ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка застосунку To Do для складання списків справ та планування їх дедлайнів мовою C# з використанням баз даних»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Данило КЛУСЕНКО
(підпис)

Виконав: здобувач вищої освіти групи ПД-41

Данило КЛУСЕНКО

Керівник: Віталій ЗАЛИВА
доктор філософії (PhD)

Рецензент: _____

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Клусенку Данилу Миколайовичу

1. Тема кваліфікаційної роботи: «Розробка застосунку To Do для складання списків справ та планування їх дедлайнів мовою C# з використанням баз даних»
керівник кваліфікаційної роботи доктор філософії (PhD), старший викладач кафедри ІПЗ Віталій ЗАЛИВА,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36.

2. Строк подання кваліфікаційної роботи «28» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи:

3.1 Теорія роботи Windows Forms.

3.2 Методи роботи з базою даних.

3.3 Науково-технічна література

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

1. Теоретична частина. Огляд та аналіз існуючих методів та технологій створення, зберігання та редагування списків справ.

2. Проектування застосунку для створення, зберігання та редагування списків справ.

3. Програмна реалізація та опис функціонування застосунку для створення, зберігання та редагування списків справ.

5. Перелік графічного матеріалу: *презентація*

5.1. Аналіз аналогів.

5.2. Вимоги до програмного забезпечення.

5.3. Програмні засоби та інструменти реалізації.

5.4. Діаграма варіантів використання.

5.5. Блок-схема функціонування застосунку.

5.6. Екранні форми

5.7. Апробація результатів дослідження.

6. Дата видачі завдання«28» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	28.02.-06.03.2024	
2	Аналіз та дослідження існуючих аналогів	07.03-13.03.2024	
3	Огляд існуючих програмних рішень для створення, зберігання та редагування списків справ	14.03-31.03.2024	
4	Проектування застосунку для реалізації створення, редагування та зберігання списків справ	01.04-04.04.2024	
5	Програмна реалізація застосунку	05.04-19.04.2024	
6	Тестування застосунку	20.04-28.04.2024	
7	Оформлення роботи: вступ, висновки, реферат	29.04-05.05.2024	
8	Розробка демонстраційних матеріалів	06.05-12.05.2024	
9	Попередній захист роботи	13.05-31.05.2024	

Здобувач вищої освіти _____ Данило КЛУСЕНКО

(підпис)

Керівник
кваліфікаційної роботи _____ Віталій ЗАЛИВА

(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 42 стор., 1 табл., 24 рис., 20 джерел.

Мета роботи – спрощення зберігання, відслідковування справ та їх термінів виконання за допомогою desktop-застосунку, створеного мовою C#.

Об'єкт дослідження – процес створення та відслідковування поставлених задач.

Предмет дослідження – десктоп-застосунок для створення, зберігання та відслідковування списку справ.

Короткий зміст роботи: В роботі проаналізовано алгоритми та методи для створення, зберігання та редагування списків справ. Проаналізовано існуючі інструментальні засоби для ведення списків справ: Taskwarrior, Simple Sticky Notes Розроблено алгоритм роботи застосунку та програмно реалізовані ключові функціональні можливості, зокрема: функції реєстрації та авторизації, створення задач, їх перегляд, редагування та видалення. Підключення та робота з хмарною базою даних. Фільтрація справ за статусом їх виконання. Розроблено та запроваджено функцію локалізації тексту. Проведено функціональне та модульне тестування додатку. В роботі використано бібліотеку MySQLConnector для підключення та роботу з базою даних, BCrypt для хешування паролю для забезпечення безпеки.

Сферою використання застосунку є організація повсякденних справ і завдань користувачів для підвищення їх продуктивності.

КЛЮЧОВІ СЛОВА: СПИСКИ СПРАВ, ПЛАНУВАННЯ, ТРЕКІНГ ЗАВДАНЬ, ТАЙМ-МЕНЕДЖМЕНТ.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми та її актуальність: Людському розуму завжди було складно запам'ятовувати великі обсяги інформації, адже він не є ідеальним механізмом: ми можемо забувати, спотворювати спогади та втрачати концентрацію. Коли виникає потреба запам'ятовувати плани на майбутнє, розподіляти завдання для полегшення виконання та загалом планувати порядок дій, корисно мати візуальне уявлення у вигляді списку справ. Списки справ допомагають структурувати наші плани, що полегшує контроль за виконанням кожного завдання. Для цього існують спеціальні інструменти, які допомагають складати такі списки та організовувати виконання завдань. Отже, щоб максимально ефективно використовувати свої розумові ресурси та уникнути перевантаження, варто звернути увагу на сучасні технології для складання та управління списками справ.

Об'єкт дослідження – процес створення та відслідковування поставлених задач.

Предмет дослідження – десктоп-застосунок для створення, зберігання та відслідковування списку справ.

Мета роботи – спрощення зберігання, відслідковування справ та їх термінів виконання за допомогою десктоп-застосунку, створеного мовою C#.

Методи дослідження – метод архітектурного проектування та прототипування, метод аналізу вимог та метод тестування.

