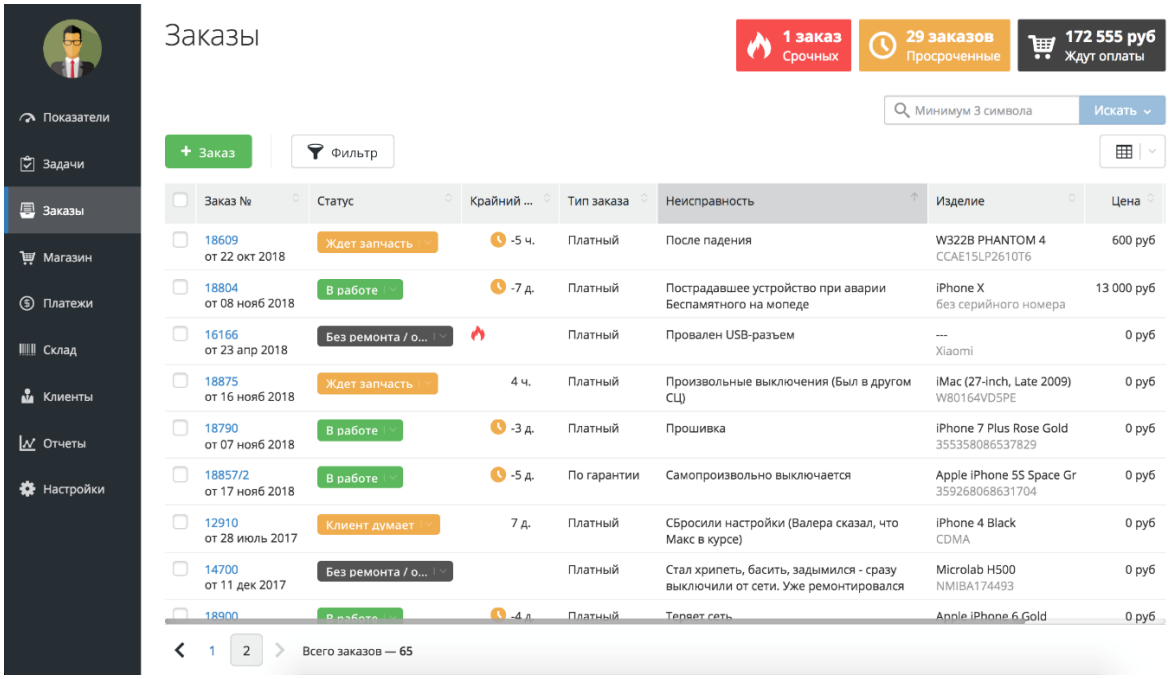



Рисунок 1.4 – Приклад використання РемОнлайн

На основі зібраних даних була сформульована наступна табл. 1.1. Як видно з таблиці, жоден з проаналізованих додатків в повній мірі не задовольняє функціональні потреби, що висунув замовник. Тому варто виконати власну розробку.

Таблиця 1.1 – Порівняльна характеристика додатків – аналогів

| Назва | Зручна форма запису | Нагадування | Простий інтер-фейс | Можли-вість зберег-ти звіт | Можли-вість роботи без доступу до мережі Інтер-нет | Безкоштов-ність | Наявність декількох статусів замовлень | Одночасний доступ великої кількості користувачів | Сума |
|---|---|---|---|---|---|---|---|---|---|
| **MS Excel** | - | - | - | + | + | - | + | - | 3 |
| **Google Tables** | - | - | - | + | - | + | + | + | 4 |
| **Microsoft ToDo** | - | + | + | - | + | + | - | - | 4 |
| **РемОнлайн** | + | - | - | + | - | - | + | + | 4 |

## 1.3 Постановка задачі

Метою проекту є створення десктопного додатку, який забезпечить роботу по збору, збереженню та обробці всієї інформації стосовно активних замовлень сервісу. Інформація повинна бути доступна з декількох робочих машин і оновлюватись через Інтернет, але треба забезпечити й можливість функціонування без доступу до мережі. Додаток не повинен мати зайвих елементів і максимально чітко відображати картину.

Задачі, які необхідно виконати для створення проекту:

- створити простий та зрозумілий інтерфейс, який відображатиме необхідну інформацію максимально вдало,
- створити базу даних, яка зберігатиме всю інформацію стосовно замовлень,
- розмістити базу даних на сервер та під'єднати її до додатку,
- провести тестування та виправлення недоліків,
- створити інсталятор додатку,
- оформити всю необхідну документацію щодо проекту.

Середовище розробки - Visual Studio 2019. Для створення графічного інтерфейсу користувача обрано API-інтерфейс WPF – це графічна система, за допомогою якої можна будувати графічні додатки для операційної системи Windows. Дана система побудована на основі мови програмування C#. Вона є доволі гнучкою, що дозволить створити необхідний проект.[1]

База даних буде побудована на основі SQL, та інтерфейсу доступу MySQL. Це дасть змогу створити необхідну базу даних, яку дуже просто обновляти та експлуатувати.[2]

Повне технічне завдання на виконання робіт міститься в додатку А, планування виконання робіт наведено в додатку Б.

# 2 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ

## 2.1 Структурно-функціональна модель

Найголовніша та сама перша діаграма, яка будується при створенні будь-якого проекту – це контекстна діаграма IDEF0. Центром контекстної діаграми являється функція. Вона відображає логіку роботи системи в цілому. В даному випадку, вона має назву «Управління виконанням замовлень».[10]

Вхідними даними основного процесу виступають:

- інформація про замовлення,
- інформація про користувачів,
- порядок виконання замовлення.

Вихідними даними виступають:

- інформація про виконання замовлення,
- архів замовлень,
- звітна документація.

Механізми проекту:

- додаток, що розробляється,
- працівники компанії,
- програмне забезбечення,
- технічне завдання.

На рисунку 2.1 зображено фінальний вигляд контекстної діаграми:


Рисунок 2.1 – Контекстна діаграма

Після контекстної діаграми створюємо діаграму декомпозиції. Першим етапом є визначення основних процесів, які відбуваються в середині проекту:[11]

- авторизація та реєстрація,
- створення нового замовлення,
- редагування замовлення,
- створення звітної документації.

Для кожного з цих процесів необхідно визначити вхідні, вихідні дані, елементи управління та механізми. Результат відображено в таблицях 2.1, 2.2, 2.3, 2.4, 2.5 і 2.6:

Таблиця 2.1 – Характеристики процесу авторизації/реєстрації

| Вхід | Вихід | Елементи управління | Механізми |
|---|---|---|---|
| Інформація про користувача | Категорія користувача «працівник» | | Технічне завдання |
| | Категорія користувача «адміністратор» | | Працівники компанії |
| | | | Додаток, що розробляється |

Таблиця 2.2 – Характеристики процесу створення нового замовлення

| Вхід | Вихід | Елементи управління | Механізми |
|---|---|---|---|
| Порядок виконання замовлення | Архів замовлень | Посадові інструкції | Технічне завдання |
| Інформація про замовлення | | Кошторис робіт | Працівники компанії |
| Категорія працівника «працівник» | | | Додаток, що розробляється |

Таблиця 2.3 – Характеристики процесу редагування замовлення

| Вхід | Вихід | Елементи управління | Механізми |
|---|---|---|---|
| Категорія користувача «працівник» | Інформація про виконання замовлення | Посадові інструкції | Технічне завдання |
| Категорія користувача «адміністратор» | | Кошторис робіт | Працівники компанії |
| Інформація про замовлення | | | Додаток, що розробляється |

Таблиця 2.4 – Характеристики процесу створення звітної документації

| Вхід | Вихід | Елементи управління | Механізми |
|---|---|---|---|
| Категорія користувача «адміністратор» | Статистика | Посадові інструкції | Технічне завдання |
| | | Кошторис робіт | Працівники компанії |
| | | | Додаток, що розробляється |

Фінальний вигляд діаграми декомпозиції зображено на рисунку 2.2:

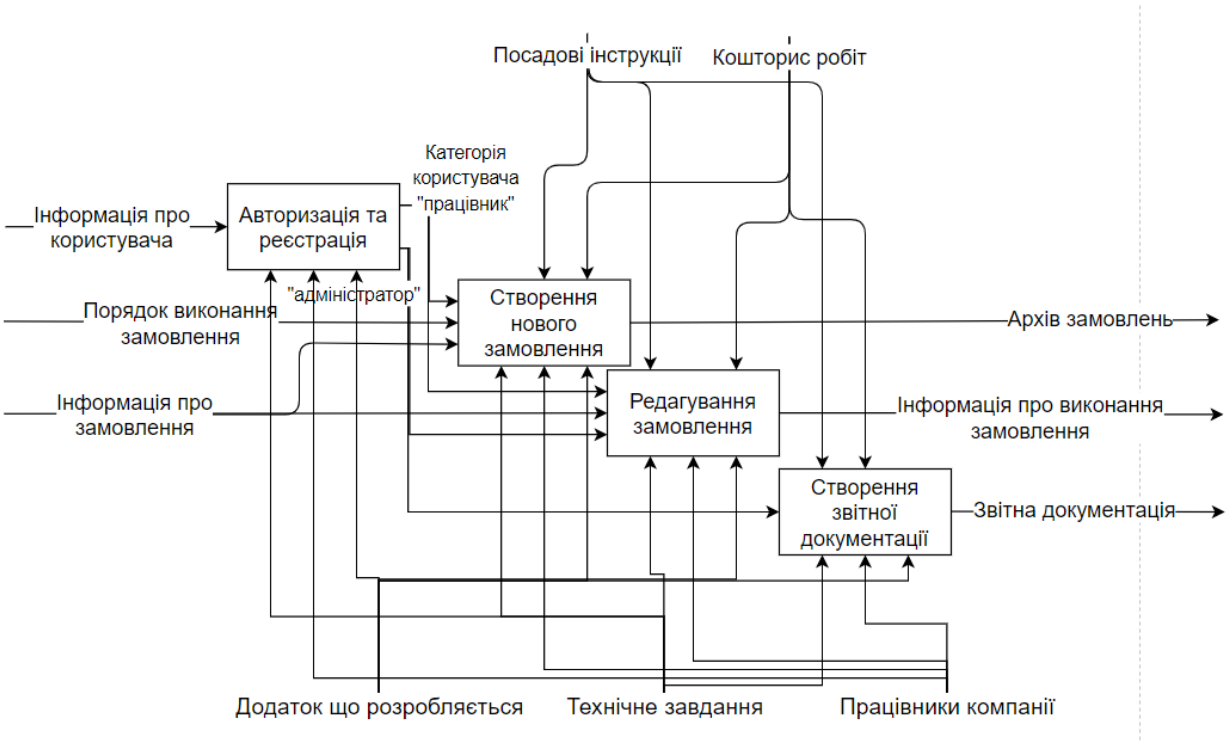



Рисунок 2.2 – Діаграма декомпозиції

## 2.2 UML-моделі

Діаграма варіантів використання – це діаграма, яка відображає всіх користувачів програми у зв’язку із функціями програми, які вони можут використовувати. Діаграма, яка відображає варіанти використання даного проекту зображена на рисунку 2.3:[12]

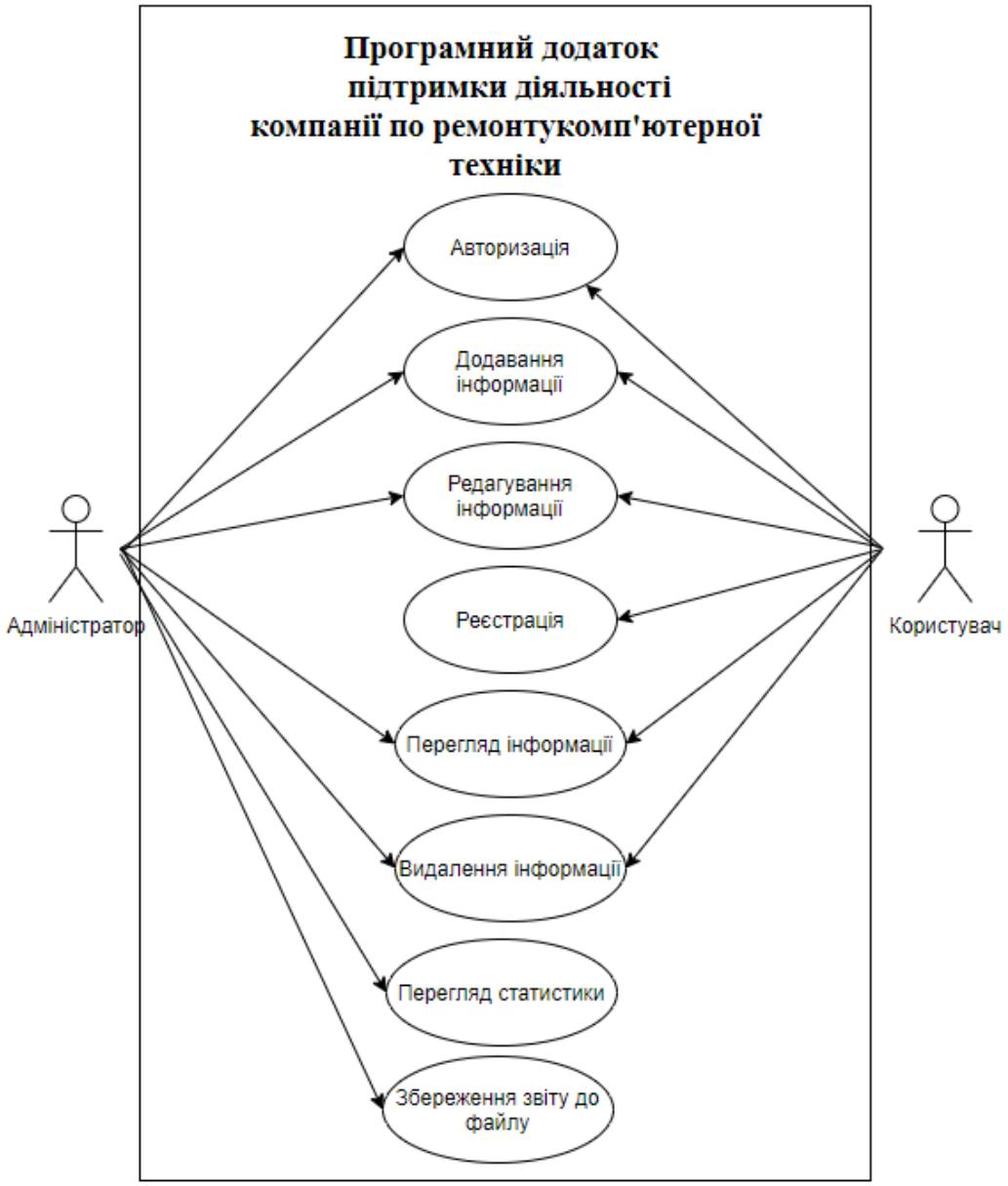



Рисунок 2.3 - Діаграма варіантів використання

Діаграми автоматів використовуються для опису поведінки системи у випадках всіх можливих положень класів цієї систем. За допомогою неї можна зрозуміти як ми можемо використовувати продукт від запуску і до завершення роботи.[12]

Діаграма автоматів додатку представлена на рисунку 2.4:

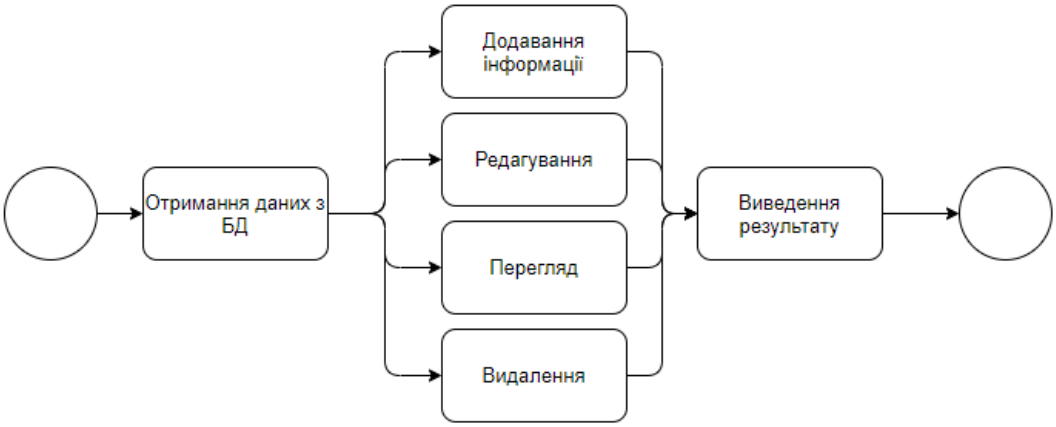



Рисунок 2.4 – Діаграма автоматів

Клас аналізу це укрупнена абстракція, яка на концептуальному рівні описує деякий фрагмент системи. Тож, діаграма класів аналізу – це діаграма, яка описує набір класів та їх зв’язки між собою.[12]

Діаграма класів аналізу представлена на рисунку 2.5:

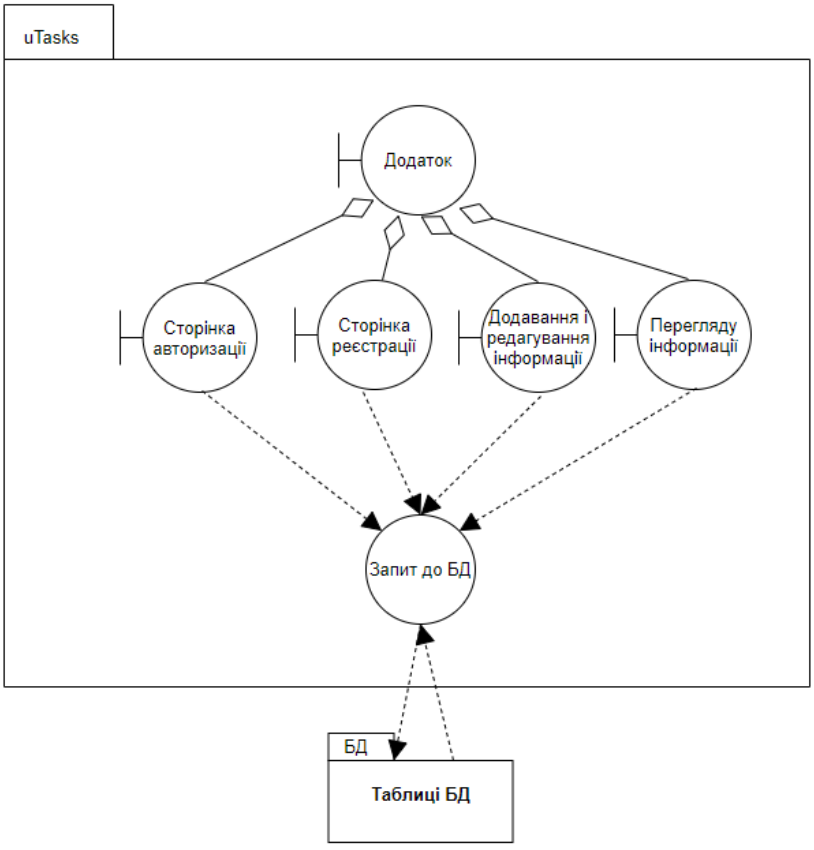



Рисунок 2.5 – Діаграма класів аналізу

Діаграма послідовності – це один із різновидів діаграм UML. Вона відображає порядок виконання дій всередині додатку.[12]

На рисунку 2.6 представлена діаграма послідовності додатку:

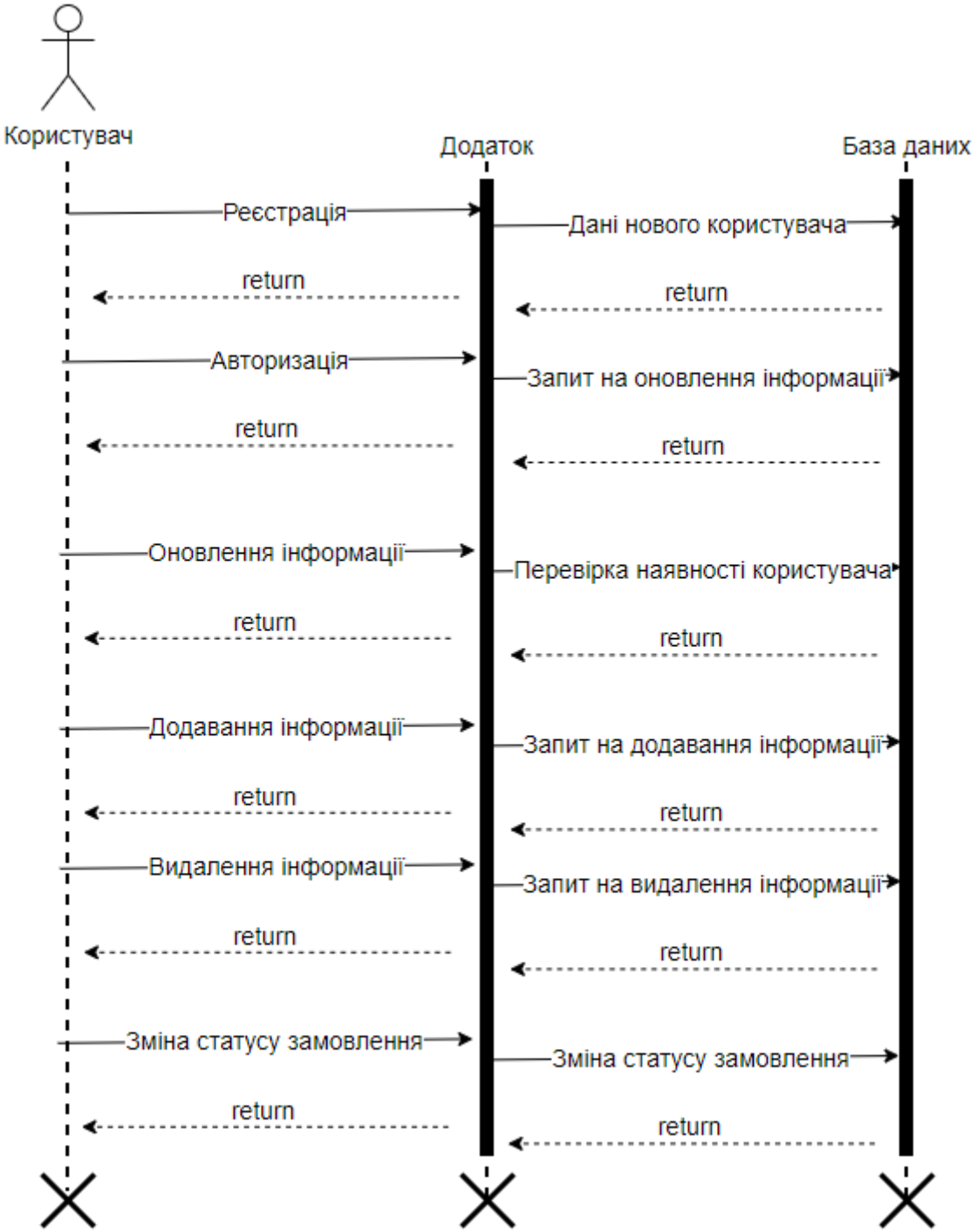



Рисунок 2.6 – Діаграма послідовності

Діаграма комунікації – це діаграма, на якій зображають взаємодію між частинами нашого програмного додатку. Тобто – як ці частини між собою «спілкуються».[12]

На рисунку 2.7 представлена діаграма комунікації:

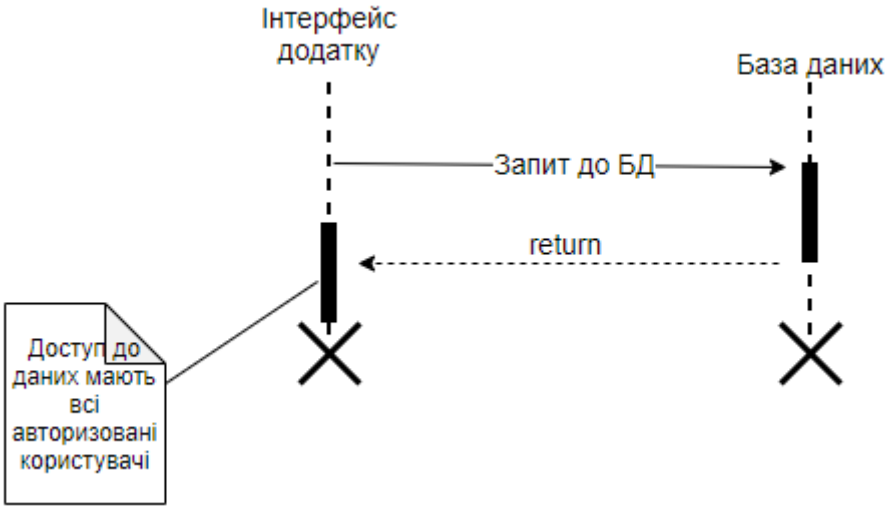



Рисунок 2.7 – Діаграма комунікації

За допомогою діаграми діяльності доволі легко візуалізувати всі можливі варіанти діяльності користувача в додатку. Це одна із різновидів моделей стандарту UML, тому для своєї побудови використовує ті ж самі інструменти.[12]

На рисунку 2.8 представлена діаграма діяльності додатку.

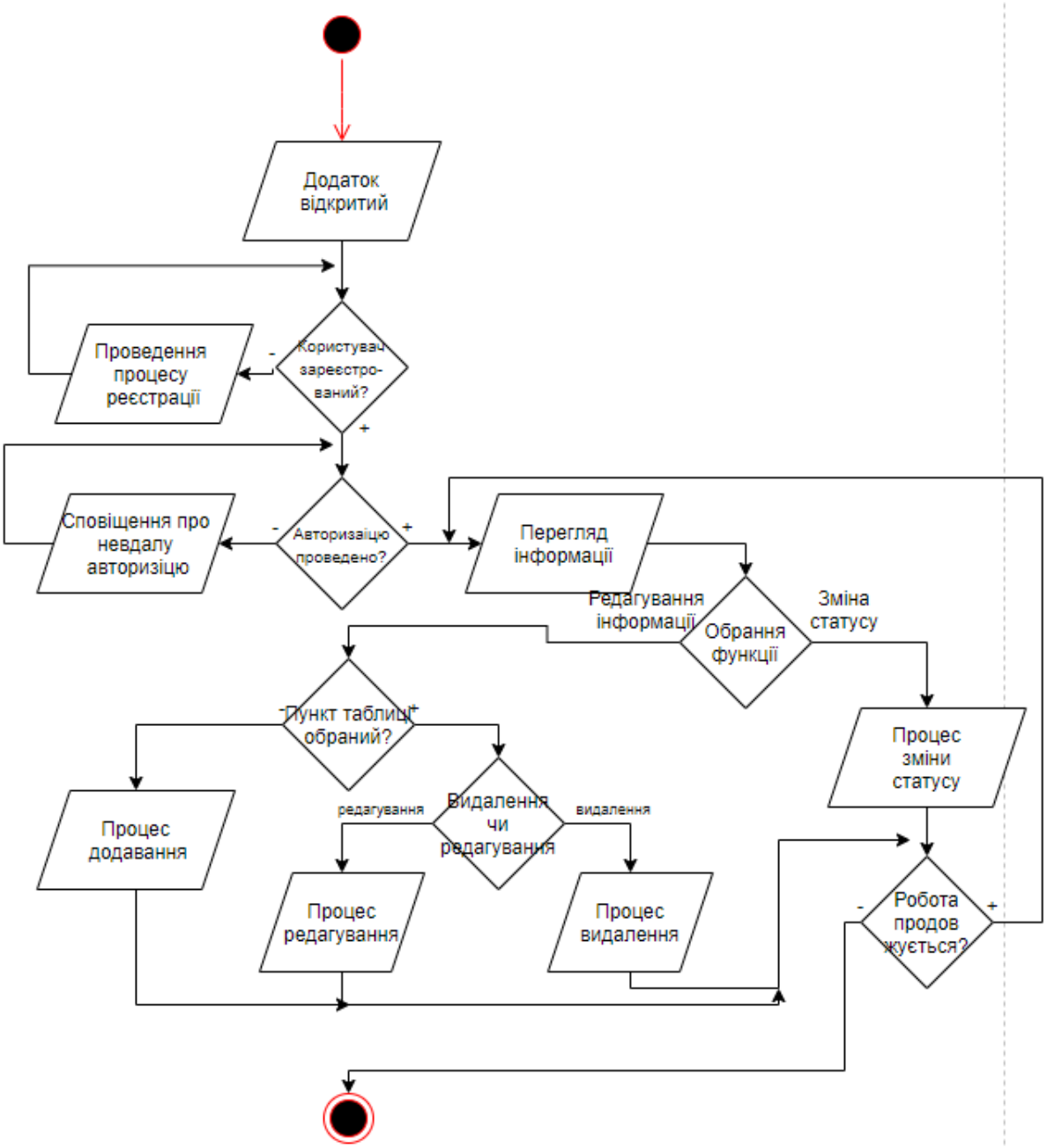



Рисунок 2.8 – Діаграма діяльності

Діаграма розгортання показує основні вузли програми та їх взаємодію. Також, вона показує взаємодію користувачів із основними вузлами.[12]

На рисунку 2.9 представлена діаграма розгортання.