Для реалізації мети необхідно вирішити такі питання:

1. Проаналізувати існуючі аналоги та визначити головні функції та характеристики майбутнього застосунку.
2. Сформулювати вимоги до функціоналу, інтерфейсу користувача, продуктивності, безпеки та інших важливих аспектів.

3.

4. Вивчити та оцінити різні інструменти та технології, які можуть бути використані для розробки застосунку, включаючи мову програмування, середовище розробки, системи керування базами даних та бібліотеки.
5. Розробити інтерфейс користувача, реалізувати функціонал для створення, редагування та видалення списків справ та їх дедлайнів, а також для фільтрації списків справ. Створити та налаштувати базу даних для зберігання інформації.
6. Провести тестування застосунку. Виконати функціональне тестування, виявити та виправити помилки та недоліки.

Практичне значення одержаних результатів – застосунок застосовується для підвищення ефективності планування та виконання задач, що призведе до підвищення продуктивності виконання задач.

Галузь використання – повсякденне життя, робота та навчання.

Робота пройшла апробацію на Всеукраїнській науково-технічній конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях»[1]

1 ТЕОРЕТИЧНА ЧАСТИНА

1.1 Аналіз предметної галузі

Сучасне життя стає все більш динамічним і насиченим, що вимагає від людей ефективного планування та управління часом. Для цього розроблено безліч інструментів, серед яких особливе місце займають додатки To-Do [2]. Вони дозволяють користувачам створювати списки завдань, встановлювати дедлайни та відстежувати їх виконання. Мова програмування C# є однією з найпопулярніших для розробки таких додатків, а використання баз даних забезпечує надійне зберігання та управління даними.

Головною метою додатків To-Do є допомога користувачам в організації їхньої діяльності. Користувачі прагнуть мати можливість швидко і легко додавати нові завдання до списку, встановлювати конкретні терміни виконання завдань, а також маркувати виконані завдання та переглядати загальний прогрес.

Для реалізації функціональності додатку ToDo на C# з використанням баз даних необхідно врахувати кілька ключових аспектів. База даних повинна забезпечувати надійне зберігання даних про завдання, їх пріоритети, дедлайни та статуси. Логіка для додавання, видалення, редагування та перегляду завдань повинна бути реалізована через відповідні CRUD (Create, Read, Update, Delete) операції.

Розробка додатку ToDo мовою C# передбачає використання таких технологій, як C# і .NET Framework для основної мови та платформи розробки, Windows Forms (WinForms) для створення інтерфейсу користувача, MySQL Connector для роботи з базою даних зі сторони застосунку, SQL Server для зберігання інформації про завдання та DBeaver для роботи зі сторони адміністрування та перегляду та редагуванню баз даних.

Розробка додатку ToDo для складання списків справ та планування їх дедлайнів мовою C# з використанням баз даних є актуальним завданням, яке може

значно покращити продуктивність користувачів. Врахування потреб користувачів, технічних вимог та вибір відповідних інструментів допоможуть створити конкурентоспроможний та корисний продукт.

1.2 Taskwarrior

Taskwarrior – це потужний інструмент з відкритим кодом для керування завданнями, який допомагає користувачам організовувати свої справи та контролювати їх виконання. Додаток пропонує безліч функцій, що дозволяють зберігати важливу інформацію про завдання та ефективно управляти своїм часом. Taskwarrior дає можливість створювати профілі завдань, додавати описи, встановлювати пріоритети та кінцеві терміни.

Одна з основних переваг Taskwarrior – це можливість вести журнал завдань. Цей журнал дозволяє користувачам фіксувати виконання завдань, відстежувати прогрес та реєструвати будь-які зміни в їх статусі. Наприклад, можна реєструвати завершення завдань, зміну пріоритетів, встановлення нових дедлайнів та інші важливі дані про свої завдання.

Додаток також включає функцію планування подій на довгострокову перспективу. Taskwarrior дозволяє створювати і керувати подіями на 90 днів вперед. Основні можливості включають:

- Встановлення нагадувань
- Планування зустрічей
- Управління проектами
- Відстеження прогресу

Цей функціонал стане в нагоді тим, хто хоче ретельно стежити за виконанням своїх завдань і не забувати про важливі справи у своєму напруженому графіку.

На рисунку 1.1 [3] показано деякі з основних команд, які використовуються в Taskwarrior для управління завданнями.

```
$ task list

ID Active Age  D P Project Tags  R Due        Description          Urg
 2 3h     3h                                  Started task           4
 8        3h                      2015-03-14 Overdue task        9.49
 9        3h                      2015-03-16 Due task            8.58
10        3h                      2015-03-31 Not yet due tasks   2.4
15        3h                    R 2015-03-31 Recurring task      2.4
13        3h                                 Blocking task         8
 3        3h    H                            High priority task    6
 4        3h    M                            Medium priority task  3.9
 5        3h    L                            Low priority task     1.8
12        3h              tag1               Tagged task           0.8
 1        3h                                 Ordinary task         0
14        3h  D                              Dependent task       -5
 7        3h      Garden                     Outdoor task          1
 6        3h      Home                       Household task        1

$ task summary

Project   Remaining Avg age Complete 0%                    100%
(none)           12 2 hours      0%
Garden            1  1 hour     66%
Home              1  1 hour     50%

3 projects
$ task ghistory

Year Month Number Added/Completed/Deleted
2015 March                                              19   3 1

Legend: Added, Completed, Deleted

$ task calendar

        March 2015                April 2015                May 2015

    Su Mo Tu We Th Fr Sa      Su Mo Tu We Th Fr Sa      Su Mo Tu We Th Fr Sa
 10  1  2  3  4  5  6  7  14                  1  2  3  4  18                 1  2
 11  8  9 10 11 12 13 14  15  5  6  7  8  9 10 11  19  3  4  5  6  7  8  9
 12 15 16 17 18 19 20 21  16 12 13 14 15 16 17 18  20 10 11 12 13 14 15 16
 13 22 23 24 25 26 27 28  17 19 20 21 22 23 24 25  21 17 18 19 20 21 22 23
 14 29 30 31              18 26 27 28 29 30        22 24 25 26 27 28 29 30
                                                   23 31

Legend: today, due, due-today, overdue, weekend, holiday, weeknumber.

$ task burndown.daily
                            Daily Burndown ()
18 |
   |
   |
   |
   |
   |
   |
   |
 9 |                                                                  Done
   |                                                                  Started
   |                                                                  Pending
   |
   |
   |
   |
   |
   |
 0 +------------------------------------------------------------------------
   23 24 25 26 27 28 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15
   Feb               Mar

  Add rate:         3.2/d      Estimated completion: No convergence
  Done/Delete rate: 0.7/d
```

Рис. 1.1 Приклад команд застосунку Taskwarrior

Переваги Taskwarrior:

- Можливість працювати офлайн
- Підтримує ієрархічну структуру задач
- Система фільтрації та пошуку задач

Недоліки:

- Для використання та керування потрібно знати спеціальні команди керування
- Відсутність графічного інтерфейсу
- Відсутня українська локалізація

1.3 Simple Sticky Notes

Simple Sticky Notes — це простий у використанні додаток, який дозволяє створювати і зберігати нотатки прямо на робочому столі. Цей інструмент особливо корисний для тих, хто шукає простий спосіб організувати свої думки, завдання та нагадування без необхідності встановлення складного програмного забезпечення чи реєстрації.

Однією з головних переваг Simple Sticky Notes є його легкість у використанні. Користувачі можуть швидко створювати нові нотатки, просто натискаючи на відповідну кнопку, і розташовувати їх у будь-якому місці на робочому столі. Це робить додаток ідеальним для швидких записів і нагадувань. Кожна нотатка може бути кастомізована: змінюйте колір фону, шрифт і розмір тексту, щоб краще організувати інформацію і зробити нотатки більш помітними.

На рисунках 1.2, 1.3 та 1.4 [4] зображено приклади створення нових нотаток, їх кастомізація та список збережених нотаток відповідно.