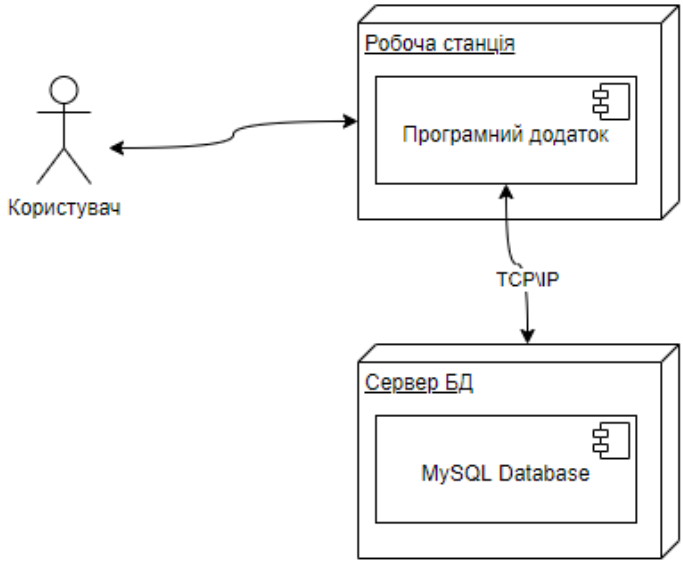



Рисунок 2.9 – Діаграма розгортання

## 2.3 Проектування бази даних

База даних, розроблена в процесі дипломного проектування, містить 7 таблиць: користувачі – для збереження інформації щодо майстрів, клієнти – інформація щодо клієнтів, замовлення – основна таблиці БД, в якій зберігається вся необхідна інформація щодо замовлень, деталі – зберігається інформація щодо запчастин, та послуги – інформація і вартість послуг. Також, присутні дві допоміжні таблиці – список запчастин і список послуг, які допомагають зберігати список запчастин і послуг відповідно, які прикріплені до певного замовлення. [11]

Описання полів таблиць бази даних наведено в табл. 2.7 – 2.13.

Таблиця 2.7 – Опис таблиці “Customers”

| Ім’я поля сутності | Тип даних | Інформація, що зберігається |
|---|---|---|
| Customer_id | INT | ID замовника |
| First_name | VARCHAR | Ім’я |
| Phone_number | VARCHAR | Номер телефону для зв’язку |

Таблиця 2.8 – Опис таблиці “Users”

| Ім’я поля сутності | Тип даних | Інформація, що зберігається |
|---|---|---|
| User_id | INT | ID користувача |
| Fname | VARCHAR | Ім’я |
| Sname | VARCHAR | Фамілія |
| Position | BOOLEAN | Посада в сервісі (адміністратор або користувач) |
| Login | VARCHAR | Логін користувача |

Таблиця 2.9 – Опис таблиці “Parts_list”

| Ім’я поля сутності | Тип даних | Інформація, що зберігається |
|---|---|---|
| Parts_list_id | INT | ID списку деталей |
| Part_id | INT | ID деталі |
| Order_id | INT | ID замовлення |

Таблиця 2.10 – Опис таблиці “Parts”

| Ім’я поля сутності | Тип даних | Інформація, що зберігається |
|---|---|---|
| Part_id | INT | ID замовника |
| Name | VARCHAR | Назва деталі |
| Price | INT | Вартість |

Таблиця 2.11 – Опис таблиці “Orders”

| Ім’я поля сутності | Тип даних | Інформація, що зберігається |
|---|---|---|
| Order_id | INT | ID послуги |
| Description | VARCHAR | Описання замовлення |
| Status | ENUM | Статус виконання замовлення |
| Receip_time | TIMESTAMP | Час отримання замовлення |
| Supose_return_time | TIMESTAMP | Орієнтовний час видачі замовлення |
| User_id | INT | ID майстра |
| Customer_id | INT | ID замовника |

Таблиця 2.12 – Опис таблиці “Services”

| Ім’я поля сутності | Тип даних | Інформація, що зберігається |
|---|---|---|
| Service_id | INT | ID послуги |
| Name | VARCHAR | Назва послуги |
| Price | INT | Вартість послуги |

Таблиця 2.13 – Опис таблиці “Services_list”

| Ім’я поля сутності | Тип даних | Інформація, що зберігається |
|---|---|---|
| Service_list_id | INT | ID списку послуг |
| Service_id | INT | ID послуги |
| Order_id | INT | ID замовлення |

Логічна модель бази даних має вигляд (рис. 2.10):

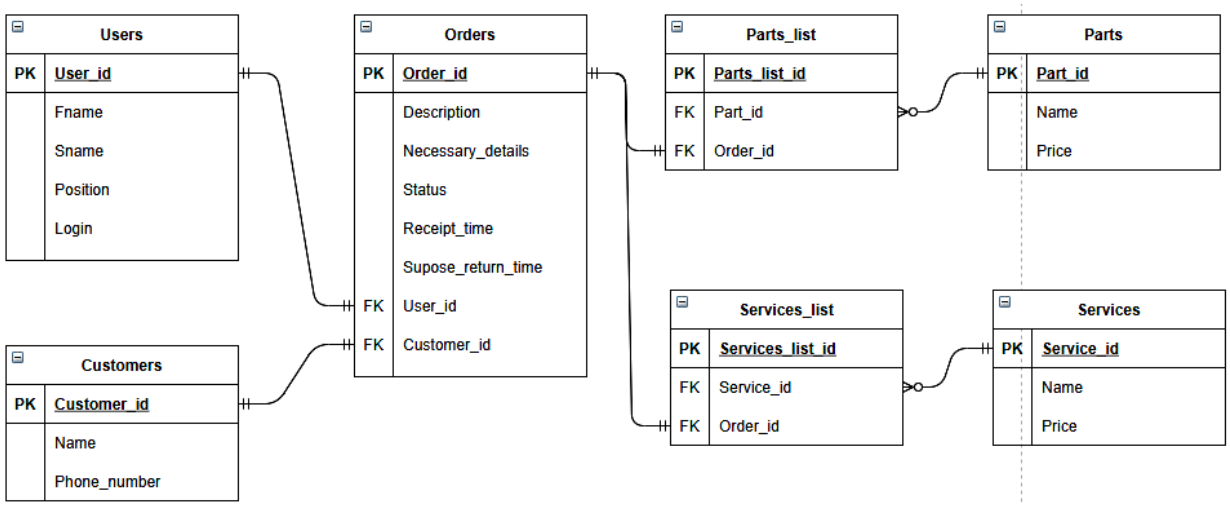



Рисунок 2.10 – Логічна модель бази даних

# 3 РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ

## 3.1 Архітектура додатку

Архітектура даного програмного додатку складається з модулів, описаних у табл. 3.1.

Таблиця 3.1 – Модулі програмного додатку

| **Назва** | **Описання** |
|---|---|
| **Файли розмітки** | |
| Login.xaml | Вікно авторизації |
| MainWindow.xaml | Основне вікно |
| PartsList.xaml | Вікно для обрання списку запчастин |
| ServicesList.xaml | Вікно для обрання списку послуг |
| Regestration.xaml | Вікно для реєстрації нового користувача |
| Statistics.xaml | Статистика та збереження звіту |
| **Файли логіки додатку** | |
| Login.xaml.cs | Логіка авторизації користувача, перевірка в БД, чи є такий користувач |
| MainWindow.xaml.cs | Заповнення основної таблиці програми, синхронізація з БД (читання, видалення та редагування даних через додаток) |
| PartsList.xaml.cs | Редагування списку необхідних деталей для виконання замовлення, редагування таблиці “parts_list” в БД |
| ServicesList.xaml .cs | Редагування списку послуг для виконання замовлення, редагування таблиці “services_list” в БД |

Продовження табл. 3.1

| | |
|---|---|
| Regestration.xaml.cs | Логіка реєстрації нового користувача (додавання нового запису в таблицю “users” ) та перевірка, чи існує введений логін в БД. |
| Statistics.xaml.cs | Логіка формування статистики. Вхідні дані – таблиця із головного вікна і період, обраний користувачем. |
| **Інші модулі** | |
| DB.cs | Клас, розроблений для швидкого підключення до БД в будь-якому місці програми. |
| App.xaml.cs | Файл, який відповідає за налаштування проекту та його запуск. |
| База даних | Модуль програми, який розміщений на віддаленому сервері. Дані в базі синхронізуються з даними в додатку за допомогою інтернет-зв’язку. |

Вказані в таблиці вище модулі взаємодіють між собою. Для наглядного представлення складено діаграму компонентів (рис. 3.1) у відповідності до стандарту UML [12].

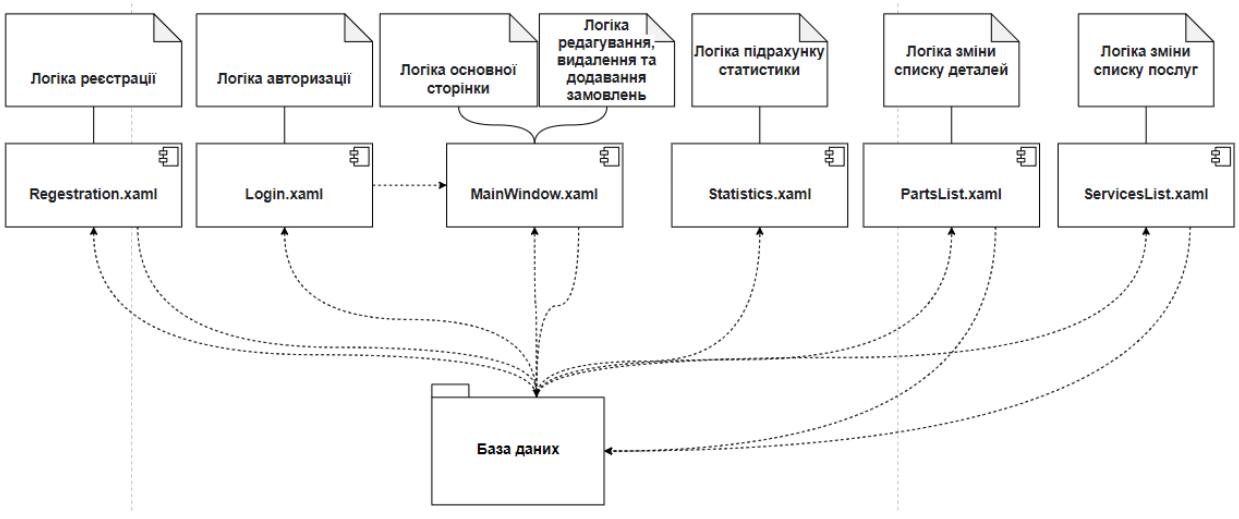



Рисунок 3.1 – Архітектура додатку

## 3.2 Програмна реалізація

### 3.2.1 Створення бази даних

Для створення бази даних було обрано реляційну систему управління базами даних(БД) MySQL через те, що реалізовувати бази даних за допомогою цієї системи доволі легко, а також існує зручний функціонал для реалізації сховища. При цьому система може опрацьовувати декілька запитів одночасно. Для створення і тестування бази даних було використано середовище Open Server. На етапі розробки і тестування продукту, база даних була розміщена на локальному сервері Open Server. Розробка БД виконувалась за допомогою phpMyAdmin.[13]

Відповідно до п 2.3, база даних повинна містить 7 таблиць (“Users”, “Customers”, “Orders”, “Parts”, “Serivces”, “PartsList”, “ServicesList”). Для їх створення було написано скрипт на мові програмування MySQL, який наведено в додатку В. Після створення, БД має наступний вигляд в дизайнері phpMyAdmin (рис. 3.2):

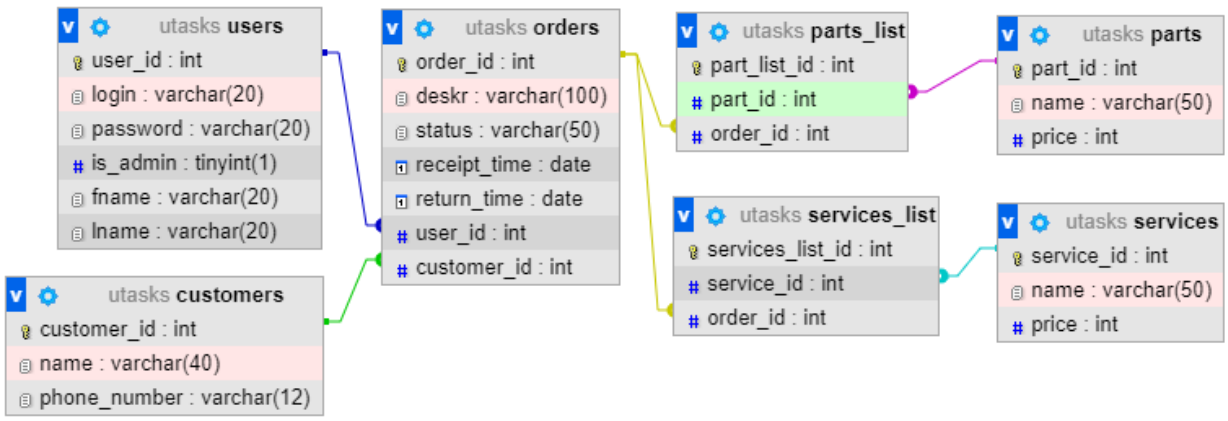
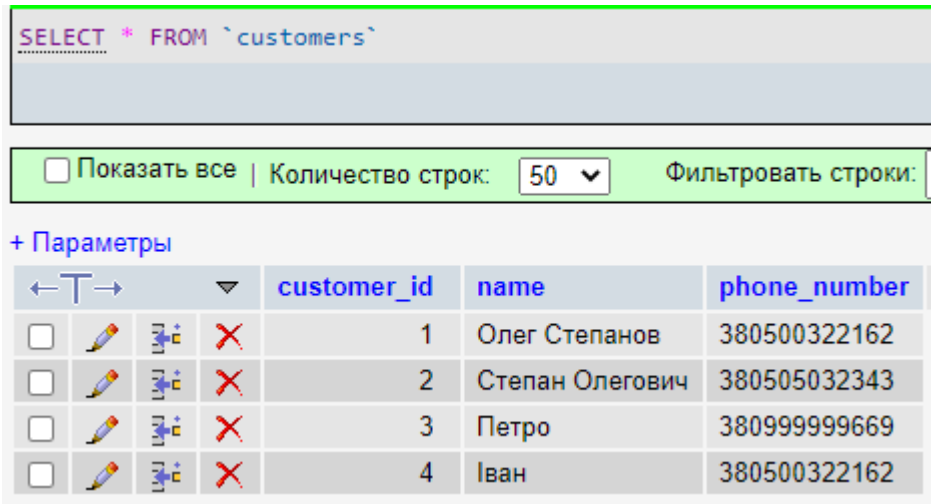



Рисунок 3.2 – Вигляд створеної БД у дизайнері phpMyAdmin

Для більш зручного тестування і розробки візуальної частини продукту необхідно заповнити таблиці тестовими даними (рис. 3.3, 3.4, 3.5, 3.6, 3.7. 3.8, 3.9).



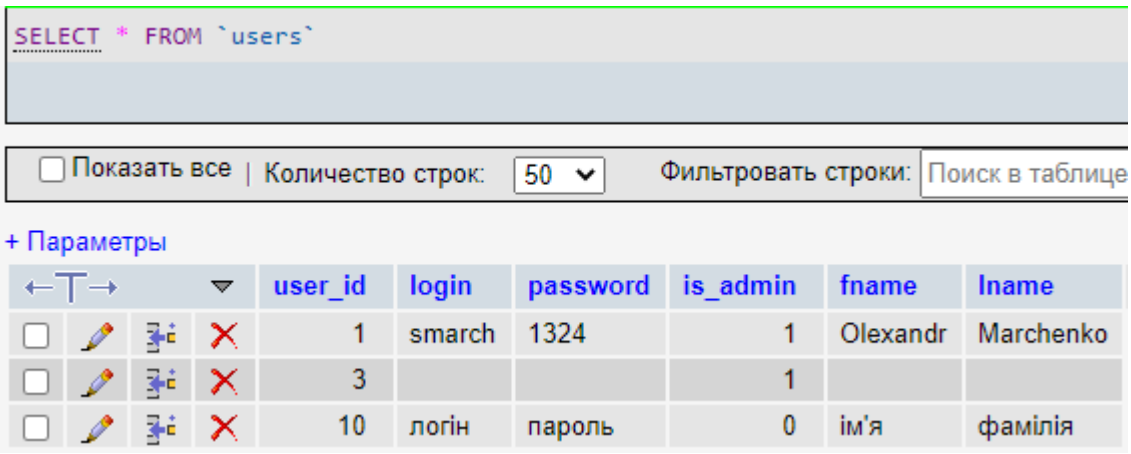
```
SELECT * FROM `customers`
```

Показать все | Количество строк: 50 Фильтровать строки:

+ Параметры

| customer_id | name | phone_number |
|---|---|---|
| 1 | Олег Степанов | 380500322162 |
| 2 | Степан Олегович | 380505032343 |
| 3 | Петро | 380999999669 |
| 4 | Іван | 380500322162 |

Рисунок 3.3 – Заповнена таблиця клієнтів



```
SELECT * FROM `users`
```

Показать все | Количество строк: 50 Фильтровать строки: Поиск в таблице

+ Параметры

| user_id | login | password | is_admin | fname | lname |
|---|---|---|---|---|---|
| 1 | smarch | 1324 | 1 | Olexandr | Marchenko |
| 3 | | | 1 | | |
| 10 | логін | пароль | 0 | ім'я | фамілія |

Рисунок 3.4 – Заповнена таблиця користувачів

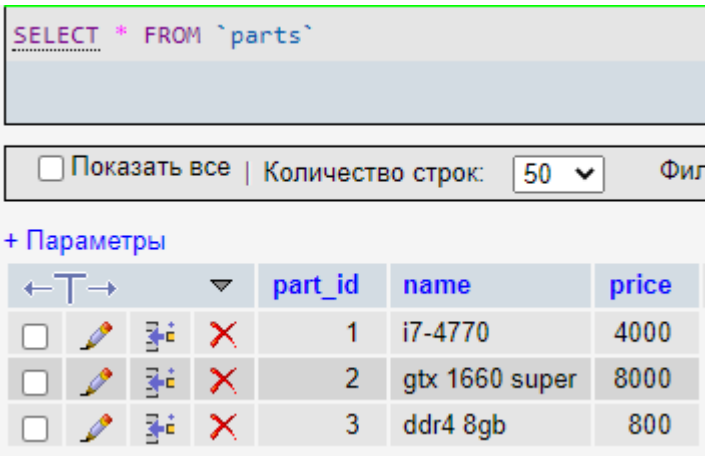



Рисунок 3.5 – Таблиця запчастин

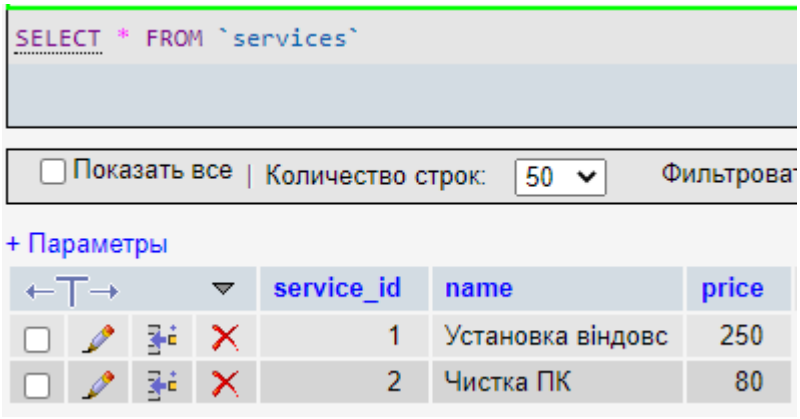



Рисунок 3.6 – Інформація щодо послуг

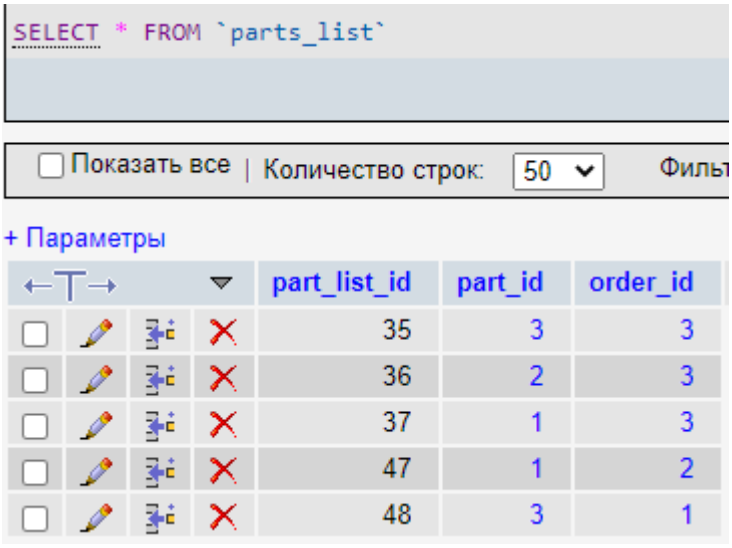



Рисунок 3.7 – Допоміжна таблиця для створення списків запчастин

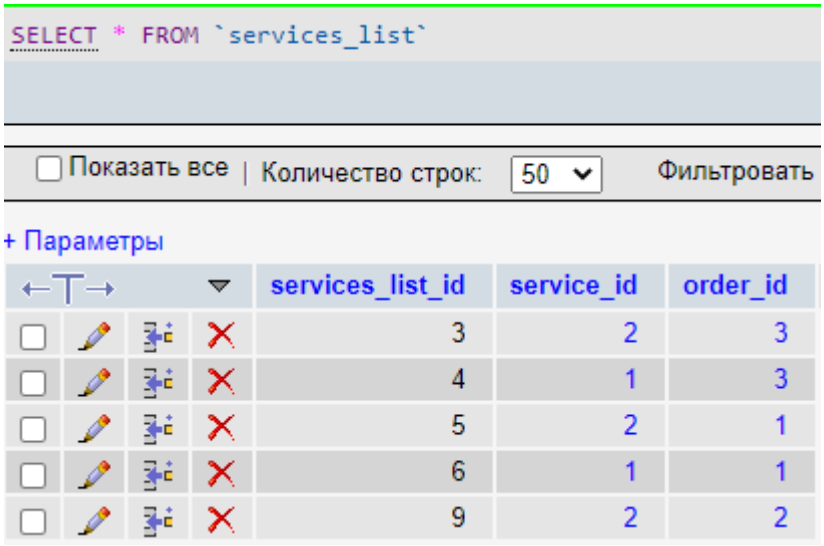



Рисунок 3.8 – Допоміжна таблиця для створення списків послуг

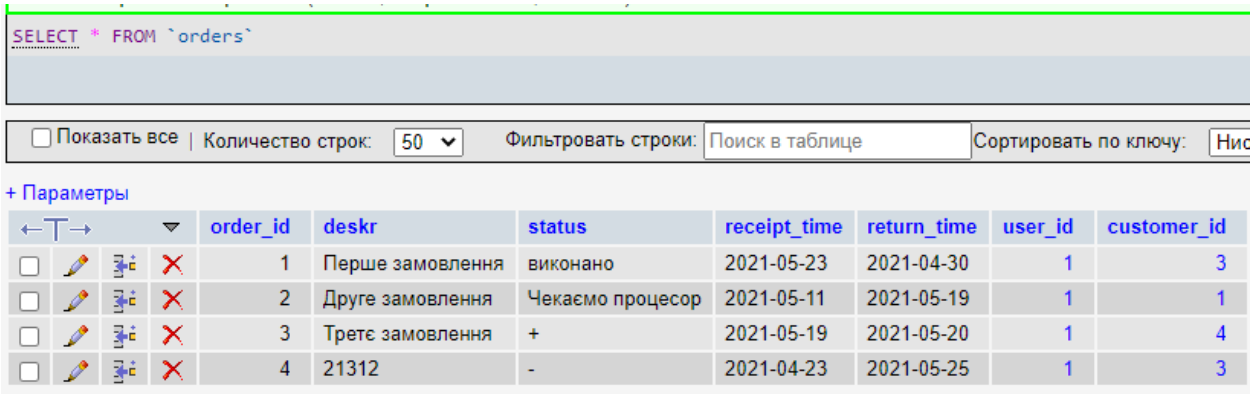



Рисунок 3.9 – Таблиця замовлень з інформацією

### 3.2.2 Створення вікон програми

Для реалізації проекту необхідно створити такі вікна:

- авторизація,
- реєстрація,
- головне вікно,
- вікна редагування списків запчастин та послуг,

- вікно статистики.

Для всіх цих модулів програми необхідно створити відповідні xaml-файли, до яких автоматично Visual Studio 2019 прив’язує файли логіки з розширенням xaml.cs. Також, створимо клас підключення до бази даних окремим файлом. В результаті маємо наступний проект (рис. 3.10).

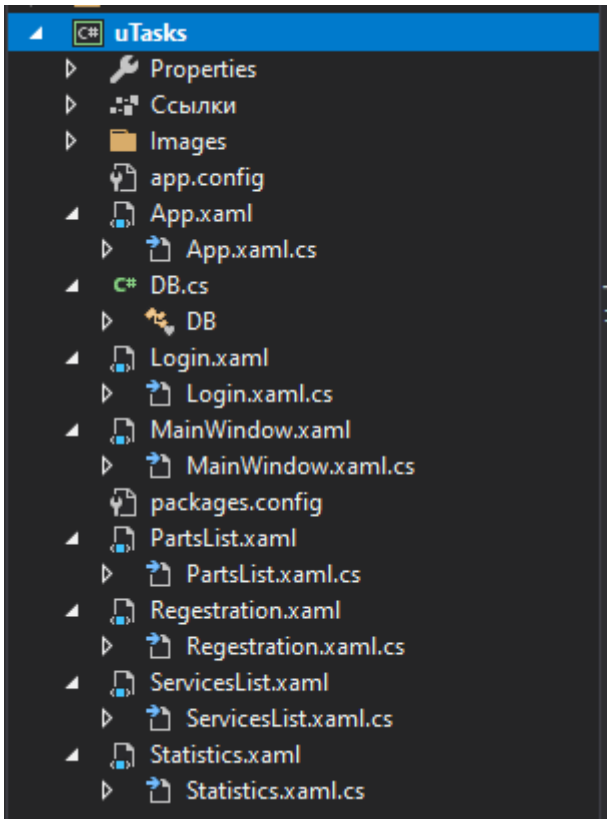



Рисунок 3.10 – Дерево проекту в Visual Studio 2019

Далі необхідно розмістити базові елементи управління WPF на створених вікнах. Після їх розміщення вікна мають наступний вигляд (рис. 3.11, 3.12, 3.13, 3.14, 3.15).[14]


Рисунок 3.11 – Базовий дизайн вікна авторизації

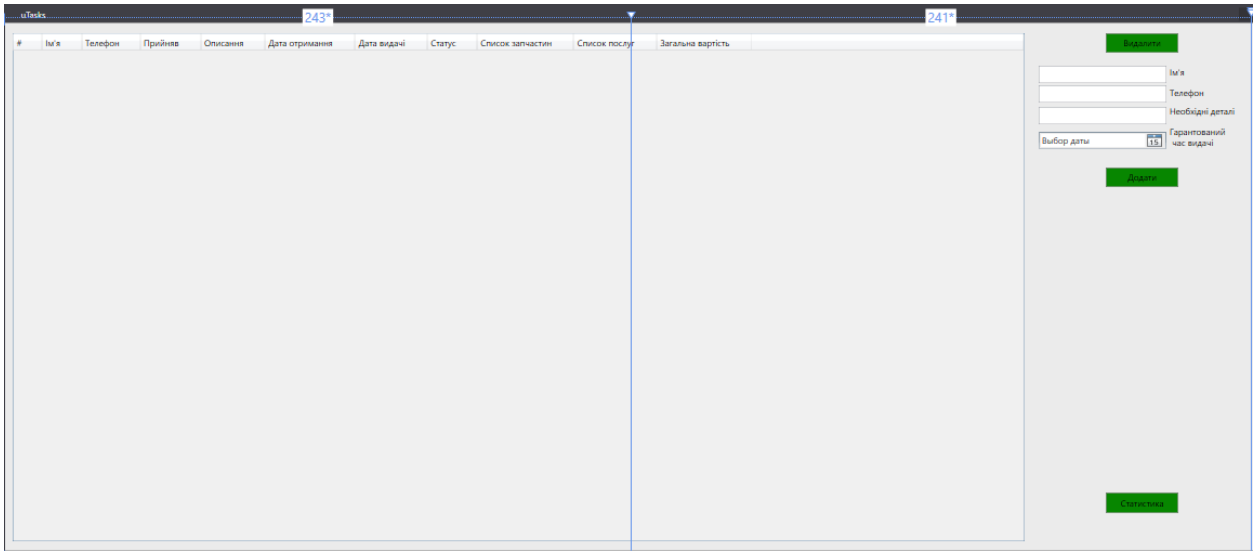



Рисунок 3.12 – Основне вікно

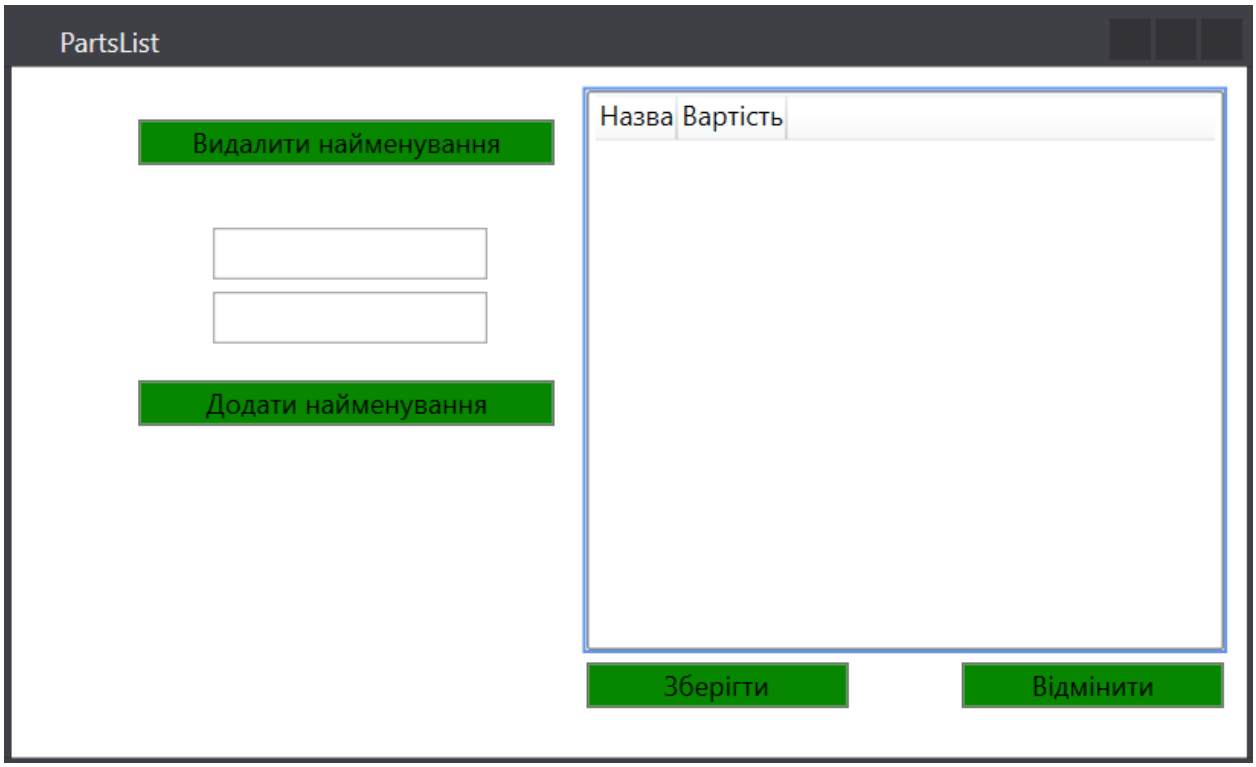



Рисунок 3.13 – Вікно вибору списку запчастин

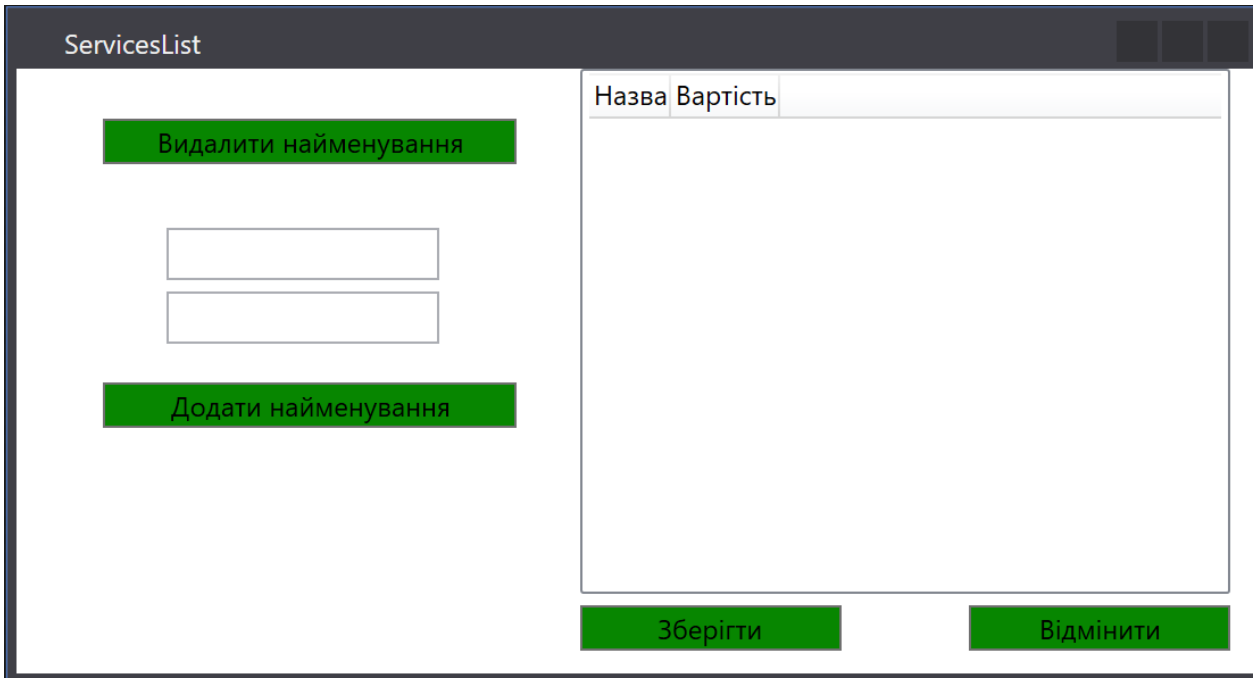



Рисунок 3.14 – Вікно обрання списку послуг

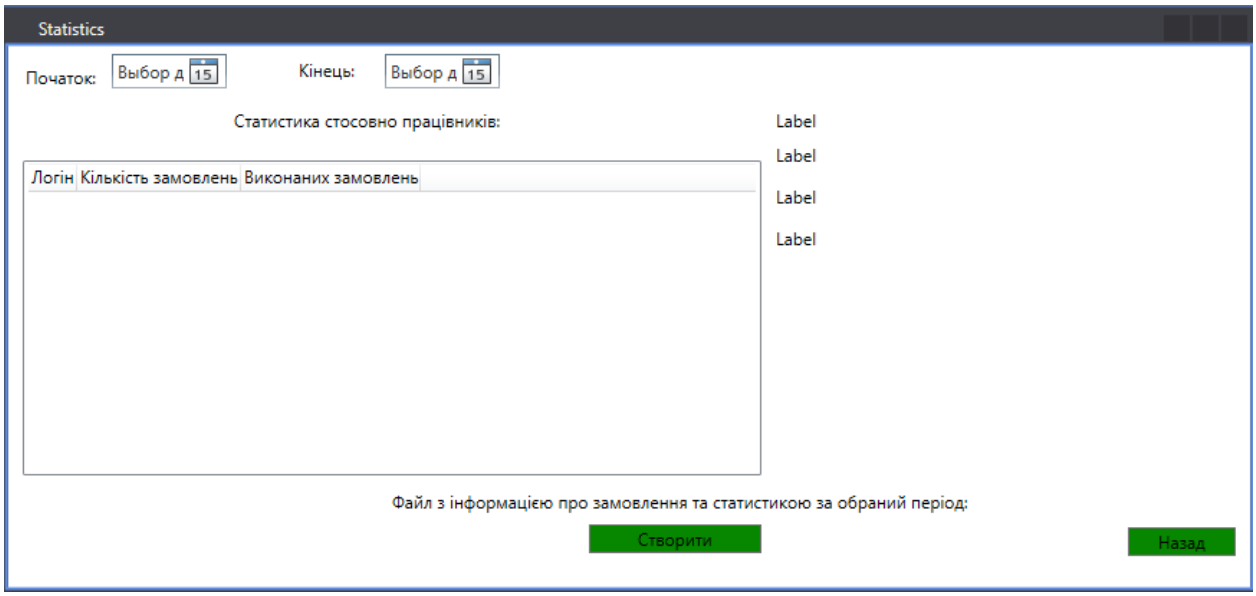



Рисунок 3.15 – Вікно статистики

Далі підключимо бібліотеку стилів Google Material Design. Для цього необхідно додати бібліотеку за допомогою інструменту керування проектами NuGet (рис. 3.16)[15]:

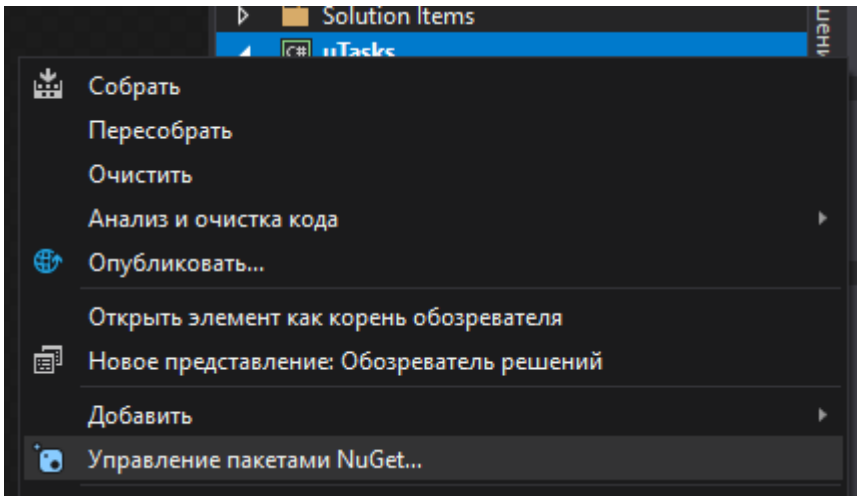



Рисунок 3.16 – Відкриття інструменту управління пакетами

Далі знаходимо бібліотеку та завантажуємо її до проекту (рис. 3.17).

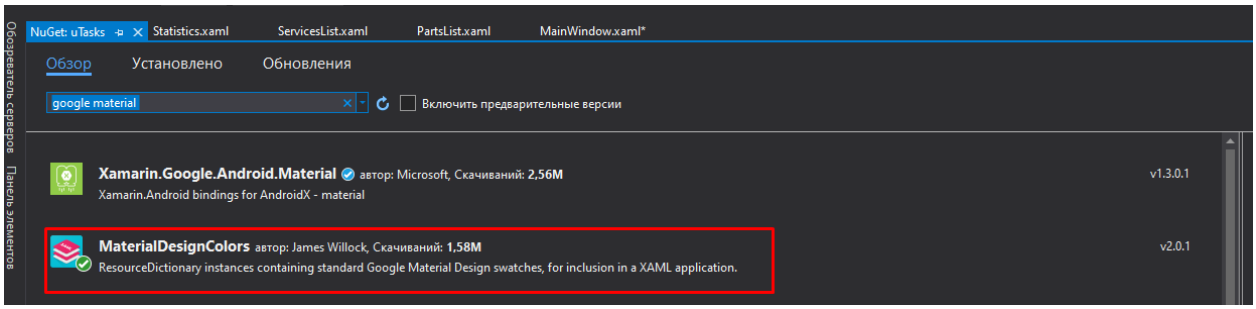



Рисунок 3.17 – Процес підключення бібліотеки стилів

Дана бібліотека автоматично змінює дизайн основних елементів програми, тому її необхідно лише підключити до файлу app.xaml (рис. 3.18), а також – підключити стилі в кожному з xaml-файлів (рис. 3. 19).

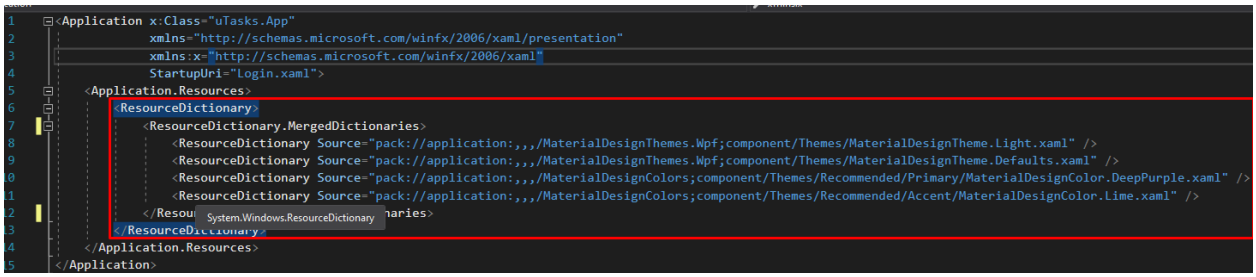



Рисунок 3.18 – Підключення бібліотеки до проекту

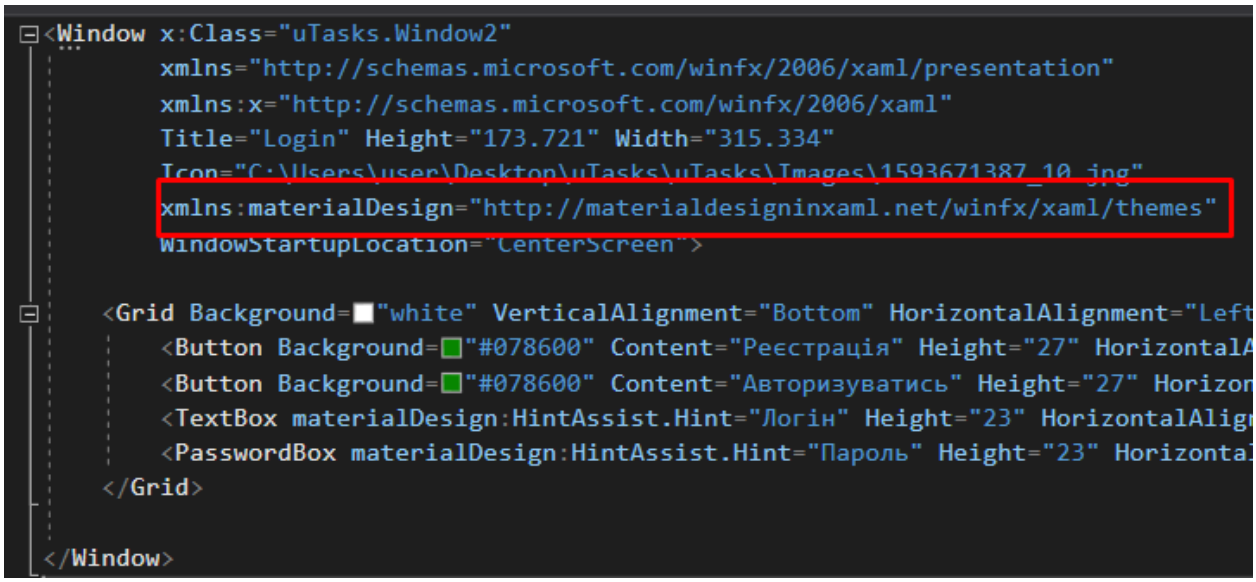



Рисунок 3.19 – Підключення Material Design до окремих файлів

Після підключення стилів – дизайн вікон значно змінюється і має наступний вигляд (рис. 3.20, 3.21, 3.22, 3.23, 3.24, 3.25).