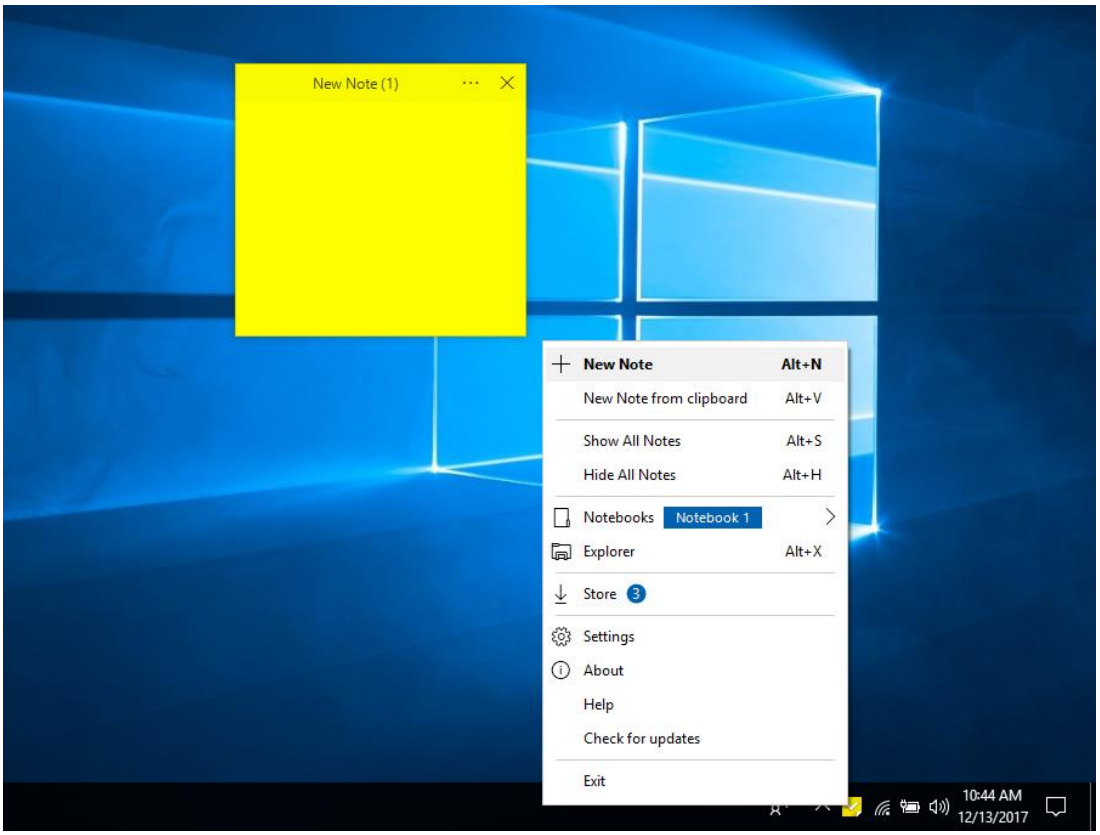


Рис. 1.2 Приклад створення задачі в Simple Sticky Notes

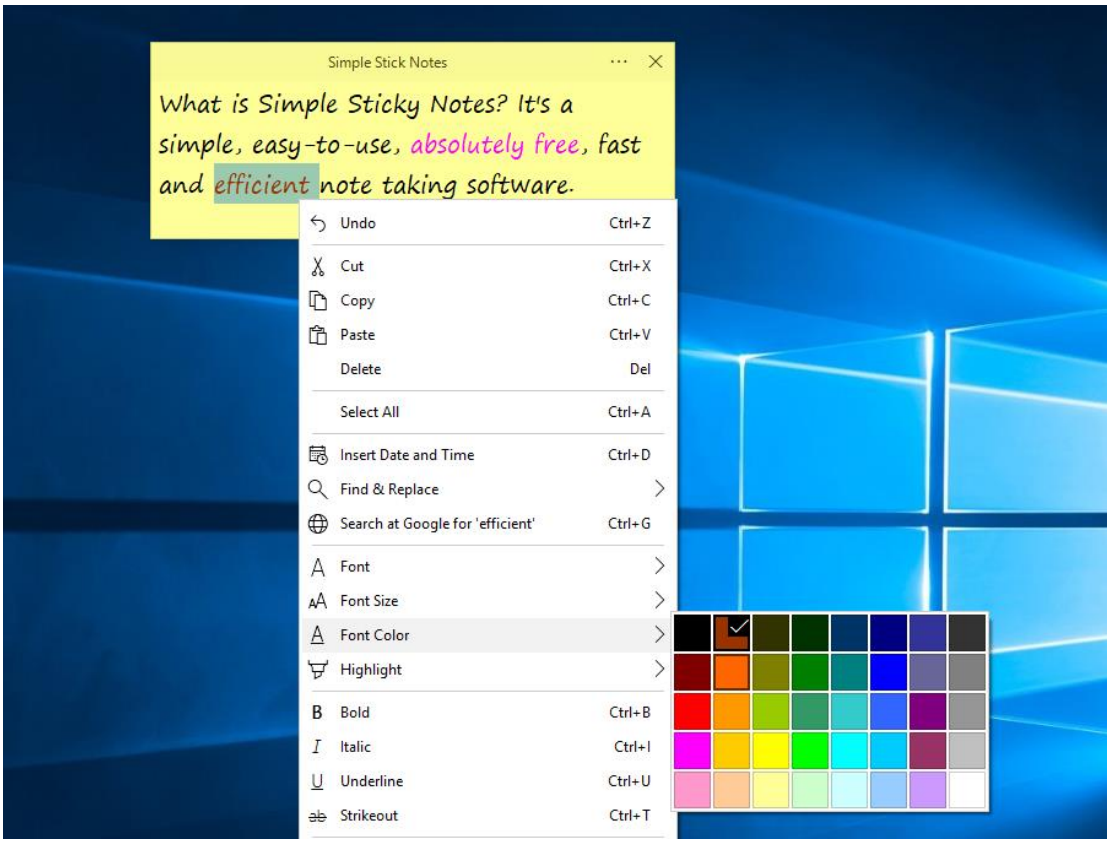


Рис. 1.3 Приклад кастомізації нотаток в Simple Sticky Notes

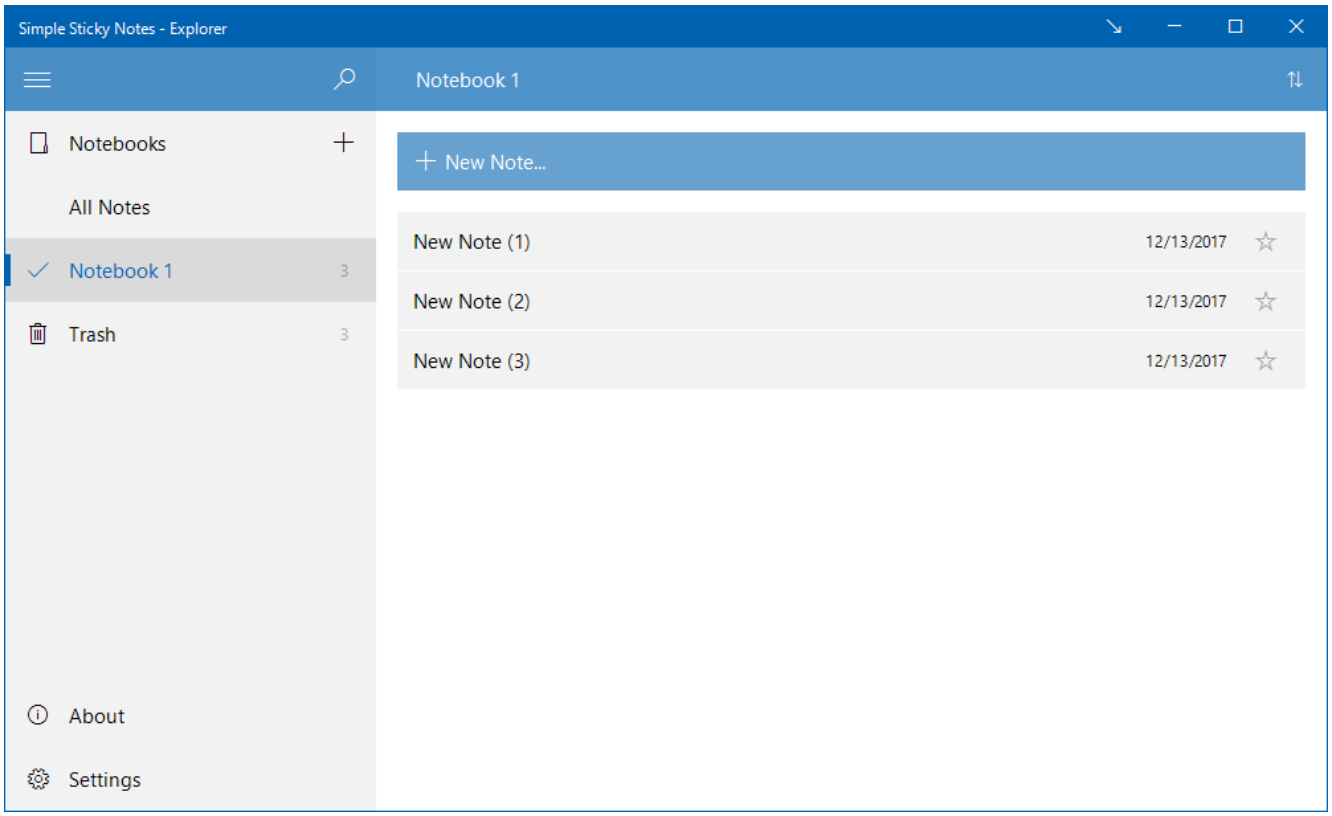


Рис. 1.4 Список створених нотаток

Переваги Simple Sticky Notes:

- Необмежена кількість задач
- Не вимагає реєстрації або встановлення додаткового програмного забезпечення
- Кастомізація тексту
- Можливість розташування заміток у будь-якому місці на робочому столі
- Присутня українська локалізація

Недоліки:

- Відсутність можливості встановлювати терміни виконання задач
- Відсутність можливості синхронізації задач між пристроями

1.4 Таблиця порівняння

Враховуючи розглянуті характеристики додатків можемо звести їх у порівняльну таблицю 1.1

Таблиця 1.1

Порівняльна таблиця аналогів

	Taskwarrior	Simple Sticky Notes	ToDo
Операційна система	Linux, macOS, Windows	Windows	Windows
Українська локалізація	-	+	+
Функція додавання опису	+	+	+
Функція встановлення кінцевого терміну	+	-	+
Розрахований на користувача-новачка	-	+	+
Синхронізація між пристроями	+	-	+
Функція трекінгу задач	+	-	+
Функція ієрархії задач	+	-	-

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Засоби реалізації

Для розробки застосунку ToDo для складання списків справ та планування їх дедлайнів було використано декілька сучасних інструментів і технологій, що забезпечують ефективний процес розробки та функціональність кінцевого продукту. Основними засобами реалізації стали мова програмування C#[5], середовище розробки Visual Studio 2022, інструменти для роботи з базами даних DBeaver[6] та MySQL[7], бібліотека MySQLConnector[8] і технологія WinForms[9].

C# була обрана як основна мова програмування через її високу продуктивність, підтримку об'єктно-орієнтованого програмування та інтеграцію з платформою .NET. Завдяки цьому, C# дозволяє створювати надійні та масштабовані застосунки. Крім того, наявність великої кількості бібліотек і підтримка від спільноти розробників роблять C# ідеальним вибором для розробки десктопних застосунків.

Visual Studio 2022 було використано як основне середовище розробки завдяки його потужним можливостям, таким як інтегрований налагоджувач, підтримка рефакторингу коду та широкий спектр інструментів для роботи з різними технологіями. Visual Studio забезпечує зручний інтерфейс для написання, тестування та налагодження коду, що значно прискорює процес розробки.

Для зберігання даних застосунку була обрана реляційна база даних MySQL. Ця СУБД є однією з найбільш популярних і широко використовуваних у світі завдяки своїй надійності, високій продуктивності та зручності в користуванні. MySQL дозволяє ефективно управляти великими обсягами даних і забезпечує високий рівень безпеки.

DBeaver було обрано як інструмент для адміністрування та управління базами даних завдяки його зручному інтерфейсу та підтримці багатьох типів баз даних. MySQL використовується як основна система управління базами даних

через її надійність, продуктивність і відкритий вихідний код. Це дозволяє зберігати та обробляти велику кількість даних, що є необхідним для функціонування ToDo-застосунку. Для взаємодії застосунку з базою даних було використано бібліотеку MySQLConnector. Ця бібліотека надає зручні методи для підключення до MySQL, виконання SQL-запитів та обробки результатів. Вона забезпечує високу продуктивність та надійність при роботі з базою даних, що критично важливо для забезпечення коректного збереження та швидкого доступу до даних.