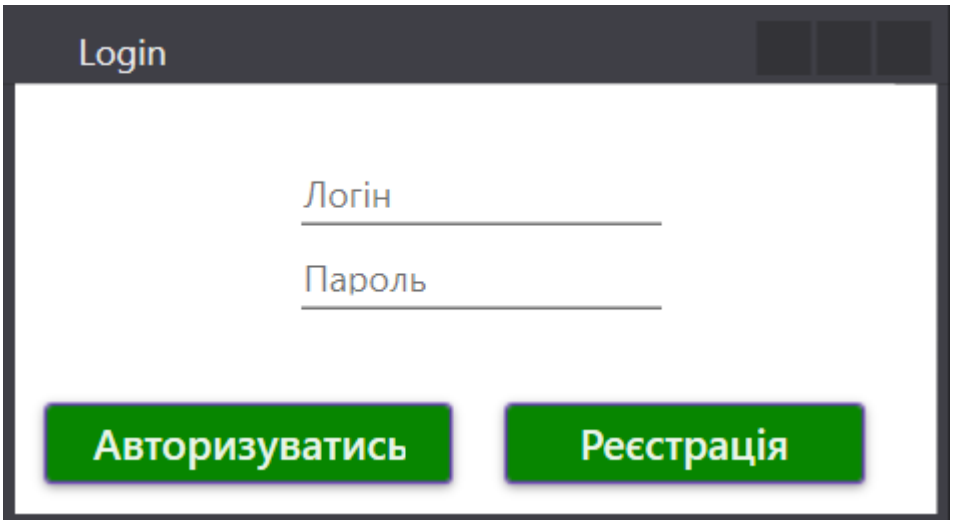



Рисунок 3.20 – Фінальний вигляд вікна авторизації

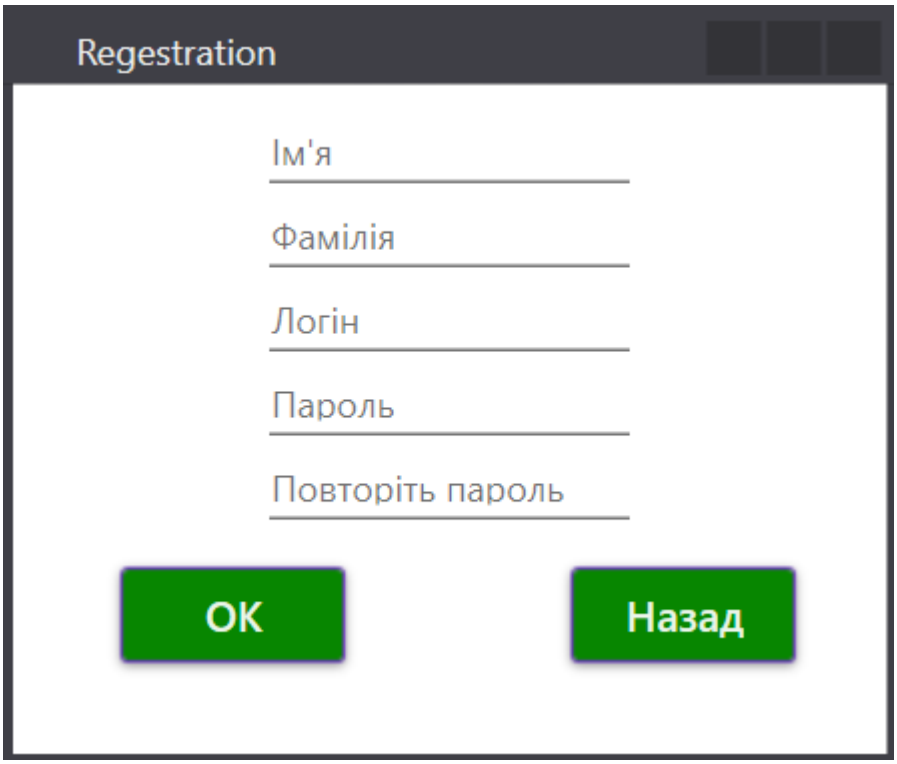



Рисунок 3.21 – Фінальний вигляд вікна реєстрації

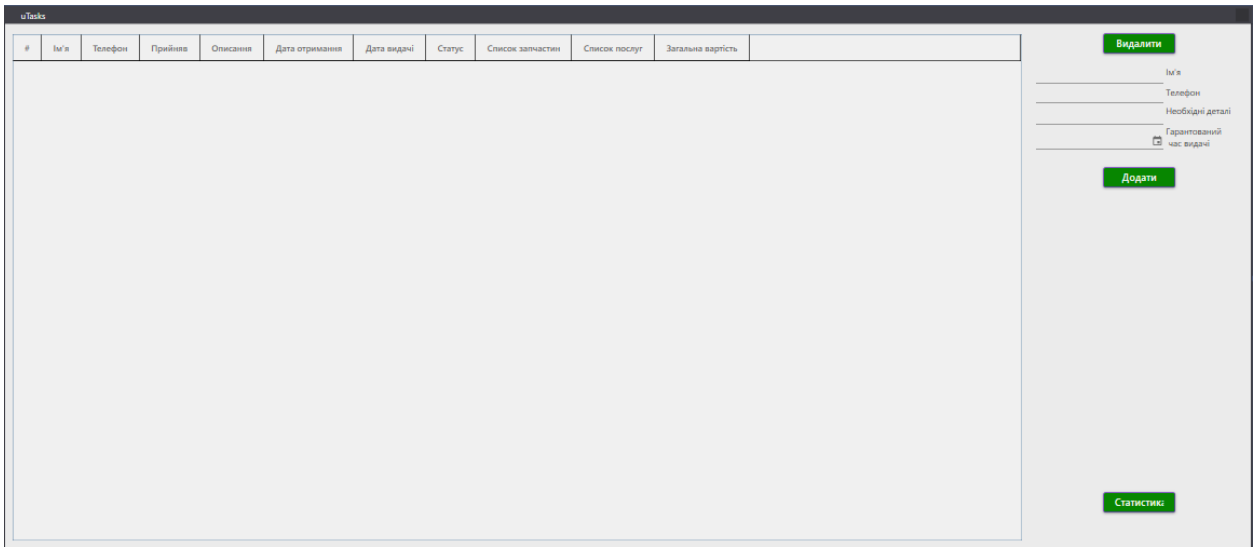



Рисунок 3.22 – Фінальний вигляд основного вікна

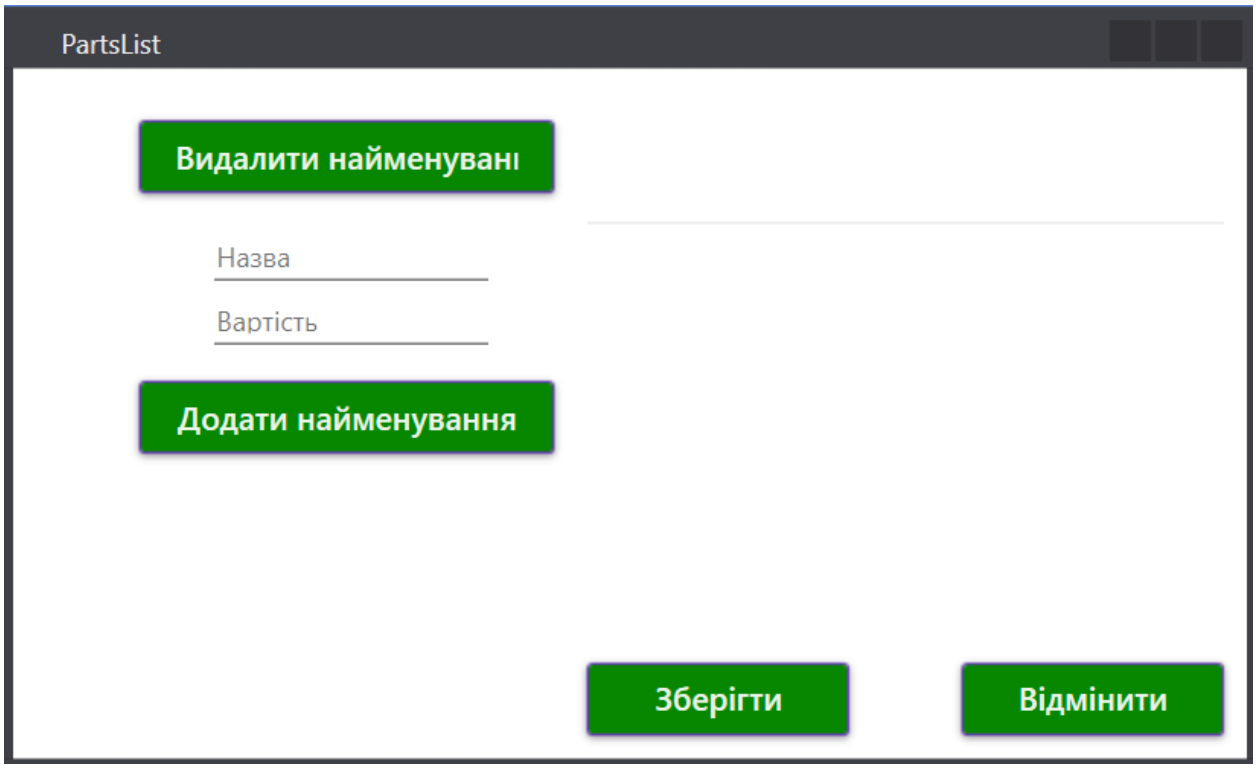



Рисунок 3.23 – Фінальний вигляд вікна вибору списку запчастин

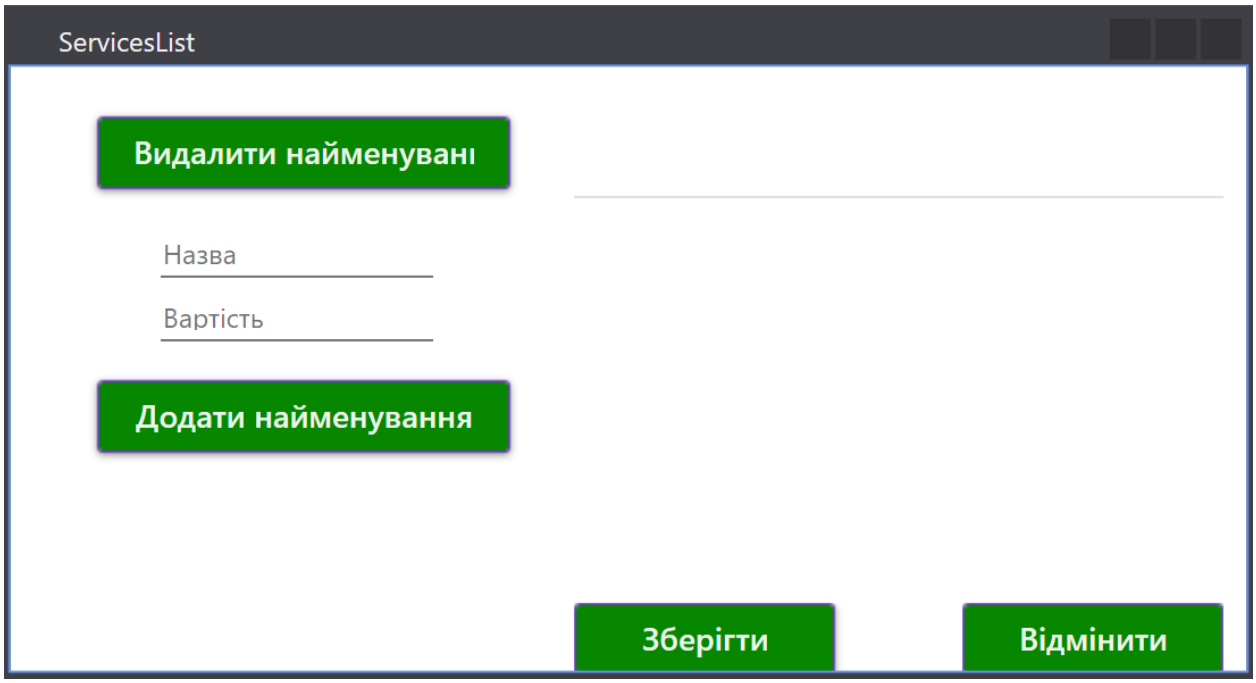



Рисунок 3.24 – Фінальний вигляд вибору списку послуг

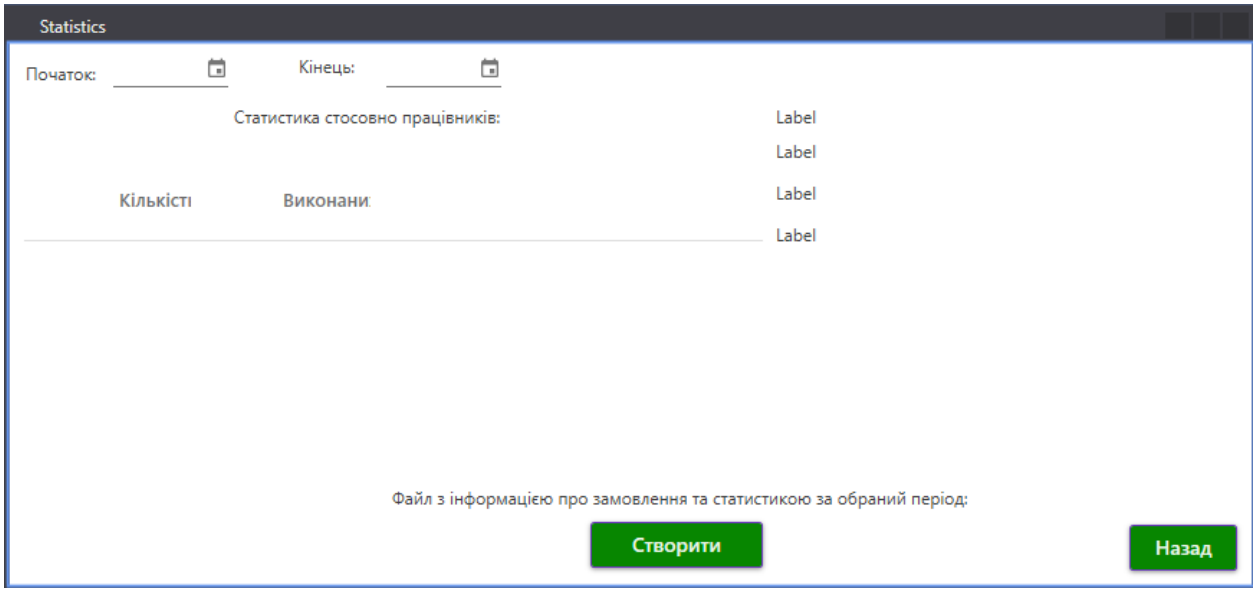



Рисунок 3.25 – Фінальний вигляд вікна статистики

### 3.2.3 Створення логіки роботи вікон

Лістинг коду всіх файлів програми наведено в додатку В. В таблиці 3.2, 3.3, 3.4, 3.5, 3.6, 3.7 приведено назви елементів, які були створені для файлів, в яких описана логіка програми:

Таблиця 3.1 – Опис методів і змінних класу вікна авторизації

| **Метод** | **Призначення** |
|---|---|
| bool isAdmin | Булева змінна, яка зберігає інформацію щодо того, чи є даний користувач адміністратором. |
| Window2() | Конструктор, який створює вікно. |
| void button1_Click(object sender, RoutedEventArgs e) | Обробник події натиснення на кнопку «авторизуватись». Відбувається підключення до БД, якщо логін і пароль введені правильно – користувач переходить на основне вікно. |
| void button2_Click(object sender, RoutedEventArgs e) | Обробник події натиснення на кнопку «реєстрація». Користувач потрапляє на вікно реєстрації. |

Таблиця 3.2 – Опис методів і змінних класу вікна реєстрації

| **Метод** | **Призначення** |
|---|---|
| Regestration.xaml.cs | |
| Regestration() | Конструктор, який створює вікно. |
| void Button_Click(object sender, RoutedEventArgs e) | Метод реєстрації користувача. |
| bool isUserExist() | Метод, який перевіряє чи є в БД логін, який ввів користувач для реєстрації |
| void Button_Click_1(object sender, RoutedEventArgs e) | Обробка події повернення на вікно реєстрації. |

Таблиця 3.3 – Опис методів і змінних класу основного вікна

| **Метод** | **Призначення** |
|---|---|
| DataTable ds | Таблиця, в якій зберігаються всі дані щодо замовлень. |
| User thisUser | Об’єкт користувальницького класу, в якому зберігається інформація щодо користувача, який виконав авторизацію в програмі. |
| class User | Описання класу для зберігання інформації про користувача. |
| Window1(bool isAdmin, int id, string login) | Конструктор, який створює вікно і зберігає дані про користувача. |
| void addBtn_click(object sender, RoutedEventArgs e) | Метод, який описує логіку додавання нового замовлення. |
| void cellEditEnding(object sender, DataGridCellEditEndingEventArgs e) | Обробник події редагування комірки в таблиці. |
| void Window_Loaded(object sender, RoutedEventArgs e) | Обробник події завантаження основного вікна. Запускає оновлення таблиці. |
| void updateTable() | Метод, який виконує оновлення таблиці. |
| void checkStatus() | Метод, який перевіряє, чи є замовлення виконаним. Якщо ні – виділяє рядок червоним кольором. |
| void PartsBtn_Click(object sender, RoutedEventArgs e) | Обробник події натиснення на кнопку редагування списку запчастин. |

Продовження табл. 3.3

| | |
|---|---|
| void button3_Click(object sender, RoutedEventArgs e) | Обробник події натиснення на кнопку «видалити». Видаляє відповідний рядок з таблиці. |
| void BtnStat_Click(object sender, RoutedEventArgs e) | Запуск вікна статистики після натиснення на кнопку «статистика». |
| void ServicesBtn_Click(object sender, RoutedEventArgs e) | Обробник події натиснення на кнопку редагування списку послуг. |

Таблиця 3.4 – Опис методів і змінних класу вікна статистики

| **Метод** | **Призначення** |
|---|---|
| DataTable dt | Об’єкт, який зберігає в собі таблицю з основного вікна. |
| DateTime startDate | Початкова дата періоду, який обрав користувач. |
| DateTime endDate | Початкова дата періоду, який обрав користувач. |
| DataTable usersStat | Таблиця, яка зберігає статистику по всім користувачам. |
| int unDoneCount | Кількість невиконаних замовлень. |
| int proceeds | Загальний прибуток. |
| int periodCount | Прибуток за період. |
| Statistics (DataTable dt, User u) | Конструктор, який завантажує вікно та ініціалізує початкові дані. |
| void updateDates() | Метод, який оновлює дані у вікні. |
| void fillStatistics() | Метод, який заповнює статистику. |
| void fillusersStatistics() | Заповнює статистику щодо користувачів. |

Продовження табл. 3.4

| | |
|---|---|
| void startDataChanged(object sender, RoutedEventArgs e) | Обробник події зміни початкової дати. |
| void BackBtn_Click(object sender, RoutedEventArgs e) | Логіка повернення користувача на попереднє вікно. |
| void endDataChanged(object sender, RoutedEventArgs e) | Обробник події зміни кінцевої дати. |
| void Button_Click(object sender, RoutedEventArgs e) | Формування звіту в файлі при натисненні кнопки «створити звіт». |

Таблиця 3.5 – Опис методів і змінних класу вікна списку деталей

| **Метод** | **Призначення** |
|---|---|
| class Row | Клас, який описує один рядок в таблиці деталей. |
| DataTable ItemsList | Список деталей, які відображаються в таблиці. |
| int selectedRow | Індекс обраного рядка в таблиці. |
| Window1 w | Посилання на основне вікно. |
| User u | Інформація щодо користувача. |
| PartsList(int selectedRow, Window1 w, User u) | Конструктор, який задає початкові значення і створює вікно. |
| void Window_Loaded(object sender, RoutedEventArgs e) | Обробник події завантаження вікна. |
| void updateList() | Оновлення списку деталей. |
| void acceptBtn_Click(object sender, RoutedEventArgs e) | Обробник події для кнопки, яка змінює список запчастин для замовлення. |
| void cancelBtn_Click(object sender, RoutedEventArgs e) | Повернення на основне вікно. |

Продовження табл. 3.5

| | |
|---|---|
| void addOneBtn_Click(object sender, RoutedEventArgs e) | Додавання нової деталі в базу. |
| void deleteBtn_Click(object sender, RoutedEventArgs e) | Видалення деталі з бази. |

Таблиця 3.6 – Опис методів і змінних класу вікна списку послуг

| **Метод** | **Призначення** |
|---|---|
| DataTable ServicesList_ | Список послуг, які відображаються в таблиці. |
| int selectedRow | Індекс обраного рядка в таблиці. |
| Window1 w | Посилання на основне вікно. |
| User u | Інформація щодо користувача. |
| ServicesList(int selectedRow, Window1 w, User u) | Конструктор, який задає початкові значення і створює вікно. |
| void Window_Loaded(object sender, RoutedEventArgs e) | Обробник події завантаження вікна. |
| void updateList() | Оновлення списку послуг. |
| void acceptBtn_Click(object sender, RoutedEventArgs e) | Обробник події для кнопки, яка змінює список послуг для замовлення. |
| void cancelBtn_Click(object sender, RoutedEventArgs e) | Повернення на основне вікно. |
| void addOneBtn_Click(object sender, RoutedEventArgs e) | Логіка додавання нової послуги в базу. |
| void deleteBtn_Click(object sender, RoutedEventArgs e) | Логіка видалення послуги з бази. |

### 3.3 Використання програмного додатку

Після запуску додатку через файл з розширенням “.exe” бачимо форму авторизації (рис. 3.26):

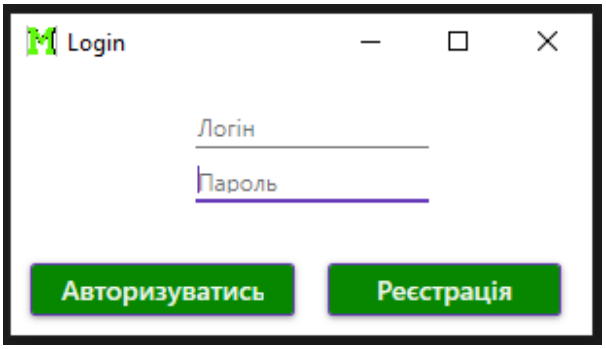



Рисунок 3.26 – Вигляд першого вікна програми

Для авторизації необхідно ввести свій логін та пароль в додатку та натиснути кнопку «авторизуватись». Якщо ж користувач не зареєстрований – потрібно перейти до вікна реєстрації, натиснувши відповідну кнопку. У випадку, якщо неправильно ввести дані для входу – отримаємо повідомлення (рис. 3.27).

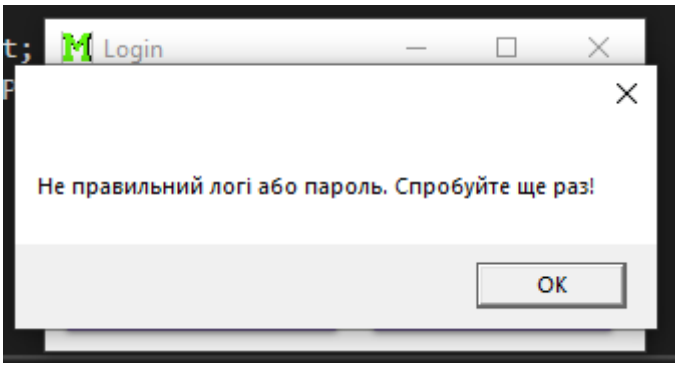



Рисунок 3.27 – Помилка при авторизації

Під час реєстрації необхідно заповнити необхідні поля (рис. 3.28). У випадку, якщо якесь поле не заповнено, паролі не співпадають або логін, введений користувачем вже знайдено в базі – користувач отримує наступне повідомлення (рис. 3.29).

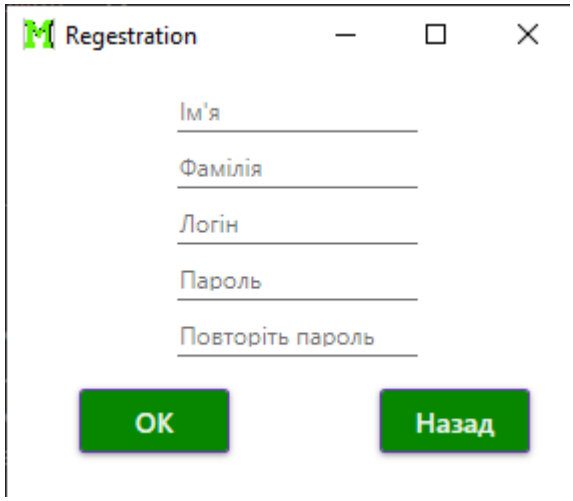



Рисунок 3.28 – Поля, які необхідно заповнити для реєстрації

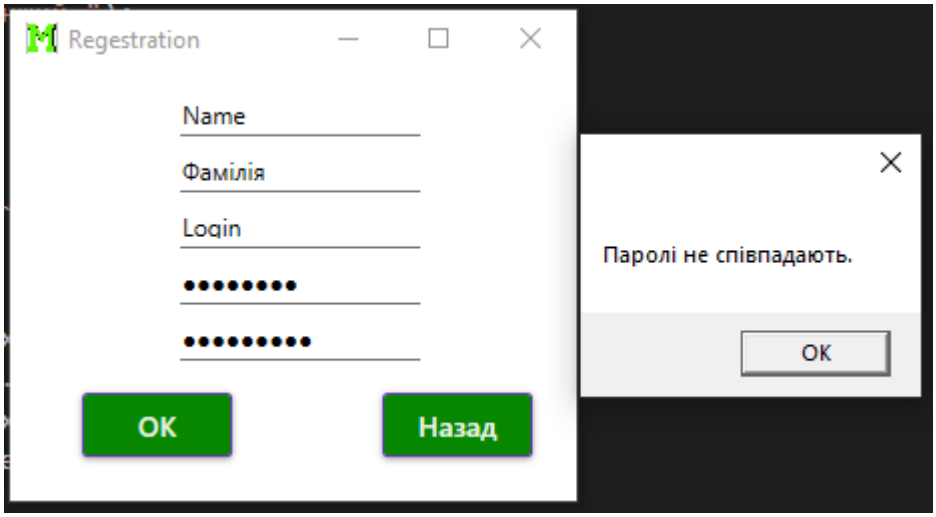



Рисунок 3.29 – Приклад помилки при реєстрації

У випадку успішної реєстрації, користувач отримує відповідне повідомлення (рис. 3.30):

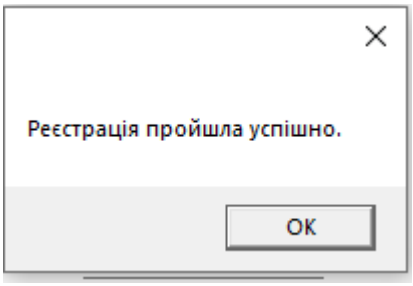



Рисунок 3.30 – Успішна реєстрація

Після успішної реєстрації дане вікно закривається і відкривається вікно авторизації, де потрібно ввести свої дані. Якщо дані для входу введені правильно – ми потрапляємо до основного вікна (рис. 3.31):

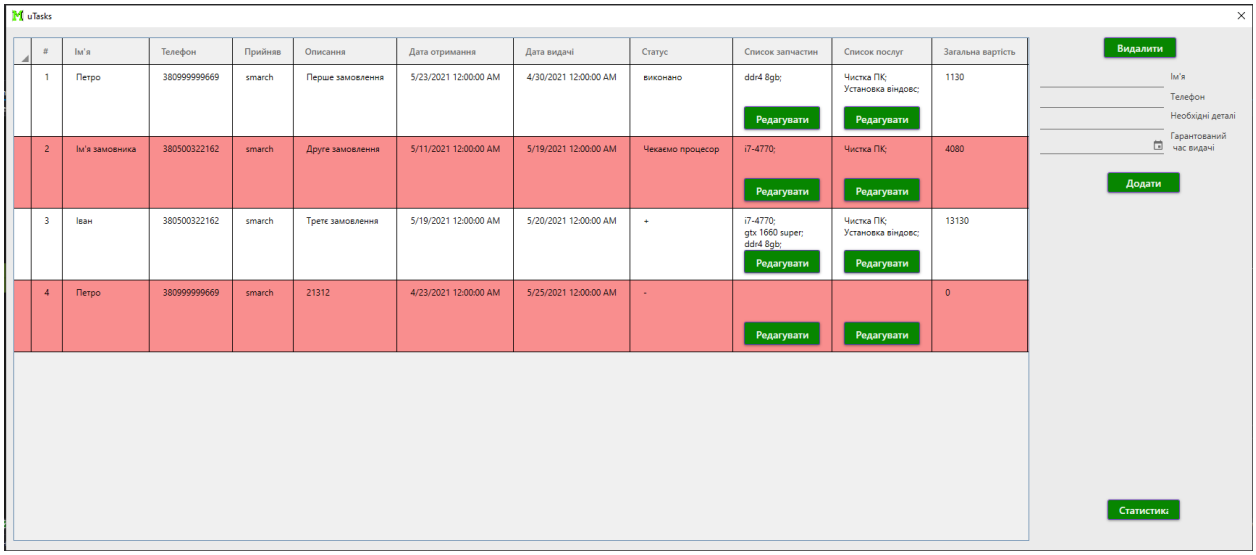



Рисунок 3.31 – Основне вікно програми

Функції основних елементів головного вікна програми:

- таблиця відображає всі замовлення на ремонт техніки та необхідну інформацію. При двійному натисненні на полях доступне редагування цих полів, при цьому деякі колонки захищені від копіювання. Для

редагування колонок «Список запчастин» та «Список послуг» необхідно натиснути на кнопку «Редагувати», після чого буде відкрите вікно редагування списків запчастин або послуг (про більше детально – далі),

- кнопка «Видалити» - видаляє обраний рядок в таблиці,
- кнопка «Додати» - додає до таблиці нове замовлення, данні щодо якого користувач повинен попередньо ввести в полях, які розташовані в полях над цим елементом. У випадку, якщо не всі поля заповнені – користувач отримує повідомлення про помилку (рис. 3.32),
- кнопка «статистика» - доступна лише для адміністратора, відкриває нове вікно статистики.

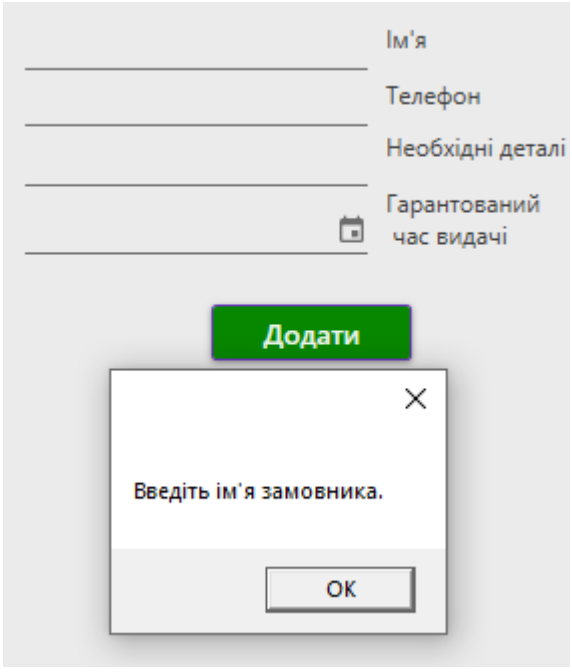



Рисунок 3.32 – Повідомлення про помилку при додаванні нового замовлення

Якщо натиснути на кнопку «Редагувати» будь-якого із списків – відкривається вікно редагування цього списку. На ньому доступний список всіх найменувань деталей або послуг з можливістю додати нове найменування (рис. 3.33):

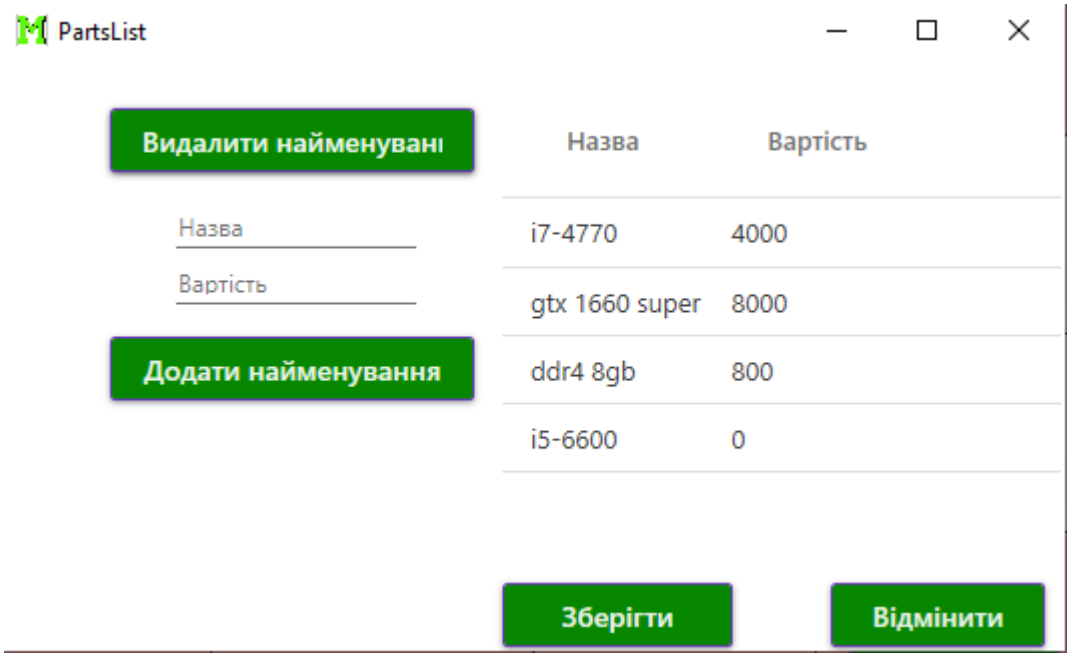



Рисунок 3.33 – Приклад вікна редагування списку

Якщо користувач є адміністратором і натискає на кнопку «Статистика» - то він потрапляє на вікно статистики (рис. 3.34):

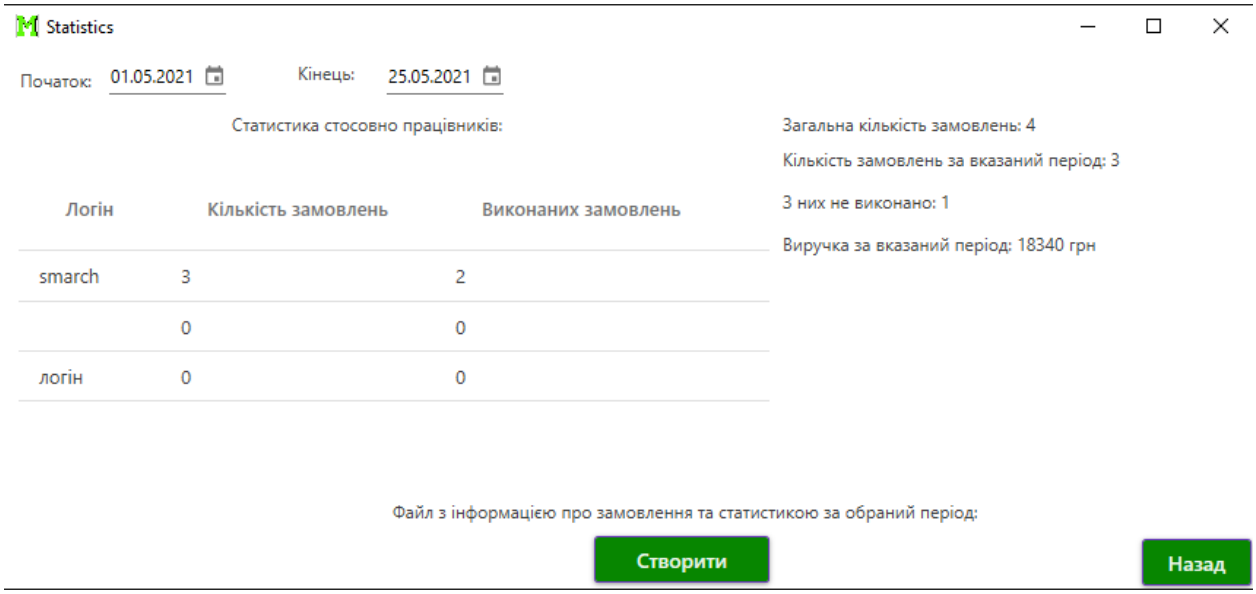



Рисунок 3.34 – Вікно статистики

На ньому можна обрати період, за яким буде відображено статистику. Обрати дати можна за допомогою елементів обрання числа, але за замовчуванням встановлюється період починаючи з початку поточного місяця, а закінчується поточною датою відкриття сторінки.

На даній сторінці присутня таблиця статистики майстрів сервісу (загальна кількість прийнятих замовлень майстром за період і кількість виконаних замовлень), а також текстова інформації щодо діяльності сервісу в цілому. Присутня кнопка «Назад», яка повертає користувача на попередню сторінку.

Також, маємо доступ до кнопки «створити», яка створює звіт та зберігає його у вигляді PDF-файлу на робочому столі. Звіт містить інформації із таблиці на основному вікні, але зберігається інформація щодо замовлень лише за вказаний період, який обирається у вікні статистики. Файл має наступний вигляд (рис. 3.35):

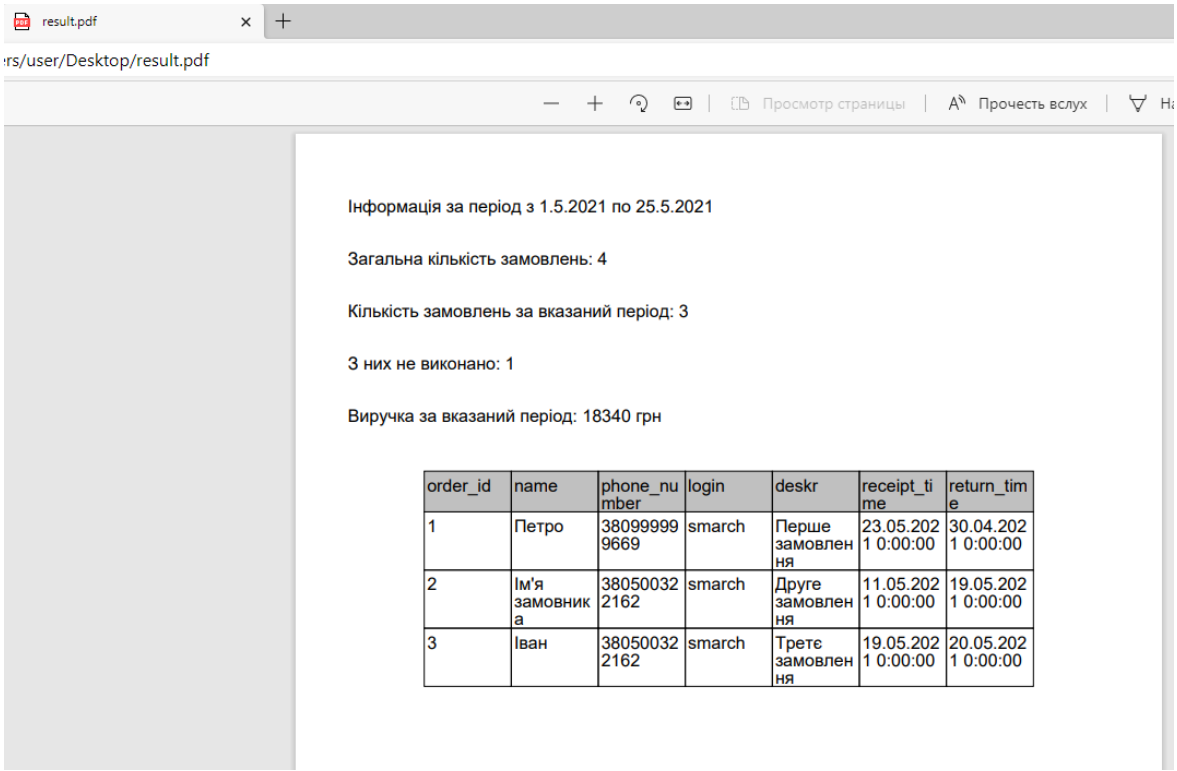



Рисунок 3.35 – Приклад звіту

# ВИСНОВКИ

Під час карантину значно збільшилась потреба в комп'ютерному обладнанні, що, в свою чергу, значно зріс попит на послуги комп'ютерних сервісів та магазинів. Через величезну кількість замовлень працівники сервісів дуже часто починають плутатись в замовленнях, що викликає незручності як для працівників, так і для клієнтів. Ситуацію може змінити програмний додаток, який буде систематизувати та зберігати інформацію про замовлення. Запит на розробку такого програмного додатку надійшов від власникау сервісного центру «Мелт».

За результатами обговорення питання з замовником було визначено основні вимоги до проекту, складено детальне технічне завдання і терміни виконання проекту.

Для збереження даних було використано СУБПД MySql, під керівництвом якої було реалізовано розроблену модель бази даних. Реалізація десктопного додатку виконано на основі WPF-проекту. Логіка програми описана за допомогою мови програмування C#. База даних розміщена на віддаленому сервері, а для додатку було створено інсталятор.

Розроблений додаток протестовано, тож він готовий до використання. Під час роботи в ньому можна виконувати реєстрацію, авторизацію, заносити дані про замовлення до таблиці, редагувати попередні замовлення, отримувати чітке візуальне відображення стадії виконання замовлення. Також, можна редагувати списки деталей та послуг. Адміністратор має змогу переглядати статистику за певний період та зберігати звіт у вигляді PDF-файлу.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Руководство по WPF [Електронний ресурс] – Режим доступу: https://metanit.com/sharp/wpf/
2. Руководство по MySQL [Електронний ресурс] – Режим доступу: https://metanit.com/sql/mysql/
3. Microsoft ToDo[Електронний ресурс] – Режим доступу: https://todo.microsoft.com/tasks/
4. Рем онлайн [Електронний ресурс] – Режим доступу: https://remonline.ua/ru/
5. IMA 2021 [Електронний ресурс] – Режим доступу: https://ksu.sumdu.edu.ua/index.php/ua/pro-ksu/novyny/5-testovaya-novost-4-vidbuvsya-konkurs-studentskikh-naukovikh-robit-za-napryamom-fizika-ta-astronomiya
6. Контроль як функція управління сучасного підприємства [Електронний ресурс] – Режим доступу: https://ru.osvita.ua/vnz/reports/management/14696/
7. Order Management – Definition, Process & Cycle [Електронний ресурс] – Режим доступу: https://www.zoho.com/inventory/articles/basic-guide-to-order-management.html
8. Microsoft Excel [Електронний ресурс] – Режим доступу: https://www.tadviser.ru/index.php/%D0%9F%D1%80%D0%BE%D0%B4%D1%83%D0%BA%D1%82:Microsoft_Excel
9. Google Tables: How I Use Google’s New Workflow Tool [Електронний ресурс] – Режим доступу: https://www.benlcollins.com/tables/google-tables/
10. Create IDF0 Diagrams [Електронний ресурс] – Режим доступу: https://support.microsoft.com/en-us/office/create-idef0-diagrams-ea7a9289-96e0-4df8-bb26-a62ea86417fc

11. Логічне проектування бази даних [Електронний ресурс] – Режим доступу: http://refleader.ru/ujgbewotryfs.html
12. UML [Електронний ресурс] – Режим доступу: https://habr.com/ru/post/511798/
13. Open Server [Електронний ресурс] – Режим доступу: https://ospanel.io/docs/
14. Windows Presentation Foundation documentation [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/dotnet/desktop/wpf/?view=netdesktop-5.0
15. Google Materio Design [Електронний ресурс] – Режим доступу: https://material.io/design

# ДОДАТОК А

## Технічне завдання

## ТЕХНІЧНЕ ЗАВДАННЯ

**на розробку інформаційної системи**
**«Програмний додаток підтримки діяльності компанії по ремонту комп'ютерної техніки»**

**ПОГОДЖЕНО:**

Доцент кафедри комп’ютерних наук

_____ Ващенко С.М.

Студент групи ІТ-71

_____ Проценко М.О.

**Суми 2021**

# 1 Призначення й мета створення додатку

## 1.1 Призначення додатку

Інформаційна система має допомагати майстрам сервісного центру «Мелт» систематизувати всі замовлення, отримувати всю необхідну інформацію щодо замовлень максимально швидко та в будь-який момент редагувати їх.

## 1.2 Мета створення

Автоматизація процесів збору, збереження та обробки інформації щодо замовлень, які виконує компанія  по ремонту комп'ютерної техніки.

## 1.3 Цільова аудиторія

До цільової аудиторії можна віднести майстрів та директора центру.

# 2 Вимоги до додатку

## 2.1 Вимоги до додатку в цілому

### 2.1.1 Вимоги до структури й функціонування додатку

Додаток має бути доступним для інсталювання шляхом передачі файлів для інсталяції на фізичному носію, або через мережу Інтернет, та мати дві складові частини: безпосередньо десктопний додаток  із зрозумілим візуальним інтерфейсом та хостингу з базою даних, на якому зберігатиметься інформація щодо замовлень. Додаток має являти собою графічну програму, виконану на основі мови програмування C#, за допомогою графічної системи WPF.

### 2.1.2 Вимоги до персоналу

Не вимагається від користувачів системи особливих навичок роботи, достатньо базового рівня загальних навичок роботи з персональним комп'ютером.

### 2.1.3 Вимоги до збереження інформації

Уся інформація надана у додатку буде зберігатися у базі даних, реалізованій засобами системи управління базами даних MySQL.

### 2.1.4 Вимоги до розмежування доступу

Доступ до програми повинен надаватись за двома режимами доступу: працівник (працівник сервісу) та адміністратор (власник сервісу). Повинна бути присутня система авторизації та реєстрації для нових користувачів.

## 2.2 Структура додатку

### 2.2.1 Загальна інформація про структуру додатку

Структура додатку являє собою набір вікон:

- Головне вікно, на якому міститься таблиця зі всіма замовленнями та інформацією щодо них, а також – кнопки, за допомогою яких відбувається доступ до редагування, додавання та видалення замовлень (при натисканні на кнопки вікно збільшується та з’являються необхідні для роботи елементи).
- Вікно авторизації.
- Вікно реєстрації.
- Вікно перегляду статистики з ремонту по певним періодам часу.
- Вікно перенесення всієї інформації щодо замовлень до PDF-файлу.

### 2.2.2 Навігація

Перехід між вікнами додатку відбувається за допомогою відповідних кнопок. З меню авторизації можна потрапити до головного вікна. В свою чергу, з головного меню можна потрапити до панелей редагування та додавання замовлень, вікна збереження інформації та статистики.

### 2.2.3 Наповнення додатку

Додаток буде побудований за допомогою таких елементів WPF:

- Label – поле для статичного тексту,
- Text Box – поле для введення даних,

- Data Grid – таблиця з інформацією
- Button – кнопка для введення даних до системи, або переходу на інше вікно.

### 2.2.4 Дизайн та структура додатку

Стиль додатку має бути простим для розуміння та виконаним в основних кольорах компанії (чорний, білий та зелений кольори).

Розташування елементів на головній сторінці додатку схематично показано на рисунку А.1.

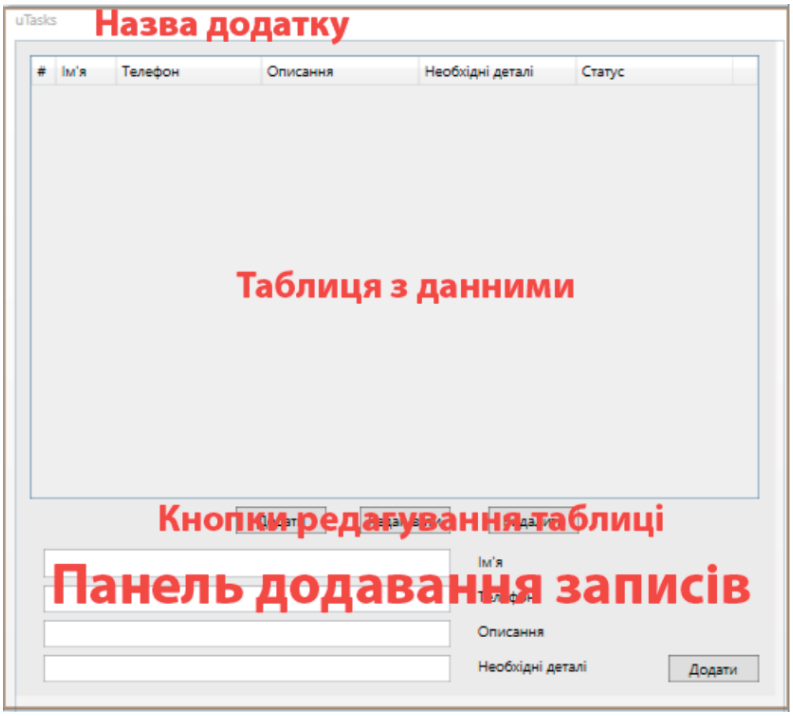



Рисунок А.1 – Схема головної сторінки

### 2.2.5 Система навігації (карта додатку)

Карта додатку зображена на рисунку А.2.

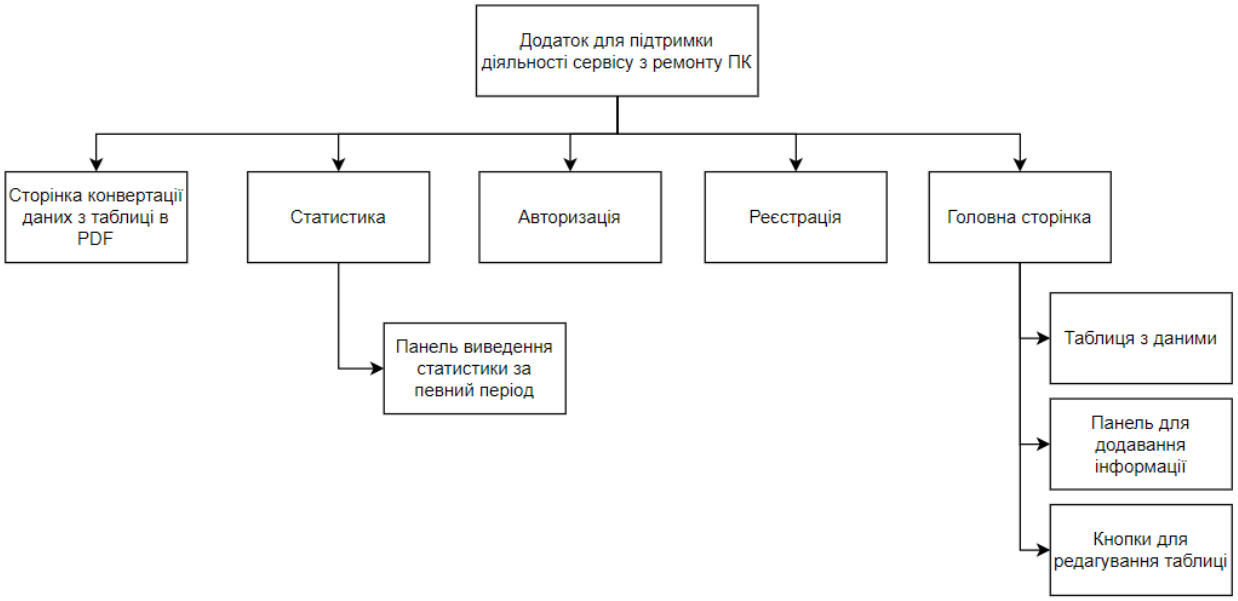



Рисунок А.2 – Карта додатку

## 2.3 Вимоги до функціонування системи

### 2.3.1 Потреби користувача

Потреби користувача, визначені на основі рішення замовника, представлені у таблиці А.1.

Таблиця А.1 – Потреби користувача

| ID | Потреби користувача | Джерело |
|---|---|---|
| UN-01 | Доступ до інформації в додатку лише за допомогою заздалегідь обговореного паролю та логіну для директора. | Директор магазину |
| UN-02 | Можливість перегляду статистики за будь-який період. | Директор магазину |
| UN-03 | Мінімальна завантаженість інтерфейсу. | Працівник |
| UN-04 | Кольорове відображення статусу виконання замовлення. | Працівник |
| UN-05 | Можливість перенесення даних з таблиці до PDF-файлу. | Директор магазину |

Продовження табл. А1

| | | |
|---|---|---|
| UN-06 | Синхронізація даних між додатком, встановленим на декількох комп'ютерах, за допомогою загальної бази даних. | Директор магазину |
| UN-07 | Можливість редагування, додавання та видалення замовлень | Працівник |

### 2.3.2 Функціональні вимоги

На основі вимог замовника були визначені такі функціональні вимоги:

- авторизація користувачів;
- форма реєстрації для нових працівників;
- пошук замовлень по номеру телефона.

### 2.3.3 Системні вимоги

Даний розділ визначає, розподіляє та вказує на системні вимоги, визначені розробником. Їх перелік наведений в таблиці А.2.

Таблиця А.2 – Системні вимоги

| ID | Системні вимоги | Пріоритет | Опис |
|---|---|---|---|
| SR-01 | Наявність елементів інтерфейсу для додавання нових записів | M | Надає можливість адміністратору додати замовлення |
| SR-02 | Список замовлень | M | Відображає всю необхідну інформацію щодо замовлень |
| SR-03 | Статистика замовлень | C | Відображає загальний прибуток за певний період та іншу інформацію, необхідну директору. |

Продовження табл. А.2

| | | | |
|---|---|---|---|
| SR-04 | База даних із замовленнями | M | Дозволяє синхронізувати дані між комп'ютерами |
| SR-05 | Панель для реєстрації нових користувачів | S | Дозволяє додати нового користувача в БД з послідуючою можливістю авторизуватись за допомогою даних нового користувача в додатку. |

Умовні позначення в таблиці А.2:

Must have (M) – вимоги, які повинні бути реалізовані в системі;

Should have (S) – вимоги, які мають бути виконані, але вони можуть почекати своєї черги;

Could have (C) – вимоги, які можуть бути реалізовані, але вони не є центральною метою проекту.

## 2.4 Вимоги до видів забезпечення

### 2.4.1 Вимоги до інформаційного забезпечення

Основна інформація, яка буде циркулювати в системі – це дані щодо замовлень (ім'я замовника, його номер телефону, короткий опис проблеми, необхідні деталі та статус виконання замовлення). Масив цих даних формується за допомогою групи елементів інтерфейсу TextBox та кнопок, які запускають процес перевірки введених даних на відповідність заданим умовам (довжина тексту, присутність лише цифр в деяких полях, обрання одного варіанту із запропонованих і тд).

Також, в додатку знаходиться інформація щодо користувача, який працює в ньому.

Вся ця інформація оновлюється за допомогою бази даних, а нова інформація – доповнює саму БД, таким чином відбувається двосторонній обмін даними між БД та додатком, а також – між користувачем та додатком (через графічний інтерфейс).

### 2.4.2 Вимоги до лінгвістичного забезпечення

Додаток має бути виконаний українською мовою.

### 2.4.3 Вимоги до програмного забезпечення

Програмне забезпечення клієнтської частини повинне задовольняти наступним вимогам:

- Версія Windows 7 або вище;
- Підключення до мережі Інтернет для синхронізації (не обов’язкове).

## 3 Склад і зміст робіт зі створення додатку

Докладний опис етапів роботи зі створення додатку наведено в таблиці А.3.

Таблиця А.3 – Етапи створення додатку

| № | Склад і зміст робіт | Строк розробки (у робочих днях) |
|---|---|---|
| 1 | Постановка цілей необхідних для досягнення необхідного результату | 1 день |
| 2 | Складання технічного завдання | 3 дні |
| 3 | Підготовка прототипу | 2 дні |
| 4 | Створення макету додатку | 2 дні |
| 5 | Створення додатку | 4 дні |
| 6 | Створення бази даних | 2 дні |
| 7 | Підключення додатку до бази даних | 1 день |
| 8 | Перевірка працездатності додатку | 1 день |
| 9 | Завершення роботи | 1 день |
| 10 | Оформлення документації щодо додатку | 3 дні |
| | Загальна тривалість робіт | 20 днів |

## 4 Вимоги до складу й змісту робіт із введення додатку в експлуатацію

Для того, щоб інформація між додатками синхронізувалась, необхідно розмістити базу даних на хостингу. При цьому, необхідно змінити запити для підключення до БД, для того щоб виконати передачу даних коректно.

# ДОДАТОК Б

## Планування робіт

**SMART-метод деталізації мети проекту.** Метою проекту є розробка програми для полегшення процесу систематизації замовлень, залишаючи на стікері лише номер замовлення, а решту інформації переносити в програмний додаток, побудований на WPF-проекті мови C#. Результат деталізації наведено в табл. Б.1.

Таблиця Б.1 − Деталізація мети методом SMART

Таблиця Б.1 – Деталізація мети методом SMART

| | |
|---|---|
| Specific | Розробити програмний продукт, який буде являти собою список замовлень та інформацію щодо них. |
| Measurable | Програма розробляється як дипломний проект, тому витрати на створення можна вважати нульовими. |
| Achievable | Програмний продукт повинен вирішувати проблеми саме упорядкування та відстеження замовлень щодо ремонту комп'ютерної техніки. Необхідно реалізувати систему входу, синхронізацію між декількома комп’ютерами, а також – програма повинна працювати за умов відсутності інтернет-зв’язку. |
| Relevant | Щодо виконання проекту не існує ніяких проблем. Він буде створений на основі WPF-проекту на мові програмування C# і зв’язаний з базою даних SQL. |
| Time-framed | Продукт повинен бути реалізований до 23.05.2021 року. |

**Планування змісту структури робіт.** Для змістовного подання обсягу необхідних робіт щодо проекту дуже владо служить WBS діаграма. Вона

максимально вдало показує взаємодію між пакетами робіт, необхідних для реалізації проекту. Це робиться для визначення послідовності виконання етапів проектування додатку.

На основі послідовності створення і впровадження проекту було створено WBS-структуру (рис. Б.1.).

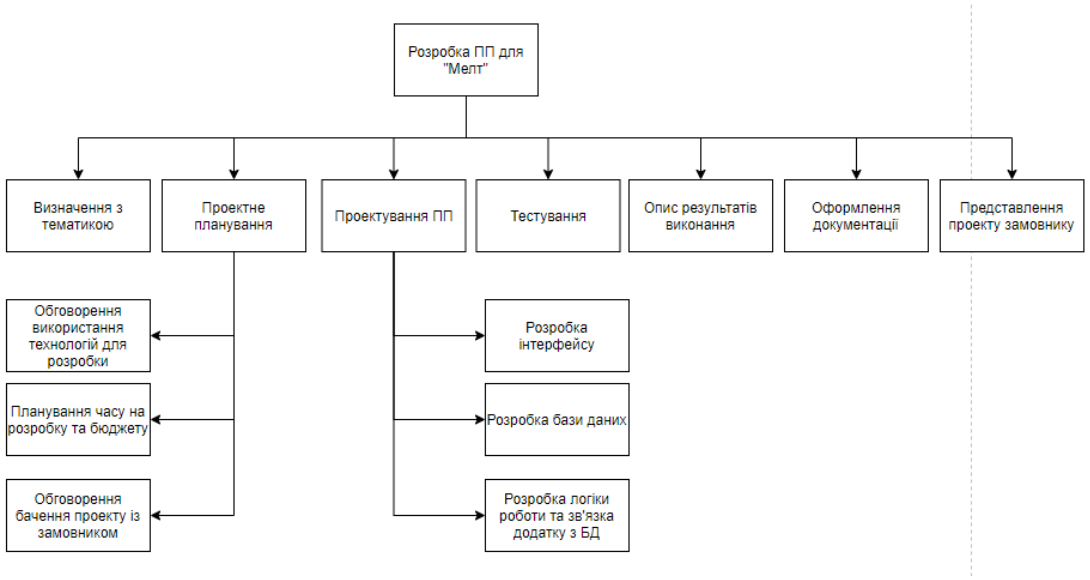



Рисунок Б.1 – WBS-структура проекту

**Планування структури організації.** Перед початком створення додатку, звичайно ж, потрібно визначитись з тим, хто цей проект буде реалізовувати. Для цього звичайно і створюються OBS діаграми. Після проведення планування структури організації маємо наступну таблицю (табл. Б.2) та схему (рис. Б.2).

Таблиця Б.2 – Виконавці проекту

| Роль | Ім’я | Проектна роль |
|---|---|---|
| Програміст | Проценко М.О. | Виконує розробку основного функціоналу проекту та виконує тестування. |
| Дипломний керівник | Ващенко С.М. | Відповідає за виконання термінів, виконує збір та аналіз даних. |

Продовження табл. Б.2

| | | |
|---|---|---|
| Організаційний менеджер | Проценко М.О. | Відповідає за створення документації і планів на виконання. |
| Тестувальник | Марченко О.С., Проценко М.О. | Виконує пошук помилок в додатку і перевіряє його на відповідність до ТЗ. |

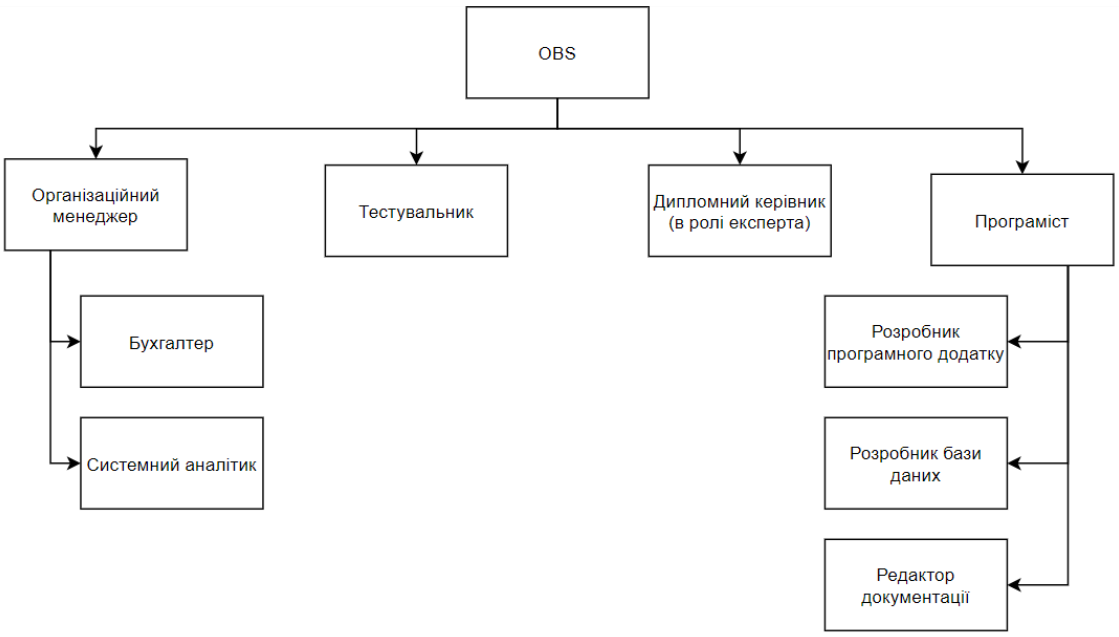



Рисунок Б.2 – OBS-структура проекту

**Діаграма Ганта.** В моєму випадку, календарний графік виконання IT-проекту представлений в формі діаграми Ганта (рис. Б.3, Б.4). Зроблено це через те, що, на мою думку, діаграма Ганта найбільш вдало відображає порядок створення проекту. До того ж, ця модель найбільш поширена. Діаграму створено за допомогою Microsoft Project.