WinForms було обрано як технологію для створення графічного інтерфейсу користувача (GUI) через її простоту використання та гнучкість. WinForms дозволяє швидко створювати інтуїтивно зрозумілі та зручні для користувача інтерфейси, що включають різноманітні елементи управління, такі як кнопки, текстові поля, списки та інші. Це сприяє підвищенню зручності використання застосунку.

Використання C#, Visual Studio 2022, DBeaver, MySQL, MySQLConnector та WinForms забезпечило створення надійного та ефективного застосунку ToDo для складання списків справ та планування їх дедлайнів. Ці інструменти та технології дозволили реалізувати всі необхідні функції та забезпечити високу продуктивність і зручність користування застосунком.

2.2 Функціональні вимоги

2.2.1 Реєстрація та авторизація користувача

Застосунок повинний вміти зберігати та виконувати валідацію введеного користувачем логіну та паролю. Логін та пароль мають зберігатися на хмарній базі даних. Застосунок також має вміти робити перевірку на існування користувача у базі даних з введеним логіном та відхиляти створення користувачем акаунту якщо такий логін вже існує .

2.2.2 Створення нових задач

Застосунок повинний вміти зберігати введені дані користувачем у базу даних з такими параметрами:

- Назва задачі: Короткий заголовок, який чітко описує суть завдання.
- Термін виконання: Дата до якої завдання має бути виконане. Це дозволяє користувачеві легко відслідковувати, які завдання мають вищий пріоритет.
- Опис задачі: Розширений текстовий опис, що надає додаткову інформацію про завдання, такі як необхідні дії, примітки або інші деталі.
- Статус задачі: Поточний стан завдання, який може включати такі значення, як "Заплановано", "В процесі" та "Виконано".

Також задача повинна мати унікальний ідентифікаційний номер та зв’язок з логіном користувача для прив’язки задачі до конкретного користувача.

2.2.3 Відображення задач користувача

Застосунок повинний вміти відображати усі задачі користувача з такими даними:

- Назва задачі: Короткий заголовок, який чітко описує суть завдання.
- Термін виконання: Дата до якої завдання має бути виконане. Це дозволяє користувачеві легко відслідковувати, які завдання мають вищий пріоритет.
- Опис задачі: Розширений текстовий опис, що надає додаткову інформацію про завдання, такі як необхідні дії, примітки або інші деталі.
- Статус задачі: Поточний стан завдання, який може включати такі значення, як "Заплановано", "В процесі" та "Виконано".

Також має бути функція фільтрації задач по статусу їх виконання для простішого відслідковування задач, які потребують їх виконання.

2.2.4 Редагування та видалення існуючих задач користувача

Застосунок повинний вміти видаляти та редагувати задачі користувача. Для цього у поля форми мають бути заповнені дані існуючої задачі які отримуються

завдяки ідентифікаційному номеру який було створено автоматично при створені задачі у базі даних. Змінені дані повинні бути змінені у базі даних.

2.2.5 Зберігання задач у базі даних

Застосунок повинний вміти працювати з віддаленою хмарною базою даних та працювати з таблицями в яких зберігаються користувачі та їх задачі. Для роботи з базою даних використовуються SQL команди які реалізовані в застосунку в форматі CRUD методів.

2.2.6 Зміна мови застосунку

Застосунок повинен мати функції зміни мови та обирання мови при першому запуску застосунку або при потребі користувача. Також застосунок має вміти працювати з різними локалізаціями та зберігати обрану мову користувачем.

2.3 Нефункціональні вимоги

2.3.1 Безпека

Паролі користувачів мають зберігатися в захешованому вигляді[10] у базі даних. Для цього буде використовуватися бібліотека BCrypt.Net яка є широко використовуваною на сьогодні.

2.3.2 Масштабованість

Застосунок повинен використовувати хмарну базу даних для збільшення продуктивності та забезпеченню змоги користувачам працювати з будь-якого пристрою без втрати даних та можливості роботи великої кількості людей одночасно з різних пристроїв. Також робота з хмарною базою даних дозволяє користувачам використовувати один застосунок під різними акаунтами без втрати, зміни чи розголошенню їх даних.

2.3.3 Локалізація

Застосунок повинен мати українську[11] та англійську локалізацію[12] та мати просту систему додавання нових локалізацій за допомогою додання у застосунок спеціальних файлів ресурсів що містять в собі локалізаційний переклад.

2.4 Діаграма використання

Діаграма використання (Use-Case Diagram) [13] є одним із ключових інструментів у мові моделювання UML (Unified Modeling Language), яка використовується для опису функціональних вимог до системи.

Вона дозволяє представити взаємодію між користувачами (акторами) та системою, показуючи різні сценарії використання (випадки використання), які реалізуються системою для досягнення певних цілей. У нашому випадку використано діаграму використання зображену на рисунку 2.1 для відображення основних функцій та взаємодії користувача і системи.

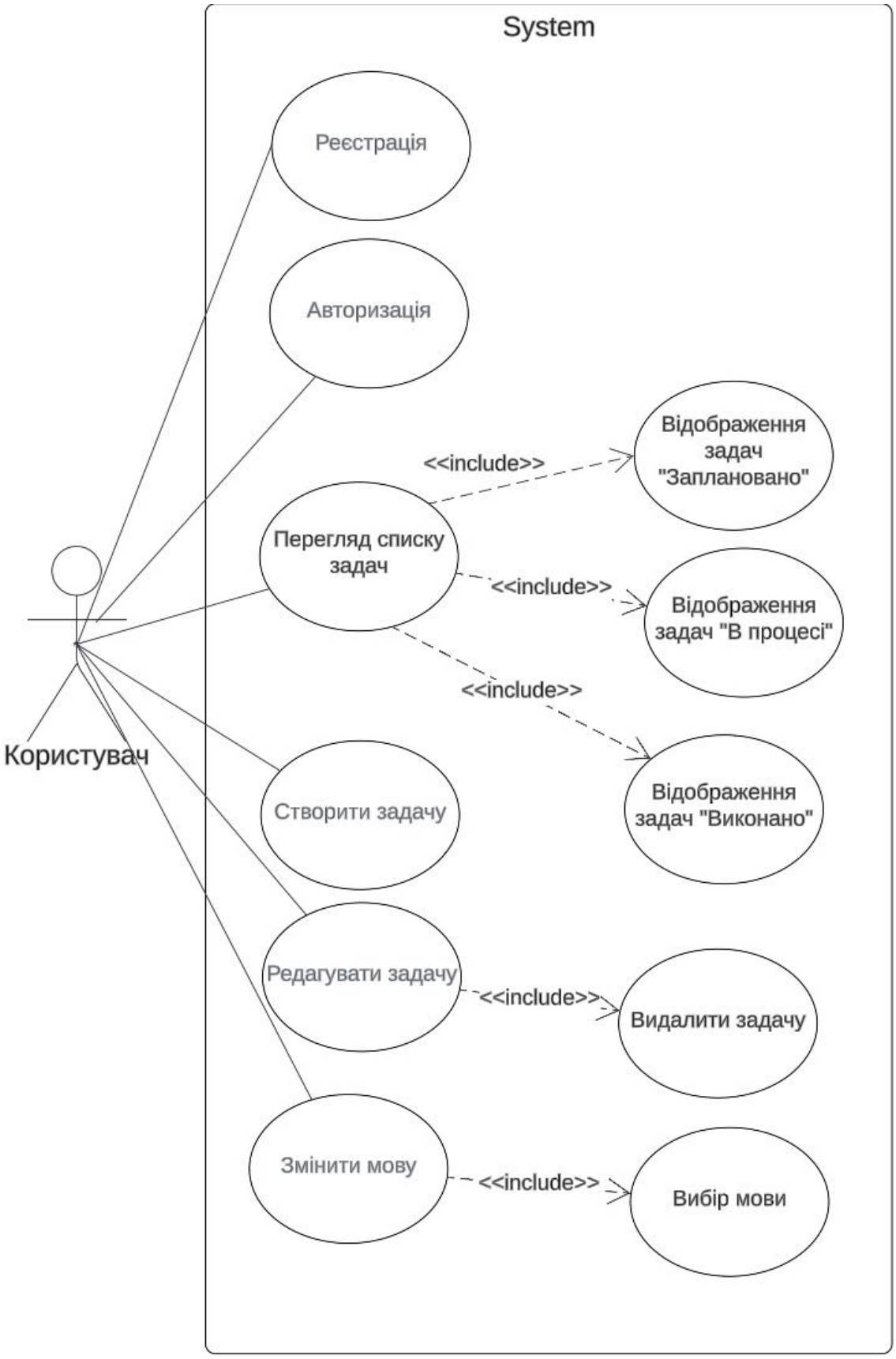


Рис. 2.1 Діаграма використання

2.5 Діаграма діяльності

Діаграма діяльності (Activity Diagram) [14] є одним із ключових інструментів у мові моделювання UML (Unified Modeling Language), яка використовується для опису динамічних аспектів системи. Вона дозволяє представити послідовність дій або кроків, що виконуються в процесі, показуючи як потоки управління та даних проходять через систему. Діаграма діяльності допомагає візуалізувати логіку виконання бізнес-процесів, робочих процесів або алгоритмів.

Діаграма діяльності складається з різних елементів, таких як дії (activity), переходи (transition), умови (decision),. Вона може також включати стартову точку і кінцеву точку що позначають початок і кінець процесу відповідно.

У нашому випадку, діаграма діяльності, зображена на рисунку 2.2, використовується для відображення послідовності дій та логіки виконання процесу в системі. Ця діаграма допомагає зрозуміти, які дії виконуються, в якому порядку, і як вони взаємодіють між собою, забезпечуючи досягнення певних цілей системи.