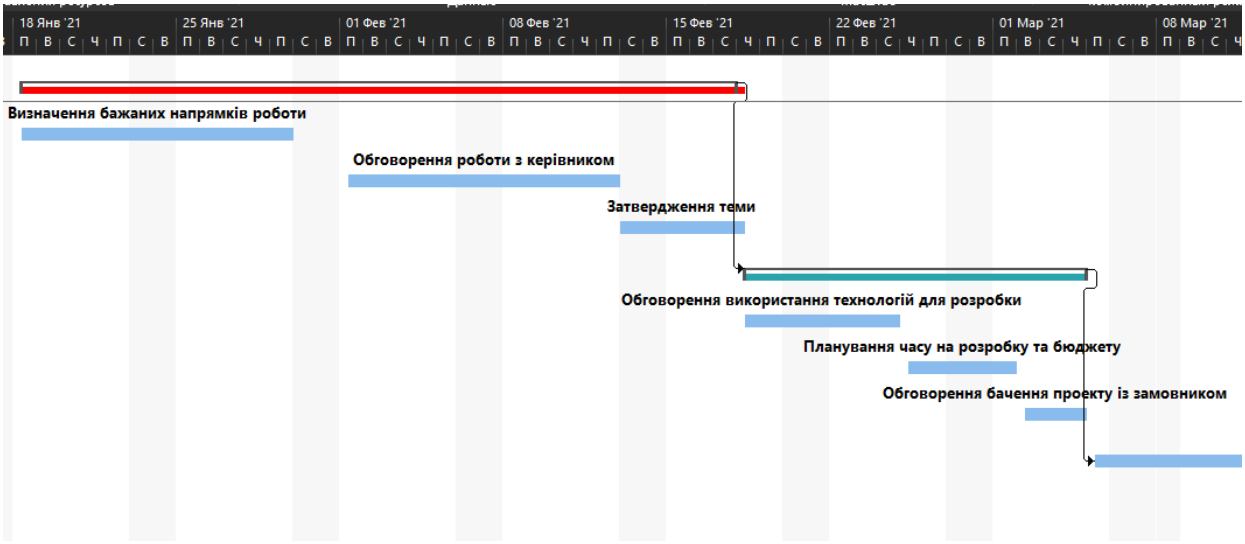



Рисунок Б.3 – Діаграма Ганта проекту

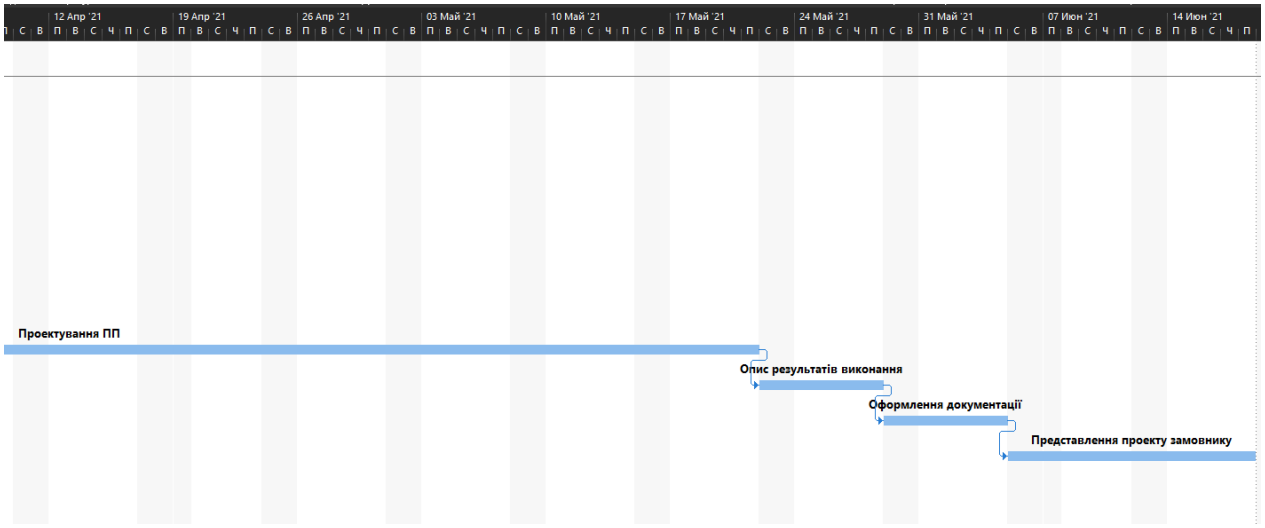



Рисунок Б.4 – Продовження діаграми Ганта

Більш розгорнута модель мереж (рис. Б.5, Б.6, Б.7).

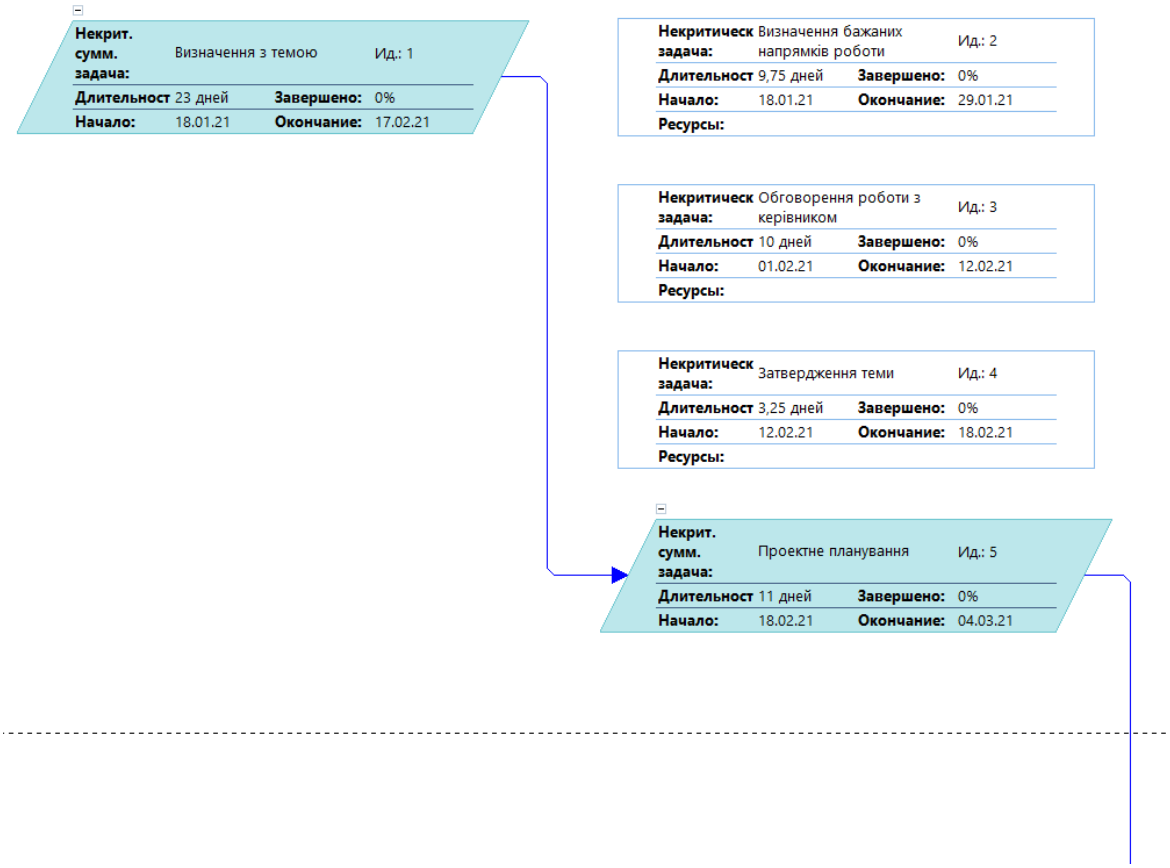



Рисунок Б.5 – Модель мереж

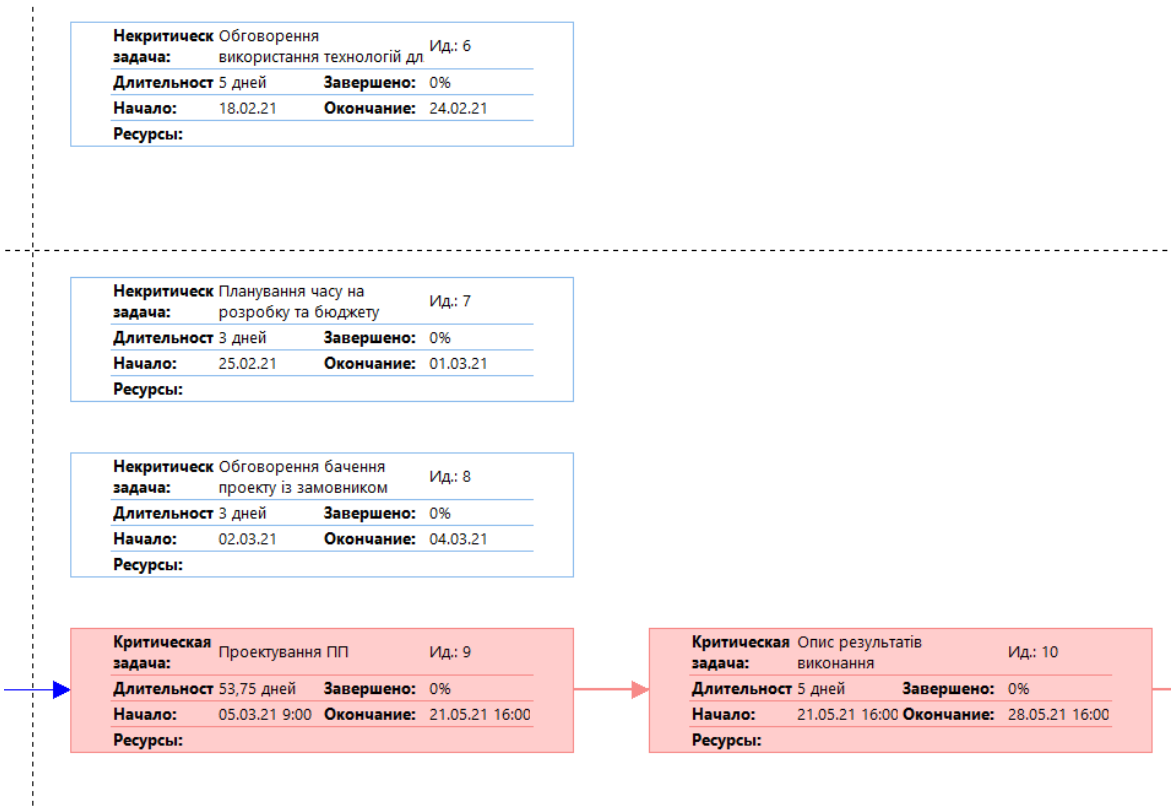



Рисунок Б.6 – Продовження моделі мереж

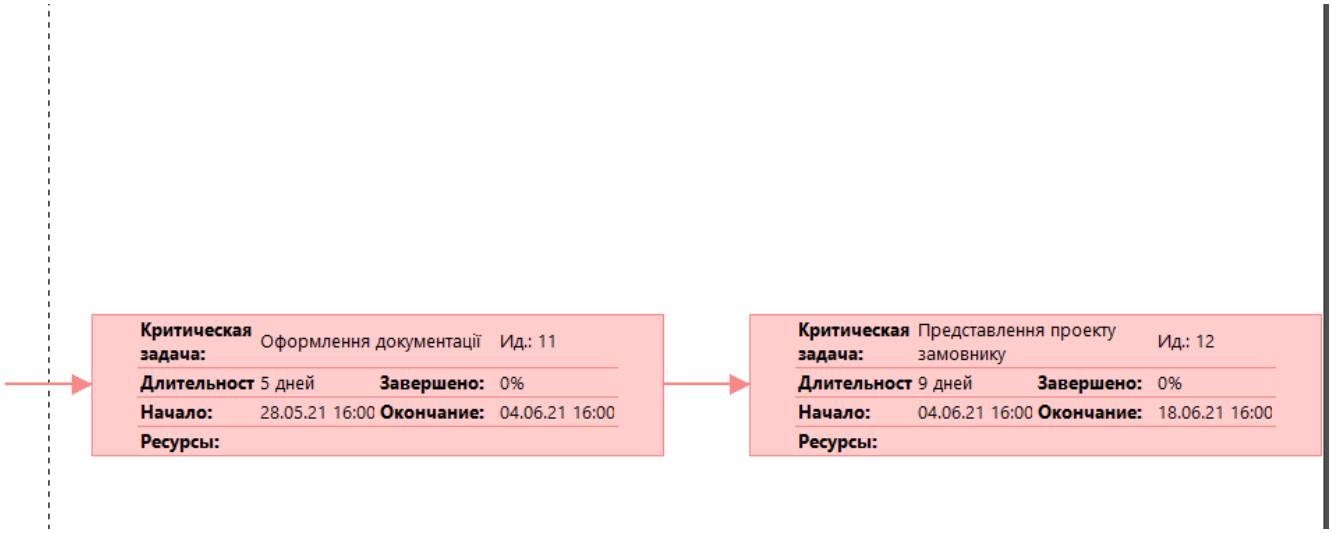



Рисунок Б.7 – Продовження моделі мереж

**Аналіз ризиків.** На основі класифікації ризиків було створено таблицю (табл. Б.3), яка відображає можливість виникнення того, чи іншого. На її основі будуємо матрицю ризиків(рис. Б.8).

Таблиця Б.3 – Класифікація ризиків

| Назва ризику | Ймовірність | Величина втрат |
|---|---|---|
| Погана продуманість ідеї | 1 | 4 |
| Недотримання дедлайнів | 1 | 2 |
| Неспроможність програміста реалізувати ті, чи інші функції | 2 | 4 |
| Поломка обладнання | 1 | 3 |
| Некоректне тестування | 2 | 1 |

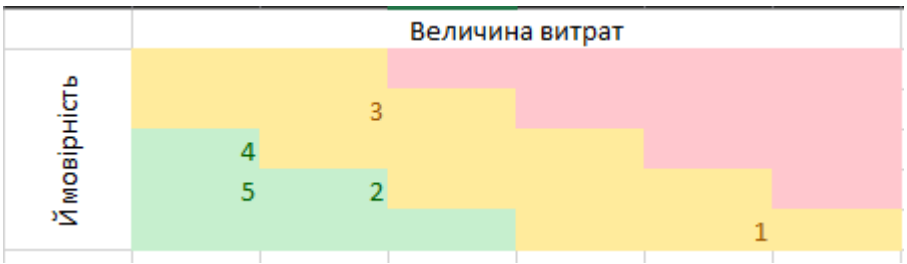



Рисунок Б.8 – Матриця ризиків

На основі даних з матриці можемо продумати план дій при виникненні ризиків (табл. Б.4).

Таблиця Б.4 – Відповідні дії при виникненні ризиків

| Ризики проекту | Реакція на ризик |
|---|---|
| Погана продуманість ідеї | Продумати ідею більш ґрунтовно та детально. Поспілкуватись з замовником та дипломним керівником. Ще раз проглянути схожі додатки, і взяти звідти краще, додавши своє. |
| Недотримання дедлайнів | Підвищення швидкості виконання завдань, або (і) перенесення термінів зі зміною календарного плану. |
| Неспроможність програміста реалізувати ті, чи інші функції | Спілкування з дипломним керівником, або іншими студентами , які можуть допомогти (можливо, різноманітні форуми, зібрання і т.д.). |
| Поломка обладнання | Якнайшвидше виконати ремонт обладнання, або перенести проект на інший пристрій і продовжити роботу. |
| Некоректне тестування | Оскільки тестувальником може виступати лише програміст і замовник, то у випадку некоректного тестування з боку програміста необхідно в короткі терміни вивчити необхідну інформацію, яка зможе пришвидшити процес. |

# ДОДАТОК В

## Лістинг коду

### Скрипт створення бази даних

```
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
START TRANSACTION;
SET time_zone = "+00:00";
/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;
--
-- База данных: `utasks`
--
-- --------------------------------------------------------
--
-- Структура таблицы `customers`
--
CREATE TABLE `customers` (
  `customer_id` int NOT NULL,
  `name` varchar(40) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL,
  `phone_number` varchar(12) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
--
-- Дамп данных таблицы `customers`
--
INSERT INTO `customers` (`customer_id`, `name`, `phone_number`) VALUES
(1, 'Олег Степанов', '380500322162'),
(2, 'Степан Олегович', '380505032343'),
(3, 'Петро', '380999999669'),
(4, 'Іван', '380500322162'),
(5, '1', '1'),
(6, '2', '2'),
(7, '3', '3'),
(8, '4', '4'),
(9, '5', '5'),
(10, '6', '6'),
(11, '7', '7'),
(12, 'Ім\'я', 'Телефон');
-- --------------------------------------------------------
--
-- Структура таблицы `orders`
--

CREATE TABLE `orders` (
  `order_id` int NOT NULL,
  `deskr` varchar(100) NOT NULL,
  `status` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL,
  `receipt_time` date NOT NULL,
  `return_time` date NOT NULL,
  `user_id` int NOT NULL,
  `customer_id` int NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
--
-- Дамп данных таблицы `orders`
--
INSERT INTO `orders` (`order_id`, `deskr`, `status`, `receipt_time`, `return_time`,
`user_id`, `customer_id`) VALUES
(1, 'Перше замовлення', 'виконано', '2021-05-23', '2021-04-30', 1, 3),
(2, 'Друге замовлення', 'Чекаємо процесор', '2021-05-11', '2021-05-19', 1, 1),
(3, 'Третє замовлення', '+', '2021-05-19', '2021-05-20', 1, 4),
```

```
(4, '21312', '+', '2021-04-23', '2021-05-25', 1, 3),
(9, '1', 'Тільки прийнято', '2021-05-26', '2021-05-27', 3, 5);
-- --------------------------------------------------------
--
-- Структура таблицы `parts`
--
CREATE TABLE `parts` (
  `part_id` int NOT NULL,
  `name` varchar(50) NOT NULL,
  `price` int NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

--
-- Дамп данных таблицы `parts`
--
INSERT INTO `parts` (`part_id`, `name`, `price`) VALUES
(1, 'i7-4770', 4000),
(2, 'gtx 1660 super', 8000),
(3, 'ddr4 8gb', 800),
(4, 'i5-6600', 0);
-- --------------------------------------------------------
--
-- Структура таблицы `parts_list`
--
CREATE TABLE `parts_list` (
  `part_list_id` int NOT NULL,
  `part_id` int NOT NULL,
  `order_id` int NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
--
-- Дамп данных таблицы `parts_list`
--
INSERT INTO `parts_list` (`part_list_id`, `part_id`, `order_id`) VALUES
(47, 1, 2),
(50, 3, 3),
(51, 1, 3),
(52, 3, 4),
(53, 1, 4),
(54, 2, 4),
(57, 4, 1),
(58, 1, 1),
(59, 2, 1),
(60, 3, 1);
-- --------------------------------------------------------
--
-- Структура таблицы `services`
--
CREATE TABLE `services` (
  `service_id` int NOT NULL,
  `name` varchar(50) NOT NULL,
  `price` int NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
--
-- Дамп данных таблицы `services`
--
INSERT INTO `services` (`service_id`, `name`, `price`) VALUES
(1, 'Установка віндовс', 250),
(2, 'Чистка ПК', 80);
-- --------------------------------------------------------
--
-- Структура таблицы `services_list`
--
CREATE TABLE `services_list` (
  `services_list_id` int NOT NULL,
  `service_id` int NOT NULL,
  `order_id` int NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
--
-- Дамп данных таблицы `services_list`
--
```

```
INSERT INTO `services_list` (`services_list_id`, `service_id`, `order_id`) VALUES
(3, 2, 3),
(4, 1, 3),
(5, 2, 1),
(6, 1, 1),
(9, 2, 2),
(10, 1, 4);
-- --------------------------------------------------------
--
-- Структура таблицы `users`
--
CREATE TABLE `users` (
  `user_id` int NOT NULL,
  `login` varchar(20) NOT NULL,
  `password` varchar(20) NOT NULL,
  `is_admin` tinyint(1) NOT NULL DEFAULT '0',
  `fname` varchar(20) NOT NULL,
  `lname` varchar(20) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
--
-- Дамп данных таблицы `users`
--

INSERT INTO `users` (`user_id`, `login`, `password`, `is_admin`, `fname`, `lname`) VALUES
(1, 'smarch', '1324', 1, 'Olexandr', 'Marchenko'),
(3, 'test', '', 1, '', ''),
(10, 'логін', 'пароль', 0, 'ім\'я', 'фамілія');

--
-- Индексы сохранённых таблиц
--
--
-- Индексы таблицы `customers`
--
ALTER TABLE `customers`
  ADD PRIMARY KEY (`customer_id`);
--
-- Индексы таблицы `orders`
--
ALTER TABLE `orders`
  ADD PRIMARY KEY (`order_id`),
  ADD KEY `customer_id` (`customer_id`),
  ADD KEY `user_id` (`user_id`);
--
-- Индексы таблицы `parts`
--
ALTER TABLE `parts`
  ADD PRIMARY KEY (`part_id`);
--
-- Индексы таблицы `parts_list`
--
ALTER TABLE `parts_list`
  ADD PRIMARY KEY (`part_list_id`),
  ADD KEY `order_id` (`order_id`),
  ADD KEY `part_id` (`part_id`);
--
-- Индексы таблицы `services`
--
ALTER TABLE `services`
  ADD PRIMARY KEY (`service_id`);
--
-- Индексы таблицы `services_list`
--
ALTER TABLE `services_list`
  ADD PRIMARY KEY (`services_list_id`),
  ADD KEY `order_id` (`order_id`),
  ADD KEY `service_id` (`service_id`);
--
-- Индексы таблицы `users`
--
```

```
ALTER TABLE `users`
  ADD UNIQUE KEY `id` (`user_id`);
--
-- AUTO_INCREMENT для сохранённых таблиц
--
--
-- AUTO_INCREMENT для таблицы `customers`
--
ALTER TABLE `customers`
  MODIFY `customer_id` int NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=13;

--
-- AUTO_INCREMENT для таблицы `orders`
--
ALTER TABLE `orders`
  MODIFY `order_id` int NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=10;
--
-- AUTO_INCREMENT для таблицы `parts`
--
ALTER TABLE `parts`
  MODIFY `part_id` int NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=7;
--
-- AUTO_INCREMENT для таблицы `parts_list`
--
ALTER TABLE `parts_list`
  MODIFY `part_list_id` int NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=61;
--
-- AUTO_INCREMENT для таблицы `services`
--
ALTER TABLE `services`
  MODIFY `service_id` int NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;
--
-- AUTO_INCREMENT для таблицы `services_list`
--
ALTER TABLE `services_list`
  MODIFY `services_list_id` int NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=11;

--
-- AUTO_INCREMENT для таблицы `users`
--
ALTER TABLE `users`
  MODIFY `user_id` int NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=11;
--
-- Ограничения внешнего ключа сохраненных таблиц
--
--
-- Ограничения внешнего ключа таблицы `orders`
--
ALTER TABLE `orders`
  ADD CONSTRAINT `orders_ibfk_1` FOREIGN KEY (`customer_id`) REFERENCES `customers`
(`customer_id`),
  ADD CONSTRAINT `orders_ibfk_3` FOREIGN KEY (`user_id`) REFERENCES `users` (`user_id`);
--
-- Ограничения внешнего ключа таблицы `parts_list`
--
ALTER TABLE `parts_list`
  ADD CONSTRAINT `parts_list_ibfk_1` FOREIGN KEY (`order_id`) REFERENCES `orders`
(`order_id`),
  ADD CONSTRAINT `parts_list_ibfk_2` FOREIGN KEY (`part_id`) REFERENCES `parts` (`part_id`);
--
-- Ограничения внешнего ключа таблицы `services_list`
--
ALTER TABLE `services_list`
  ADD CONSTRAINT `services_list_ibfk_1` FOREIGN KEY (`order_id`) REFERENCES `orders`
(`order_id`),
  ADD CONSTRAINT `services_list_ibfk_2` FOREIGN KEY (`service_id`) REFERENCES `services`
(`service_id`);
COMMIT;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
```

```
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

## App.xaml

```
/<Application x:Class="uTasks.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             StartupUri="Login.xaml">
    <Application.Resources>
        <ResourceDictionary>
            <ResourceDictionary.MergedDictionaries>
                <ResourceDictionary
Source="pack://application:,,,/MaterialDesignThemes.Wpf;component/Themes/MaterialDesignTheme
.Light.xaml" />
                <ResourceDictionary
Source="pack://application:,,,/MaterialDesignThemes.Wpf;component/Themes/MaterialDesignTheme
.Defaults.xaml" />
                <ResourceDictionary
Source="pack://application:,,,/MaterialDesignColors;component/Themes/Recommended/Primary/Mat
erialDesignColor.DeepPurple.xaml" />
                <ResourceDictionary
Source="pack://application:,,,/MaterialDesignColors;component/Themes/Recommended/Accent/Mate
rialDesignColor.Lime.xaml" />
            </ResourceDictionary.MergedDictionaries>
        </ResourceDictionary>
    </Application.Resources>
</Application>
```

## DB.cs

```
/using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using MySql.Data.MySqlClient;

namespace uTasks
{
    class DB
    {
        MySqlConnection connection = new MySqlConnection("server=141.8.193.236;
username=f0546657_root;" +
            " password=f0546657_root; database=f0546657_utasks");

        public void openConnection()
        {
            if(connection.State == System.Data.ConnectionState.Closed)
            {
                connection.Open();
            }
        }

        public void closeConnection()
        {
            if (connection.State == System.Data.ConnectionState.Open)
            {
                connection.Close();
            }
        }

        public MySqlConnection GetConnection()
        {
            return connection;
        }
    }
}
```

## Login.xaml

```
/<Window x:Class="uTasks.Window2"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Login" Height="173.721" Width="315.334"
        Icon="C:\Users\user\Desktop\uTasks\uTasks\Images\1593671387_10.jpg"
        xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"
        WindowStartupLocation="CenterScreen">
    <Grid Background="white" VerticalAlignment="Bottom" HorizontalAlignment="Left"
Margin="0,0,-1,0" Height="143" Width="293">
        <Button Background="#078600" Content="Реєстрація" Height="27"
HorizontalAlignment="Left" Margin="163,106,0,0" x:Name="button1_Copy"
VerticalAlignment="Top" Width="120" Click="button2_Click" />
        <Button Background="#078600" Content="Авторизуватись" Height="27"
HorizontalAlignment="Left" Margin="10,106,0,0" Name="button1" VerticalAlignment="Top"
Width="136" Click="button1_Click" />
        <TextBox materialDesign:HintAssist.Hint="Логін" Height="23"
HorizontalAlignment="Left" Margin="95,24,0,0" Name="LoginBox" VerticalAlignment="Top"
Width="120" AcceptsTab="False" />
        <PasswordBox materialDesign:HintAssist.Hint="Пароль" Height="23"
HorizontalAlignment="Left" Margin="95,52,0,0" Name="PasswordBox" VerticalAlignment="Top"
Width="120" />
    </Grid>
</Window>
```

## Login.xaml.cs

```
>using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using MySql.Data.MySqlClient;
using System.Data;

namespace uTasks
{
    /// <summary>
    /// Логика взаимодействия для Window2.xaml
    /// </summary>
    public partial class Window2 : Window
    {
        private bool isAdmin = false;
        public Window2()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, RoutedEventArgs e)//login try
        {
            DB db = new DB();
            DataTable table = new DataTable();

            MySqlDataAdapter adapter = new MySqlDataAdapter();

            MySqlCommand cmd = new MySqlCommand("SELECT user_id, login, is_admin FROM users
WHERE `login` = @ul " +
                "AND `password` = @up", db.GetConnection());
            cmd.Parameters.Add("@ul", MySqlDbType.VarChar).Value = LoginBox.Text;
            cmd.Parameters.Add("@up", MySqlDbType.VarChar).Value = PasswordBox.Password;
```

```
            adapter.SelectCommand = cmd;
            adapter.Fill(table);

            if (table.Rows.Count>0){
                MessageBox.Show("Успішна авторизація!");
                if (Convert.ToString(table.Rows[0][2]) == "True")
                    isAdmin = true;
                Window1 w2 = new Window1(isAdmin, Convert.ToInt32(table.Rows[0][0]),
Convert.ToString(table.Rows[0][1]));
                w2.Show();
                Close();
            }
            else{
                MessageBox.Show("Не правильний логі або пароль. Спробуйте ще раз!");
            }
        }

        private void button2_Click(object sender, RoutedEventArgs e)//open regestration
        {
            Regestration w2 = new Regestration();
            w2.Show();
            Close();
        }
    }
}
```

## MainWindow.xaml

```
<Window x:Class="uTasks.Window1"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" xmlns:col="clr-
namespace:System.Collections;assembly=mscorlib"
        xmlns:sys="clr-namespace:System;assembly=mscorlib"
        Title="uTasks" Height="759.667" Width="1736.667"
        Loaded="Window_Loaded"
        ResizeMode="NoResize"
        Icon="C:\Users\user\Desktop\uTasks\uTasks\Images\1593671387_10.jpg"
        DataContext="{Binding}"
        Background="#FFEBEBEB"
        TextOptions.TextFormattingMode="Ideal"
        TextOptions.TextRenderingMode="Auto"
        WindowStartupLocation="CenterScreen">

    <Grid Margin="0,4,4,0">

        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="243*" />
            <ColumnDefinition Width="241*"/>
            <ColumnDefinition Width="0*" />
        </Grid.ColumnDefinitions>
        <DataGrid Style="{x:Null}" CellEditEnding="cellEditEnding"
AutoGenerateColumns="False" Name="DataGrid" Margin="10,10,0,13" CanUserAddRows="False"
                  Grid.ColumnSpan="2" RenderTransformOrigin="0.459,0.203"
HorizontalAlignment="Left" Width="1403">

            <DataGrid.RowStyle>
                <Style TargetType="DataGridRow">
                    <Style.Triggers>
                        <DataTrigger Binding="{Binding isActive}" Value="true">
                            <Setter Property="Background" Value="#F98E8E"/>
                        </DataTrigger>
                    </Style.Triggers>
                </Style>
            </DataGrid.RowStyle>

            <DataGrid.Columns>
                <DataGridTextColumn Header="#" IsReadOnly="True" Binding="{Binding
Path=order_id}" Width="Auto"></DataGridTextColumn>
```

```
                <DataGridTextColumn Header="Ім'я" IsReadOnly="False"  Binding="{Binding
Path=name, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}"
Width="Auto"></DataGridTextColumn>
                <DataGridTextColumn Header="Телефон" IsReadOnly="False"  Binding="{Binding
Path=phone_number, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}"
Width="Auto"></DataGridTextColumn>
                <DataGridTextColumn Header="Прийняв" IsReadOnly="True"  Binding="{Binding
Path=login, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}"
Width="Auto"></DataGridTextColumn>
                <DataGridTextColumn Header="Описання" IsReadOnly="False"  Binding="{Binding
Path=deskr, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}"
Width="Auto"></DataGridTextColumn>
                <DataGridTextColumn Header="Дата отримання" IsReadOnly="False"
Binding="{Binding Path=receipt_time, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}"
Width="Auto"></DataGridTextColumn>
                <DataGridTextColumn Header="Дата видачі" IsReadOnly="False"
Binding="{Binding Path=return_time, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}"
Width="Auto"></DataGridTextColumn>
                <DataGridTextColumn Header="Статус" IsReadOnly="False"  Binding="{Binding
Path=status, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}"
Width="Auto"></DataGridTextColumn>

                <DataGridTemplateColumn Header="Список запчастин" IsReadOnly="True">
                    <DataGridTemplateColumn.CellTemplate>
                        <DataTemplate>
                            <StackPanel>
                                <TextBlock Height ="50" Name ="myTextBox" Text="{Binding
partsList}" />
                                <Button Background="#078600" Click="PartsBtn_Click"
Content="Редагувати" />
                            </StackPanel>
                        </DataTemplate>
                    </DataGridTemplateColumn.CellTemplate>
                </DataGridTemplateColumn>

                <DataGridTemplateColumn Header="Список послуг" IsReadOnly="True">
                    <DataGridTemplateColumn.CellTemplate>
                        <DataTemplate>
                            <StackPanel>
                                <TextBlock Height ="50" Name ="myTextBoxServices"
Text="{Binding servicesList}" />
                                <Button Background="#078600" Click="ServicesBtn_Click"
Content="Редагувати" />
                            </StackPanel>
                        </DataTemplate>
                    </DataGridTemplateColumn.CellTemplate>
                </DataGridTemplateColumn>

                <DataGridTextColumn Header="Загальна вартість" IsReadOnly="True"
Binding="{Binding Path=price}" Width="Auto"></DataGridTextColumn>
            </DataGrid.Columns>
        </DataGrid>

        <Button Background="#078600" Content="Статистика" Height="28" Margin="659,647,0,0"
x:Name="BtnStat" VerticalAlignment="Top" Click="BtnStat_Click" Grid.Column="1"
HorizontalAlignment="Left" Width="100" RenderTransformOrigin="1.34,0.536" />
        <Button Background="#078600" Content="Видалити" Height="28" Margin="0,10,101,0"
Name="button3" VerticalAlignment="Top" Click="button3_Click"
RenderTransformOrigin="0.495,0.536" Grid.Column="1" HorizontalAlignment="Right" Width="100"
/>
        <Button Background="#078600" Content="Додати" Height="28" Margin="659,196,0,0"
x:Name="addBtn" VerticalAlignment="Top" RenderTransformOrigin="0.147,0.696"
HorizontalAlignment="Left" Width="100" Click="addBtn_click" Grid.Column="1" />
        <TextBox Height="24" Margin="566,55,118,0" Name="addName" VerticalAlignment="Top"
DataContext="{Binding}" AcceptsReturn="False" Grid.Column="1" />
        <TextBox Height="24" Margin="566,83,118,0" Name="addPhone" VerticalAlignment="Top"
Grid.Column="1" />
        <TextBox Height="24" Margin="566,113,118,0" Name="addDetails"
VerticalAlignment="Top" Grid.Column="1" />
```

```
        <Label Content="Ім'я" Height="28" Margin="742,51,2,0" Name="label1"
VerticalAlignment="Top" Grid.Column="1" RenderTransformOrigin="-1.133,0.5" />
        <Label Content="Телефон" Height="28" Margin="742,79,2,0" Name="label2"
VerticalAlignment="Top" Grid.Column="1" />
        <Label Content="Необхідні деталі" Height="28" HorizontalAlignment="Left"
Margin="742,105,0,0" Name="label4" VerticalAlignment="Top" Width="100" Grid.Column="1" />
        <Label Content="Гарантований&#xD;&#xA; час видачі" Height="42"
HorizontalAlignment="Left" Margin="742,133,0,0" x:Name="label4_Copy" VerticalAlignment="Top"
Width="100" Grid.Column="1" RenderTransformOrigin="0.45,2.179" />
        <DatePicker Name ="DatePicker" Margin="566,147,118,0" VerticalAlignment="Top"
RenderTransformOrigin="2.04,0.792" Height="24" Grid.Column="1"/>
    </Grid>
</Window>
```

## MainWindow.xaml.cs

```
>using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using System.Data;
using System.Collections.ObjectModel;
using MySql.Data.MySqlClient;
using System.Threading.Tasks;
using System.Reflection;

namespace uTasks
{
    public partial class Window1 : Window
    {
        private DataTable ds;
        private User thisUser;

        public class User
        {
            public User(bool isAdmin, int id, string login)
            {
                this.isAdmin = isAdmin;
                this.id = id;
                this.login = login;
            }
            public bool isAdmin { get; set; }
            public int id { get; set; }
            public string login { get; set; }
        }


        public Window1(bool isAdmin, int id, string login)
        {
            InitializeComponent();
            thisUser = new User(isAdmin, id, login);
            if (!thisUser.isAdmin)
            {
                BtnStat.Visibility = Visibility.Hidden;
            }
        }

        private void addBtn_click(object sender, RoutedEventArgs e)
        {
            if (addName.Text == "")
            {
```

```
                MessageBox.Show("Введіть ім'я замовника.");
            }
            else if (addPhone.Text == "")
            {
                MessageBox.Show("Введіть номер замовника.");
            }
            else if (Convert.ToString(DatePicker.SelectedDate) == "")
            {
                MessageBox.Show("Оберіть гарантовану дату видачі.");
            }
            else
            {
                DB db = new DB();
                db.openConnection();
                MySqlCommand cmd = new MySqlCommand("SELECT customer_id FROM customers WHERE
name = @name", db.GetConnection());
                cmd.Parameters.Add("@name", MySqlDbType.VarChar).Value = addName.Text;
                DataTable buf = new DataTable();
                MySqlDataAdapter adapter = new MySqlDataAdapter();
                adapter.SelectCommand = cmd;
                adapter.Fill(buf);

                if (buf.Rows.Count < 1)
                {
                    cmd = new MySqlCommand("INSERT INTO customers (customer_id, name,
phone_number) VALUES (NULL, @name, @pn)", db.GetConnection());
                    cmd.Parameters.Add("@name", MySqlDbType.VarChar).Value = addName.Text;
                    cmd.Parameters.Add("@pn", MySqlDbType.VarChar).Value = addPhone.Text;
                    cmd.ExecuteNonQuery();
                    MessageBox.Show("Додано нового користувача!");

                    cmd = new MySqlCommand("SELECT customer_id FROM customers WHERE name =
@name", db.GetConnection());
                    cmd.Parameters.Add("@name", MySqlDbType.VarChar).Value = addName.Text;
                    adapter = new MySqlDataAdapter();
                    adapter.SelectCommand = cmd;
                    adapter.Fill(buf);
                }
                else
                {
                    MessageBoxResult result = MessageBox.Show("Такий кристувач вже існує.
Обновити його номер?", "Info", MessageBoxButton.YesNo);
                    switch (result)
                    {
                        case MessageBoxResult.Yes:
                            cmd = new MySqlCommand("UPDATE customers SET phone_number =
@number WHERE customers.name = @name", db.GetConnection());
                            cmd.Parameters.Add("@name", MySqlDbType.VarChar).Value =
addName.Text;
                            cmd.Parameters.Add("@number", MySqlDbType.VarChar).Value =
addPhone.Text;
                            cmd.ExecuteNonQuery();
                            break;
                    }

                }

                cmd = new MySqlCommand("INSERT INTO `orders` (`order_id`, `deskr`, `status`,
`receipt_time`, `return_time`, `user_id`, `customer_id`)" +
                    " VALUES(NULL, @deskr, 'Тільки прийнято', @receipt, @retrunTime,
@userId, @customerId)", db.GetConnection());
                cmd.Parameters.Add("@deskr", MySqlDbType.VarChar).Value = addDetails.Text;
                cmd.Parameters.Add("@receipt", MySqlDbType.Date).Value = DateTime.Now;
                cmd.Parameters.Add("@retrunTime", MySqlDbType.Date).Value =
DatePicker.SelectedDate;
                cmd.Parameters.Add("@userId", MySqlDbType.Int32).Value = thisUser.id;
                cmd.Parameters.Add("@customerId", MySqlDbType.Int32).Value = buf.Rows[0][0];
                cmd.ExecuteNonQuery();

                db.closeConnection();
```

```
                ds.Rows.Add(ds.Rows.Count + 1, addName.Text, addPhone.Text, thisUser.login,
addDetails.Text, DateTime.Now, DatePicker.SelectedDate, "Тільки прийнято", 0, true);
            }
        }

        void cellEditEnding(object sender, DataGridCellEditEndingEventArgs e)//обновляємо БД
після редагування таблиці
        {
            checkStatus();//перевіряємо, чи змінився статус
            DB db = new DB();
            db.openConnection();

            MySqlCommand cmd = new MySqlCommand("UPDATE orders SET " +
                " deskr = @ds, status = @st, receipt_time = @rc, return_time = @srt" +
                " WHERE orders.order_id = @c", db.GetConnection());

            cmd.Parameters.Add("@ds", MySqlDbType.VarChar).Value =
ds.Rows[DataGrid.Items.IndexOf(DataGrid.SelectedItem)][4];
            cmd.Parameters.Add("@st", MySqlDbType.VarChar).Value =
ds.Rows[DataGrid.Items.IndexOf(DataGrid.SelectedItem)][7];
            cmd.Parameters.Add("@rc", MySqlDbType.DateTime).Value =
ds.Rows[DataGrid.Items.IndexOf(DataGrid.SelectedItem)][5];
            cmd.Parameters.Add("@srt", MySqlDbType.DateTime).Value =
ds.Rows[DataGrid.Items.IndexOf(DataGrid.SelectedItem)][6];
            cmd.Parameters.Add("@c", MySqlDbType.Int16).Value =
ds.Rows[DataGrid.Items.IndexOf(DataGrid.SelectedItem)][0];
            cmd.ExecuteNonQuery();

            cmd = new MySqlCommand("Update customers SET phone_number = @pn WHERE name =
@name", db.GetConnection());
            cmd.Parameters.Add("@pn", MySqlDbType.VarChar).Value =
ds.Rows[DataGrid.Items.IndexOf(DataGrid.SelectedItem)][2];
            cmd.Parameters.Add("@name", MySqlDbType.VarChar).Value =
ds.Rows[DataGrid.Items.IndexOf(DataGrid.SelectedItem)][1];
            cmd.ExecuteNonQuery();

            //перевіряємо, чи є такий користувач вже в бд
            DataTable buf = new DataTable();
            MySqlDataAdapter adapter = new MySqlDataAdapter();
            cmd = new MySqlCommand("SELECT * FROM `customers` WHERE `name` = @n",
db.GetConnection());
            cmd.Parameters.Add("@n", MySqlDbType.VarChar).Value =
ds.Rows[DataGrid.Items.IndexOf(DataGrid.SelectedItem)][1];
            adapter.SelectCommand = cmd;
            adapter.Fill(buf);
            if(buf.Rows.Count<1)//якщо немає - додаємо нового
            {
                cmd = new MySqlCommand("INSERT INTO customers (customer_id, name,
phone_number) VALUES (NULL, @name, @number)", db.GetConnection());
                cmd.Parameters.Add("@name", MySqlDbType.VarChar).Value =
ds.Rows[DataGrid.Items.IndexOf(DataGrid.SelectedItem)][1];
                cmd.Parameters.Add("@number", MySqlDbType.VarChar).Value =
ds.Rows[DataGrid.Items.IndexOf(DataGrid.SelectedItem)][2];
                cmd.ExecuteNonQuery();

                cmd = new MySqlCommand("SELECT customer_id FROM customers WHERE name =
@name", db.GetConnection());
                cmd.Parameters.Add("@name", MySqlDbType.VarChar).Value =
ds.Rows[DataGrid.Items.IndexOf(DataGrid.SelectedItem)][1];
                adapter.SelectCommand = cmd;
                DataTable customerId = new DataTable();
                adapter.Fill(customerId);

                cmd = new MySqlCommand("UPDATE orders SET customer_id = @newCId WHERE
order_id = @c", db.GetConnection());
                cmd.Parameters.Add("@c", MySqlDbType.Int16).Value =
ds.Rows[DataGrid.Items.IndexOf(DataGrid.SelectedItem)][0];
                cmd.Parameters.Add("@newCId", MySqlDbType.Int32).Value =
customerId.Rows[0][0];
```

```
                cmd.ExecuteNonQuery();
            }
            else
            {
                cmd = new MySqlCommand("SELECT customer_id FROM customers WHERE name =
@name", db.GetConnection());
                cmd.Parameters.Add("@name", MySqlDbType.VarChar).Value =
ds.Rows[DataGrid.Items.IndexOf(DataGrid.SelectedItem)][1];
                adapter.SelectCommand = cmd;
                DataTable customerId = new DataTable();
                adapter.Fill(customerId);

                cmd = new MySqlCommand("UPDATE orders SET customer_id = @newCId WHERE
order_id = @c", db.GetConnection());
                cmd.Parameters.Add("@c", MySqlDbType.Int16).Value =
ds.Rows[DataGrid.Items.IndexOf(DataGrid.SelectedItem)][0];
                cmd.Parameters.Add("@newCId", MySqlDbType.Int32).Value =
customerId.Rows[0][0];
                cmd.ExecuteNonQuery();
            }

            updateTable();
            db.closeConnection();
        }

        private void Window_Loaded(object sender, RoutedEventArgs e)//заповнюємо таблицю
після завантаження
        {
            try
            {
                updateTable();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

        private void updateTable()
        {
            //беремо дані з таблиці замовлень та вставляємо в таблицю
            DB db = new DB();
            db.openConnection();
            MySqlCommand cmd = new MySqlCommand("SELECT orders.order_id, customers.name,
customers.phone_number, users.login, " +
                "orders.deskr, orders.receipt_time, orders.return_time, orders.status FROM
orders, customers, users WHERE orders.customer_id = customers.customer_id " +
                "AND orders.user_id = users.user_id", db.GetConnection());
            MySqlDataAdapter adp = new MySqlDataAdapter(cmd);
            ds = new DataTable();
            adp.Fill(ds);

            //знаходимо загальну вартість деталей
            ds.Columns.Add("price", typeof(int));
            cmd = new MySqlCommand("SELECT parts_list.order_id, parts.price FROM parts_list,
parts WHERE parts_list.part_id = parts.part_id", db.GetConnection());
            adp = new MySqlDataAdapter(cmd);
            DataTable partsPriceBuf = new DataTable();
            adp.Fill(partsPriceBuf);
            db.closeConnection();

            int sum;
            for (int i = 0; i < ds.Rows.Count; i++)
            {
                sum = 0;
                for (int j = 0; j < partsPriceBuf.Rows.Count; j++)
                {
                    if (Convert.ToInt16(partsPriceBuf.Rows[j][0]) ==
Convert.ToInt16(ds.Rows[i][0]))
                    {
```

```
                    sum += Convert.ToInt16(partsPriceBuf.Rows[j][1]);
                }
            }
            ds.Rows[i][8] = sum;
        }

        //знаходимо загальну вартість послуг
        cmd = new MySqlCommand("SELECT services_list.order_id, services.price FROM
services_list, services WHERE services_list.service_id = services.service_id",
            db.GetConnection());
        adp = new MySqlDataAdapter(cmd);
        DataTable servicesPriceBuf = new DataTable();
        adp.Fill(servicesPriceBuf);
        db.closeConnection();
        for (int i = 0; i < ds.Rows.Count; i++)
        {
            sum = 0;
            for (int j = 0; j < servicesPriceBuf.Rows.Count; j++)
            {
                if (Convert.ToInt16(servicesPriceBuf.Rows[j][0]) ==
Convert.ToInt16(ds.Rows[i][0]))
                {
                    sum += Convert.ToInt16(servicesPriceBuf.Rows[j][1]);
                }
            }
            ds.Rows[i][8] = Convert.ToInt16(ds.Rows[i][8]) + sum;
        }

        ds.Columns.Add("isActive", typeof(bool));
        checkStatus();

        //створюємо список деталей
        ds.Columns.Add("partsList", typeof(string));
        cmd = new MySqlCommand("SELECT parts_list.order_id, parts.name FROM parts_list,
parts WHERE parts_list.part_id = parts.part_id", db.GetConnection());
        adp = new MySqlDataAdapter(cmd);
        DataTable partsList = new DataTable();
        adp.Fill(partsList);
        db.closeConnection();

        int z;
        for (int i = 0; i < partsList.Rows.Count; i++)
        {
            for (int j = 0; j < ds.Rows.Count; j++)
            {
                if((int)partsList.Rows[i][0] == (int)ds.Rows[j][0])
                {
                    ds.Rows[j][10] += Convert.ToString(partsList.Rows[i][1]) + ";\n";
                }
            }
        }

        //створюємо список послуг
        ds.Columns.Add("servicesList", typeof(string));
        cmd = new MySqlCommand("SELECT services_list.order_id, services.name FROM
services_list, services " +
            "WHERE services_list.service_id = services.service_id", db.GetConnection());
        adp = new MySqlDataAdapter(cmd);
        DataTable servicesList = new DataTable();
        adp.Fill(servicesList);

        for (int i = 0; i < servicesList.Rows.Count; i++)
        {
            for (int j = 0; j < ds.Rows.Count; j++)
            {
                if ((int)servicesList.Rows[i][0] == (int)ds.Rows[j][0])
                {
                    ds.Rows[j][11] += Convert.ToString(servicesList.Rows[i][1]) + ";\n";
                }
            }
```

```
            }



            DataGrid.ItemsSource = new DataView(ds);
            db.closeConnection();
        }

        private void checkStatus()
        {
            string str;
            for (int i = 0; i < ds.Rows.Count; i++)
            {
                str = Convert.ToString(ds.Rows[i][7]).ToLower();
                if (str=="виконано" || Convert.ToString(ds.Rows[i][7]) == "+")
                {
                    ds.Rows[i][9] = false;
                }
                else
                {
                    ds.Rows[i][9] = true;
                }
            }
        }

        private void PartsBtn_Click(object sender, RoutedEventArgs e)
        {
            PartsList w = new
PartsList((int)ds.Rows[DataGrid.Items.IndexOf(DataGrid.SelectedItem)][0], this, thisUser);
            w.Show();
        }

        private void button3_Click(object sender, RoutedEventArgs e)
        {
            DB db = new DB();
            db.openConnection();

            MySqlCommand cmd = new MySqlCommand("DELETE FROM parts_list WHERE order_id =
@id", db.GetConnection());

            int i = (int)ds.Rows[DataGrid.Items.IndexOf(DataGrid.SelectedItem)][0];
            var b = DataGrid.SelectedItem;
            cmd.Parameters.Add("@id", MySqlDbType.Int16).Value =
ds.Rows[DataGrid.Items.IndexOf(DataGrid.SelectedItem)][0];
            cmd.ExecuteNonQuery();

            cmd = new MySqlCommand("DELETE FROM services_list WHERE order_id = @id",
db.GetConnection());
            cmd.Parameters.Add("@id", MySqlDbType.Int16).Value =
ds.Rows[DataGrid.Items.IndexOf(DataGrid.SelectedItem)][0];
            cmd.ExecuteNonQuery();

            cmd = new MySqlCommand("DELETE FROM orders WHERE order_id = @id",
db.GetConnection());
            cmd.Parameters.Add("@id", MySqlDbType.Int16).Value =
ds.Rows[DataGrid.Items.IndexOf(DataGrid.SelectedItem)][0];
            cmd.ExecuteNonQuery();

            db.closeConnection();

            ds.Rows[DataGrid.Items.IndexOf(DataGrid.SelectedItem)].Delete();
            ds.AcceptChanges();
        }

        private void BtnStat_Click(object sender, RoutedEventArgs e)
        {
            Statistics s = new Statistics(ds, thisUser);
            s.Show();
            this.Close();
```

```
        }
        private void ServicesBtn_Click(object sender, RoutedEventArgs e)
        {
            ServicesList ws = new
ServicesList((int)ds.Rows[DataGrid.Items.IndexOf(DataGrid.SelectedItem)][0], this,
thisUser);
            ws.Show();
        }
    }
}
```

## PartsList.xaml

```
}<Window x:Class="uTasks.PartsList"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"
        xmlns:local="clr-namespace:uTasks"
        mc:Ignorable="d"
        Icon="C:\Users\user\Desktop\uTasks\uTasks\Images\1593671387_10.jpg"
        WindowStartupLocation="CenterScreen"
        Title="PartsList" Height="333.042" Width="548.188" Loaded="Window_Loaded">
    <Grid>
        <ListView Name ="List" HorizontalAlignment="Left" Height="245" Margin="251,10,0,0"
VerticalAlignment="Top" Width="279" RenderTransformOrigin="0.48,0.523" SelectedIndex="4"
SelectionMode="Extended">
            <ListView.View>
                <GridView>
                    <GridViewColumn Header="Назва" DisplayMemberBinding="{Binding _name}"/>
                    <GridViewColumn Header="Вартість" DisplayMemberBinding="{Binding
_price}"/>
                </GridView>
            </ListView.View>
        </ListView>
        <Button Background="#078600" x:Name="acceptBtn" Content="Зберігти"
HorizontalAlignment="Left" Margin="251,260,0,0" VerticalAlignment="Top" Width="115"
Click="acceptBtn_Click"/>
        <Button Background="#078600" x:Name="cancelBtn" Content="Відмінити"
Margin="415,260,10,0" VerticalAlignment="Top" Click="cancelBtn_Click"/>
        <Button Background="#078600" x:Name="addOneBtn" Content="Додати найменування"
HorizontalAlignment="Left" Margin="55,137,0,0" VerticalAlignment="Top" Width="182"
Click="addOneBtn_Click"/>
        <Button Background="#078600" x:Name="deleteBtn" Content="Видалити найменування"
HorizontalAlignment="Left" Margin="55,23,0,0" VerticalAlignment="Top" Width="182"
Click="deleteBtn_Click"/>
        <TextBox x:Name="newOneTxtPrice" HorizontalAlignment="Left" Height="23"
Margin="88,98,0,0" TextWrapping="Wrap" materialDesign:HintAssist.Hint="Вартість"
VerticalAlignment="Top" Width="120"/>
        <TextBox x:Name="newOneTxtName" HorizontalAlignment="Left" Height="23"
Margin="88,70,0,0" TextWrapping="Wrap" materialDesign:HintAssist.Hint="Назва"
VerticalAlignment="Top" Width="120"/>
    </Grid>
</Window>
```

## PartsList.xaml.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
```

```
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using static uTasks.Window1;

namespace uTasks
{
    public class Row
    {
        public Row(string str, int p)
        {
            _name = str;
            _price = p;
        }

        private string name;
        public string _name
        {
            get
            {
                return name;
            }
            set
            {
                name = value;
            }

        }

        private int price;
        public int _price
        {
            get
            {
                return price;
            }
            set
            {
                price = value;
            }
        }
    }
    /// <summary>
    /// Логика взаимодействия для PartsList.xaml
    /// </summary>
    public partial class PartsList : Window
    {
        private DataTable ItemsList = new DataTable();
        private int selectedRow;
        private Window1 w;
        User u;
        public PartsList(int selectedRow, Window1 w, User u)
        {
            this.selectedRow = selectedRow;
            this.w = w;
            this.u = u;
            InitializeComponent();
        }

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            updateList();
        }

        private void updateList()
        {
            try
            {
```

```
                DB db = new DB();
                db.openConnection();
                MySqlCommand cmd = new MySqlCommand("SELECT name, part_id, price FROM
parts", db.GetConnection());
                MySqlDataAdapter adp = new MySqlDataAdapter(cmd);
                ItemsList.Clear();
                adp.Fill(ItemsList);
                db.closeConnection();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }

            ObservableCollection<Row> collection = new ObservableCollection<Row>();
            this.List.ItemsSource = collection;
            for (int i = 0; i < ItemsList.Rows.Count; i++)
            {
                Row r = new Row(Convert.ToString(ItemsList.Rows[i][0]),
Convert.ToInt32(ItemsList.Rows[i][2]));
                collection.Add(r);
            }
        }

        private void acceptBtn_Click(object sender, RoutedEventArgs e)
        {
            //удаление старого листа
            DB db = new DB();
            db.openConnection();
            MySqlCommand cmd = new MySqlCommand("DELETE FROM `parts_list` WHERE
`parts_list`.`order_id` = @c", db.GetConnection());
            cmd.Parameters.Add("@c", MySqlDbType.Int16).Value = selectedRow;
            cmd.ExecuteNonQuery();

            //добавление отдельных записей для нового листа
            int selectedIndex = 0;
            for (int i = 0; i < List.SelectedItems.Count; i++)
            {
                for (int j = 0; j < ItemsList.Rows.Count; j++)
                {
                    Row r = (Row)List.SelectedItems[i];
                    if (r._name == Convert.ToString(ItemsList.Rows[j][0]))
                    {
                        selectedIndex = Convert.ToInt16(ItemsList.Rows[j][1]);
                        cmd = new MySqlCommand("INSERT INTO `parts_list` (`part_list_id`,
`part_id`, `order_id`)" +
                            " VALUES (NULL, @part_id, @order_id)", db.GetConnection());
                        cmd.Parameters.Add("@part_id", MySqlDbType.Int16).Value =
selectedIndex;
                        cmd.Parameters.Add("@order_id", MySqlDbType.Int16).Value =
selectedRow;
                        cmd.ExecuteNonQuery();
                    }
                }
            }
            db.closeConnection();

            w.Close();
            w = new Window1(u.isAdmin, u.id, u.login);
            w.Show();

            this.Close();
        }

        private void cancelBtn_Click(object sender, RoutedEventArgs e)
        {
            w.Close();
            w = new Window1(u.isAdmin, u.id, u.login);
            w.Show();
            this.Close();
```

```
        }

        private void addOneBtn_Click(object sender, RoutedEventArgs e)
        {
            if (newOneTxtName.Text == "" || newOneTxtPrice.Text == "")
            {
                MessageBox.Show("Заповніть поля для додавання нового елементу!");
            }
            else
            {
                DB db = new DB();
                db.openConnection();
                MySqlCommand cmd = new MySqlCommand("INSERT INTO parts (part_id, name,
price) VALUES (NULL, @name, @price)", db.GetConnection());
                cmd.Parameters.Add("@name", MySqlDbType.VarChar).Value = newOneTxtName.Text;
                cmd.Parameters.Add("@price", MySqlDbType.Int16).Value =
Convert.ToInt32(newOneTxtPrice.Text);
                cmd.ExecuteNonQuery();
                updateList();
                db.closeConnection();
            }
        }

        private void deleteBtn_Click(object sender, RoutedEventArgs e)
        {
            if(List.SelectedItems.Count<1)
            {
                MessageBox.Show("Оберіть елементи для видалення");
            }
            else
            {
                DB db = new DB();
                db.openConnection();

                int selectedIndex = 0;
                for (int i = 0; i < List.SelectedItems.Count; i++)
                {
                    for (int j = 0; j < ItemsList.Rows.Count; j++)
                    {
                        Row r = (Row)List.SelectedItems[i];
                        if (r._name == Convert.ToString(ItemsList.Rows[j][0]))
                        {
                            selectedIndex = Convert.ToInt16(ItemsList.Rows[j][1]);

                            MySqlCommand cmd = new MySqlCommand("DELETE FROM parts_list
WHERE part_id = @id", db.GetConnection());
                            cmd.Parameters.Add("@id", MySqlDbType.Int16).Value =
selectedIndex;
                            cmd.ExecuteNonQuery();

                            cmd = new MySqlCommand("DELETE FROM parts WHERE part_id = @id",
db.GetConnection());
                            cmd.Parameters.Add("@id", MySqlDbType.Int16).Value =
selectedIndex;
                            cmd.ExecuteNonQuery();
                        }
                    }
                }
                db.closeConnection();
            }
            updateList();
        }
    }
}
```

## Regestration.xaml

```
}<Window x:Class="uTasks.Regestration"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

```
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"
        xmlns:local="clr-namespace:uTasks"
        mc:Ignorable="d"
        Icon="C:\Users\user\Desktop\uTasks\uTasks\Images\1593671387_10.jpg"
        Title="Regestration" Height="254" Width="298.667"
        WindowStartupLocation="CenterScreen">
    <Grid Margin="0,0,2,1">
        <TextBox materialDesign:HintAssist.Hint="Ім'я" x:Name="NameBox"
HorizontalAlignment="Left" Height="23" TextWrapping="Wrap" VerticalAlignment="Top"
Width="120" Margin="85,10,0,0"/>
        <TextBox materialDesign:HintAssist.Hint="Фамілія" x:Name="LNameBox"
HorizontalAlignment="Left" Height="23" TextWrapping="Wrap" VerticalAlignment="Top"
Width="120" Margin="85,38,0,0"/>
        <TextBox materialDesign:HintAssist.Hint="Логін" x:Name="LoginBox"
HorizontalAlignment="Left" Height="23" TextWrapping="Wrap" VerticalAlignment="Top"
Width="120" Margin="85,66,0,0"/>
        <PasswordBox materialDesign:HintAssist.Hint="Пароль" x:Name="PasswordBox"
HorizontalAlignment="Left" Height="23" VerticalAlignment="Top" Width="120"
Margin="85,94,0,0"/>
        <PasswordBox materialDesign:HintAssist.Hint="Повторіть пароль" x:Name="PasswordBox2"
HorizontalAlignment="Left" Height="23" VerticalAlignment="Top" Width="120"
Margin="85,122,0,0" RenderTransformOrigin="0.508,0.507"/>
        <Button Background="#078600" Content="OK" HorizontalAlignment="Left"
VerticalAlignment="Top" Width="75" Margin="36,161,0,0" Click="Button_Click"/>
        <Button Background="#078600" Content="Назад" HorizontalAlignment="Left"
VerticalAlignment="Top" Width="75" Margin="186,161,0,0" Click="Button_Click_1"/>
    </Grid>
</Window>
```

## Regestration.xaml.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace uTasks
{
    /// <summary>
    /// Логика взаимодействия для Regestration.xaml
    /// </summary>
    public partial class Regestration : Window
    {
        public Regestration()
        {
            InitializeComponent();
        }

        private void Button_Click(object sender, RoutedEventArgs e)//regestration try
        {
            if(PasswordBox.Password != PasswordBox2.Password)
            {
                MessageBox.Show("Паролі не співпадають. ");
            }
            else if(LoginBox.Text == "")
            {
```

```
                MessageBox.Show("Введіть ім'я.");
            }
            else if(PasswordBox.Password == "")
            {
                MessageBox.Show("Введіть пароль.");
            }
            else if (NameBox.Text == "")
            {
                MessageBox.Show("Введіть ім'я.");
            }
            else if (LNameBox.Text == "")
            {
                MessageBox.Show("Введіть фамілію.");
            }
            else if(isUserExist())
            {
                MessageBox.Show("Користувач з таким логіном вже існує. Спробуйте інший.");
            }
            else
            {
                DB db = new DB();
                MySqlCommand cmd = new MySqlCommand("INSERT INTO `users` (`user_id`,
`login`," +
                    " `password`, `is_admin`, `fname`, `lname`) VALUES (NULL, @ul, @up," +
                    " 0, @ufn, @uln)", db.GetConnection());
                cmd.Parameters.Add("@ul", MySqlDbType.VarChar).Value = LoginBox.Text;
                cmd.Parameters.Add("@up", MySqlDbType.VarChar).Value = PasswordBox.Password;
                cmd.Parameters.Add("@ufn", MySqlDbType.VarChar).Value = NameBox.Text;
                cmd.Parameters.Add("@uln", MySqlDbType.VarChar).Value = LNameBox.Text;

                db.openConnection();
                MessageBox.Show("Реєстрація пройшла успішно.");
                Window2 w = new Window2();
                w.Show();
                Close();
                db.closeConnection();
            }
        }

        public bool isUserExist()//check is user exist
        {
            DB db = new DB();
            DataTable table = new DataTable();

            MySqlDataAdapter adapter = new MySqlDataAdapter();

            MySqlCommand cmd = new MySqlCommand("SELECT * FROM `users` WHERE `login` = @ul
",
                db.GetConnection());
            cmd.Parameters.Add("@ul", MySqlDbType.VarChar).Value = LoginBox.Text;

            adapter.SelectCommand = cmd;
            adapter.Fill(table);

            if (table.Rows.Count > 0)
            {
                MessageBox.Show("Такий логін вже існує.");
                return true;
            }
            else
            {
                return false;
            }
        }

        private void Button_Click_1(object sender, RoutedEventArgs e)//back to login
        {
            Window2 w = new Window2();
            w.Show();
            Close();
```

```
        }
    }
}
```

## ServicesList.xaml

```
<Window x:Class="uTasks.ServicesList"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"
        xmlns:local="clr-namespace:uTasks"
        mc:Ignorable="d"
        Icon="C:\Users\user\Desktop\uTasks\uTasks\Images\1593671387_10.jpg"
        WindowStartupLocation="CenterScreen"
        Title="ServicesList" Height="317.938" Width="551.645" Loaded="Window_Loaded">
    <Grid>
        <ListView Name ="List" HorizontalAlignment="Left" Height="231" Margin="248,0,0,0"
VerticalAlignment="Top" Width="286" RenderTransformOrigin="0.48,0.523" SelectedIndex="4"
SelectionMode="Multiple">
            <ListView.View>
                <GridView>
                    <GridViewColumn Header="Назва" DisplayMemberBinding="{Binding _name}"/>
                    <GridViewColumn Header="Вартість" DisplayMemberBinding="{Binding
_price}"/>
                </GridView>
            </ListView.View>
        </ListView>
        <Button Background="#078600" x:Name="acceptBtn" Content="Зберігти"
HorizontalAlignment="Left" Margin="248,236,0,0" VerticalAlignment="Top" Width="115"
Click="acceptBtn_Click"/>
        <Button Background="#078600" x:Name="cancelBtn" Content="Відмінити"
Margin="419,236,10,0" VerticalAlignment="Top" Click="cancelBtn_Click"/>
        <Button Background="#078600" x:Name="addOneBtn" Content="Додати найменування"
HorizontalAlignment="Left" Margin="38,138,0,0" VerticalAlignment="Top" Width="182"
Click="addOneBtn_Click"/>
        <Button Background="#078600" x:Name="deleteBtn" Content="Видалити найменування"
HorizontalAlignment="Left" Margin="38,22,0,0" VerticalAlignment="Top" Width="182"
Click="deleteBtn_Click"/>
        <TextBox x:Name="newOneTxtPrice" HorizontalAlignment="Left" Height="23"
Margin="66,98,0,0" TextWrapping="Wrap" materialDesign:HintAssist.Hint="Вартість"
VerticalAlignment="Top" Width="120"/>
        <TextBox x:Name="newOneTxtName" HorizontalAlignment="Left" Height="23"
Margin="66,70,0,0" TextWrapping="Wrap" materialDesign:HintAssist.Hint="Назва"
VerticalAlignment="Top" Width="120"/>
    </Grid>
</Window>
```

ServicesList.xaml.cs

```
>using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using static uTasks.Window1;

namespace uTasks
```

```
{
    /// <summary>
    /// Логика взаимодействия для ServicesList.xaml
    /// </summary>
    public partial class ServicesList : Window
    {
        private DataTable ServicesList_ = new DataTable();
        private int selectedRow;
        private Window1 w;
        private User u;
        public ServicesList(int selectedRow, Window1 w, User u)
        {
            this.selectedRow = selectedRow;
            this.w = w;
            this.u = u;
            InitializeComponent();
        }

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            try
            {
                updateList();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

        private void updateList()
        {
            ServicesList_.Clear();
            DB db = new DB();
            db.openConnection();
            MySqlCommand cmd = new MySqlCommand("SELECT name, service_id, price FROM
services", db.GetConnection());
            MySqlDataAdapter adp = new MySqlDataAdapter(cmd);
            adp.Fill(ServicesList_);
            db.closeConnection();
            ObservableCollection<Row> collection = new ObservableCollection<Row>();
            this.List.ItemsSource = collection;
            for (int i = 0; i < ServicesList_.Rows.Count; i++)
            {
                Row r = new Row(Convert.ToString(ServicesList_.Rows[i][0]),
Convert.ToInt32(ServicesList_.Rows[i][2]));
                collection.Add(r);
            }
        }
        private void acceptBtn_Click(object sender, RoutedEventArgs e)
        {
            //удаление старого листа
            DB db = new DB();
            db.openConnection();
            MySqlCommand cmd = new MySqlCommand("DELETE FROM `services_list` WHERE
`services_list`.`order_id` = @c", db.GetConnection());
            cmd.Parameters.Add("@c", MySqlDbType.Int16).Value = selectedRow;
            cmd.ExecuteNonQuery();

            //добавление отдельных записей для нового листа
            int selectedIndex = 0;
            for (int i = 0; i < List.SelectedItems.Count; i++)
            {
                for (int j = 0; j < ServicesList_.Rows.Count; j++)
                {
                    Row r = (Row)List.SelectedItems[i];
                    if (r._name == Convert.ToString(ServicesList_.Rows[j][0]))
                    {
                        selectedIndex = Convert.ToInt16(ServicesList_.Rows[j][1]);
```

```
                        cmd = new MySqlCommand("INSERT INTO `services_list`
(`services_list_id`, `service_id`, `order_id`)" +
                             " VALUES (NULL, @service_id, @order_id)", db.GetConnection());
                        cmd.Parameters.Add("@service_id", MySqlDbType.Int16).Value =
selectedIndex;
                        cmd.Parameters.Add("@order_id", MySqlDbType.Int16).Value =
selectedRow;
                        cmd.ExecuteNonQuery();
                    }
                }
            }
            db.closeConnection();

            w.Close();
            w = new Window1(u.isAdmin, u.id, u.login);
            w.Show();

            this.Close();
        }

        private void cancelBtn_Click(object sender, RoutedEventArgs e)
        {
            w.Close();
            w = new Window1(u.isAdmin, u.id, u.login);
            w.Show();
            this.Close();
        }

        private void addOneBtn_Click(object sender, RoutedEventArgs e)
        {
            if (newOneTxtName.Text == "" || newOneTxtPrice.Text == "")
            {
                MessageBox.Show("Заповніть поля для додавання нового елементу!");
            }
            else
            {
                DB db = new DB();
                db.openConnection();
                MySqlCommand cmd = new MySqlCommand("INSERT INTO services (service_id, name,
price) VALUES (NULL, @name, @price)", db.GetConnection());
                cmd.Parameters.Add("@name", MySqlDbType.VarChar).Value = newOneTxtName.Text;
                cmd.Parameters.Add("@price", MySqlDbType.Int16).Value =
Convert.ToInt32(newOneTxtPrice.Text);
                cmd.ExecuteNonQuery();
                updateList();
                db.closeConnection();
                newOneTxtName.Text = "";
                newOneTxtPrice.Text = "";
            }
        }

        private void deleteBtn_Click(object sender, RoutedEventArgs e)
        {
            if (List.SelectedItems.Count < 1)
            {
                MessageBox.Show("Оберіть елементи для видалення");
            }
            else
            {
                DB db = new DB();
                db.openConnection();

                int selectedIndex = 0;
                for (int i = 0; i < List.SelectedItems.Count; i++)
                {
                    for (int j = 0; j < ServicesList_.Rows.Count; j++)
                    {
                        Row r = (Row)List.SelectedItems[i];
                        if (r._name == Convert.ToString(ServicesList_.Rows[j][0]))
                        {
```

```
                            selectedIndex = Convert.ToInt16(ServicesList_.Rows[j][1]);

                            MySqlCommand cmd = new MySqlCommand("DELETE FROM services_list
WHERE service_id = @id", db.GetConnection());
                            cmd.Parameters.Add("@id", MySqlDbType.Int16).Value =
selectedIndex;
                            cmd.ExecuteNonQuery();

                            cmd = new MySqlCommand("DELETE FROM services WHERE service_id =
@id", db.GetConnection());
                            cmd.Parameters.Add("@id", MySqlDbType.Int16).Value =
selectedIndex;
                            cmd.ExecuteNonQuery();
                        }
                    }
                }
                db.closeConnection();
            }
            updateList();
        }
    }
}
```

## Statistics.xaml

```
<Window
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:uTasks"
        xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"
x:Class="uTasks.Statistics"
        mc:Ignorable="d"
        Icon="C:\Users\user\Desktop\uTasks\uTasks\Images\1593671387_10.jpg"
        Title="Statistics" Height="407.925" Width="872.065"
        WindowStartupLocation="CenterScreen">
    <Grid>
        <Label x:Name="L1" Content="Label" HorizontalAlignment="Left" Margin="529,39,0,0"
VerticalAlignment="Top" AutomationProperties.Name="L1"/>
        <Label x:Name="L2" Content="Label" HorizontalAlignment="Left" Margin="529,63,0,0"
VerticalAlignment="Top" AutomationProperties.Name="L2"/>
        <Label x:Name="L3" Content="Label" HorizontalAlignment="Left" Margin="529,92,0,0"
VerticalAlignment="Top" AutomationProperties.Name="L3" RenderTransformOrigin="0.472,0.458"/>
        <Label x:Name="L4" Content="Label" HorizontalAlignment="Left" Margin="529,121,0,0"
VerticalAlignment="Top" RenderTransformOrigin="-0.295,-1.027"
AutomationProperties.Name="L4"/>
        <Label Content="Файл з інформацією про замовлення та статистикою за обраний період:"
HorizontalAlignment="Left" Margin="262,304,0,0" VerticalAlignment="Top"/>
        <Button Background="#078600" Content="Створити" HorizontalAlignment="Left"
Margin="404,333,0,0" VerticalAlignment="Top" Width="120" Click="Button_Click"/>
        <Label x:Name="L1_Copy" Content="Початок: " HorizontalAlignment="Left"
Margin="7,9,0,0" VerticalAlignment="Top" AutomationProperties.Name="L1"
RenderTransformOrigin="0.519,0.514"/>
        <DatePicker x:Name="startDatePick" SelectedDateChanged="startDataChanged"
HorizontalAlignment="Left" Margin="72,6,0,0" VerticalAlignment="Top"
RenderTransformOrigin="0.483,0.507" Width="80"/>
        <DatePicker x:Name="endDatePick" SelectedDateChanged="endDataChanged"
HorizontalAlignment="Left" Margin="262,6,0,0" VerticalAlignment="Top"
RenderTransformOrigin="0.5,0.5" Width="80"/>
        <Label x:Name="L1_Copy1" Content="Кінець: " HorizontalAlignment="Left"
Margin="197,4,0,0" VerticalAlignment="Top" AutomationProperties.Name="L1"
RenderTransformOrigin="0.519,0.514"/>

        <ListView x:Name ="List" HorizontalAlignment="Left" Height="219" Margin="10,80,0,0"
VerticalAlignment="Top" Width="514">
            <ListView.View>
                <GridView>
                    <GridViewColumn Header="Логін" DisplayMemberBinding="{Binding login}"/>
```

```
                    <GridViewColumn Header="Кількість замовлень"
DisplayMemberBinding="{Binding ordersCount}"/>
                    <GridViewColumn Header="Виконаних замовлень"
DisplayMemberBinding="{Binding undoneOrdersCount}"/>
                </GridView>
            </ListView.View>
        </ListView>
        <Label x:Name="L1_Copy2" Content="Статистика стосовно працівників:"
HorizontalAlignment="Left" Margin="152,39,0,0" VerticalAlignment="Top"
AutomationProperties.Name="L1"/>
        <Button Background="#078600" x:Name="BackBtn" Content="Назад"
HorizontalAlignment="Left" Margin="779,335,0,0" VerticalAlignment="Top" Width="75"
Click="BackBtn_Click"/>

    </Grid>
</Window>
```

## Statistics.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using System.Collections.ObjectModel;
using iTextSharp.text;
using iTextSharp.text.pdf;
//using System;
using System.IO;
using System.Data;
using MySql.Data.MySqlClient;
using static uTasks.Window1;
//using System.Text;


namespace uTasks
{
    /// <summary>
    /// Логика взаимодействия для Statistics.xaml
    /// </summary>
    public partial class Statistics : Window
    {
        private DataTable dt;
        private DateTime startDate;
        private DateTime endDate;
        private DataTable usersStat;
        private int unDoneCount, proceeds, periodCount;
        private User u;
        public Statistics(DataTable dt, User u)
        {
            this.u = u;
            InitializeComponent();
            this.dt = dt;//копіюємо таблицю з основної форми в цей клас

            startDate = DateTime.Now;
            startDate = new DateTime(startDate.Year, startDate.Month, 1);
            endDate = DateTime.Now;
            endDate = new DateTime(endDate.Year, endDate.Month, endDate.Day);

            startDatePick.SelectedDate = startDate;
            endDatePick.SelectedDate = endDate;

            fillusersStatistics();
            fillStatistics();
```

```
        }

        private void updateDates()
        {
            startDate = (DateTime)startDatePick.SelectedDate;
            startDate = new DateTime(startDate.Year, startDate.Month, startDate.Day);
            endDate = (DateTime)endDatePick.SelectedDate;
            endDate = new DateTime(endDate.Year, endDate.Month, endDate.Day);
        }

        private void fillStatistics()//встановлюємо значення статистики
        {
            L1.Content = "Загальна кількість замовлень: " + dt.Rows.Count;
            unDoneCount = 0; proceeds = 0; periodCount = 0;
            for (int i = 0; i < dt.Rows.Count; i++)
            {
                if (Convert.ToDateTime(dt.Rows[i][5])>startDate &&
Convert.ToDateTime(dt.Rows[i][6]) < endDate)
                {
                    proceeds += Convert.ToInt16(dt.Rows[i][8]);
                    periodCount++;
                    if (Convert.ToBoolean(dt.Rows[i][9]) == true)
                    {
                        unDoneCount++;
                    }
                }
            }
            L2.Content = "Кількість замовлень за вказаний період: " + periodCount;
            L3.Content = "З них не виконано: " + unDoneCount;
            L4.Content = "Виручка за вказаний період: " + proceeds + " грн";
        }

        private void fillusersStatistics()
        {
            DataTable buf;
            DB db = new DB();
            db.openConnection();

            MySqlCommand cmd = new MySqlCommand("SELECT user_id, login FROM users",
db.GetConnection());
            MySqlDataAdapter adp = new MySqlDataAdapter(cmd);
            usersStat = new DataTable();
            adp.Fill(usersStat);
            usersStat.Columns.Add("ordersCount", typeof(int));
            usersStat.Columns.Add("undoneOrdersCount", typeof(int));
            for (int i = 0; i < usersStat.Rows.Count; i++)
            {
                //додаємо запис про загальну кількість прийнятих замовлень
                buf = new DataTable();
                cmd = new MySqlCommand("SELECT COUNT(*) FROM orders WHERE
orders.receipt_time > @rt" +
                    " AND orders.return_time < @srt AND orders.user_id = @uid",
db.GetConnection());
                cmd.Parameters.Add("@uid", MySqlDbType.Int16).Value = usersStat.Rows[i][0];
                cmd.Parameters.Add("@rt", MySqlDbType.DateTime).Value = startDate;
                cmd.Parameters.Add("@srt", MySqlDbType.DateTime).Value = endDate;
                adp = new MySqlDataAdapter(cmd);
                adp.Fill(buf);
                usersStat.Rows[i][2] = buf.Rows[0][0];

                int z = Convert.ToInt32(buf.Rows[0][0]);
                //додаємо запис про кількість виконаних замовлень

                buf = new DataTable();
                cmd = new MySqlCommand("SELECT COUNT(*) FROM orders WHERE
orders.receipt_time > @rt AND orders.return_time < @srt" +
                    " AND orders.user_id = @uid AND (orders.status = '+' OR orders.status =
'виконано' OR orders.status = 'Виконано')", db.GetConnection());
                cmd.Parameters.Add("@uid", MySqlDbType.Int16).Value = usersStat.Rows[i][0];
                cmd.Parameters.Add("@rt", MySqlDbType.DateTime).Value = startDate;
```

```
                cmd.Parameters.Add("@srt", MySqlDbType.DateTime).Value = endDate;
                adp = new MySqlDataAdapter(cmd);
                adp.Fill(buf);
                usersStat.Rows[i][3] = buf.Rows[0][0];

                int z2 = Convert.ToInt32(buf.Rows[0][0]);
            }

            List.ItemsSource = new DataView(usersStat);
            db.closeConnection();
        }

        private void startDataChanged(object sender, RoutedEventArgs e)
        {
            endDatePick.SelectedDate = endDate;
            updateDates();
            fillStatistics();
            fillusersStatistics();
        }

        private void BackBtn_Click(object sender, RoutedEventArgs e)
        {
            Window1 w = new Window1(u.isAdmin, u.id, u.login);
            w.Show();
            this.Close();
        }

        private void endDataChanged(object sender, RoutedEventArgs e)
        {
            startDatePick.SelectedDate = startDate;
            updateDates();
            fillStatistics();
            fillusersStatistics();
        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            DB db = new DB();
            DataTable buf = new DataTable();
            MySqlCommand cmd = new MySqlCommand("SELECT orders.order_id, customers.name,
customers.phone_number, users.login, " +
                "orders.deskr, orders.receipt_time, orders.return_time, orders.status FROM
orders, customers, users " +
                "WHERE orders.receipt_time > @rt AND orders.return_time < @srt AND
orders.customer_id = customers.customer_id " +
                "AND orders.user_id = users.user_id", db.GetConnection());
            cmd.Parameters.Add("@rt", MySqlDbType.DateTime).Value = startDate;
            cmd.Parameters.Add("@srt", MySqlDbType.DateTime).Value = endDate;
            MySqlDataAdapter adp = new MySqlDataAdapter(cmd);
            adp.Fill(buf);

            var document = new iTextSharp.text.Document();
            using (var writer = PdfWriter.GetInstance(document, new
FileStream("C:/Users/user/Desktop/result.pdf", FileMode.Create)))
            {
                document.Open();

                BaseFont baseFont = BaseFont.CreateFont("C:/Windows/Fonts/arial.ttf",
BaseFont.IDENTITY_H, BaseFont.NOT_EMBEDDED);
                iTextSharp.text.Font font = new iTextSharp.text.Font(baseFont,
iTextSharp.text.Font.DEFAULTSIZE, iTextSharp.text.Font.NORMAL);
                var helvetica = new
iTextSharp.text.Font(iTextSharp.text.Font.FontFamily.HELVETICA, 12);
                var helveticaBase = helvetica.GetCalculatedBaseFont(false);
                writer.DirectContent.BeginText();
                writer.DirectContent.SetFontAndSize(baseFont, 14f);

                document.Add(new iTextSharp.text.Paragraph("Інформація за період з " +
startDate.Day + "." + startDate.Month + "." + startDate.Year + " по " +
                    + endDate.Day + "." + endDate.Month + "." + endDate.Year, font));
```

```
                document.Add(new iTextSharp.text.Paragraph("\nЗагальна кількість замовлень:
" + dt.Rows.Count, font));
                document.Add(new iTextSharp.text.Paragraph("\nКількість замовлень за
вказаний період: " + periodCount, font));
                document.Add(new iTextSharp.text.Paragraph("\nЗ них не виконано: " +
unDoneCount, font));
                document.Add(new iTextSharp.text.Paragraph("\nВиручка за вказаний період: "
+ proceeds + " грн", font));
                writer.DirectContent.EndText();

                document.Add(new iTextSharp.text.Paragraph("\n"));

                ////////////////////////////
                //Создаем объект таблицы и передаем в нее число столбцов таблицы из нашего
датасета
                PdfPTable table = new PdfPTable(buf.Columns.Count-1);

                //Добавим в таблицу общий заголовок
                PdfPCell cell = new PdfPCell(new Phrase("БД"));

                cell.Colspan = buf.Columns.Count-1;
                cell.HorizontalAlignment = 1;
                //Убираем границу первой ячейки, чтобы балы как заголовок
                cell.Border = 0;
                table.AddCell(cell);

                //Сначала добавляем заголовки таблицы
                for (int j = 0; j < buf.Columns.Count-1; j++)
                {
                    cell = new PdfPCell(new Phrase(new Phrase(buf.Columns[j].ColumnName,
font)));
                    //Фоновый цвет (необязательно, просто сделаем по красивее)
                    cell.BackgroundColor = iTextSharp.text.BaseColor.LIGHT_GRAY;
                    table.AddCell(cell);
                }

                //Добавляем все остальные ячейки
                for (int j = 0; j < buf.Rows.Count; j++)
                {
                    for (int k = 0; k < buf.Columns.Count-1; k++)
                    {
                        table.AddCell(new Phrase(buf.Rows[j][k].ToString(), font));
                    }
                }
                //Добавляем таблицу в документ
                document.Add(table);

                MessageBox.Show("Документ збережено на робочий стіл");

                ////////////////////////////
                document.Close();
                writer.Close();
            }

        }
    }
}
```