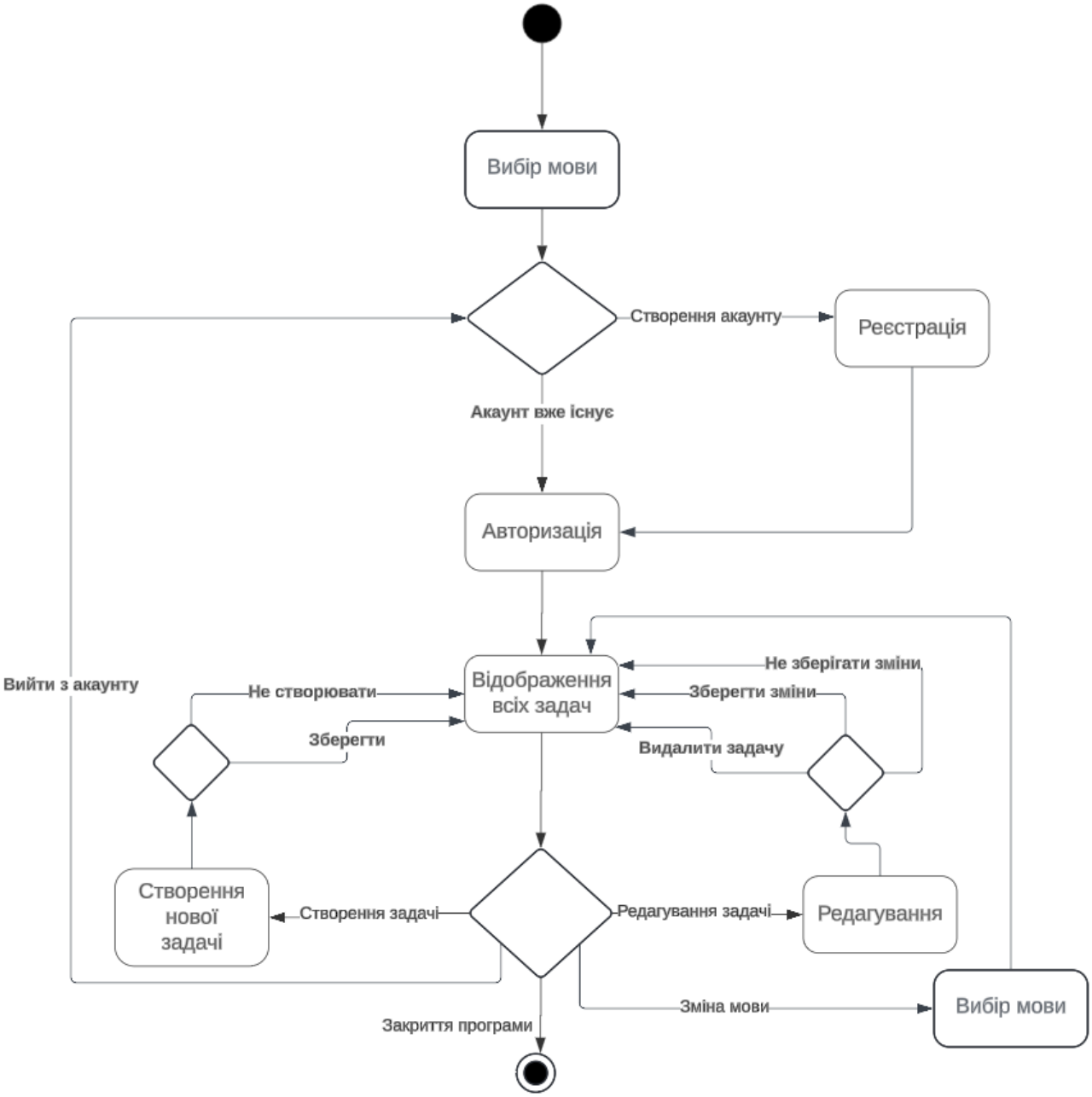


Рис. 2.2 Діаграма діяльності

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Засоби та інструменти розробки

3.1.1 Мова C#

C# [15]– це об'єктно-орієнтована[16] мова програмування, розроблена корпорацією Microsoft. Вона підтримує всі популярні платформи, такі як Windows, Mac OS і Android. Однією з найважливіших особливостей мови є автоматичне управління пам'яттю та забезпечення безпеки програм за допомогою вбудованих механізмів, наприклад, garbage collector. Цей механізм аналізує програму під час виконання та звільняє пам’ять, очищуючи невикористовувані дані. Крім того, C# має велику кількість допоміжних бібліотек, які спрощують і прискорюють процес розробки програмного забезпечення.

Основні переваги C#:

1. Об'єктно-орієнтований підхід: C# базується на об'єктно-орієнтованій парадигмі, що дозволяє організовувати код у вигляді класів і об'єктів, сприяючи модульності, повторному використанню і масштабованості програм.
2. Управління пам'яттю: Мова використовує автоматичне управління пам'яттю (Garbage Collection), що запобігає проблемам з ручним виділенням і звільненням пам'яті.
3. Безпека типів: C# має строгу типізацію, яка перевіряє правильність використання типів даних на етапі компіляції, допомагаючи уникнути помилок під час виконання програми.
4. Підтримка багатопоточності: C# включає вбудовану підтримку багатопоточного програмування, що дозволяє створювати програми, які можуть працювати з кількома потоками одночасно, підвищуючи ефективність роботи додатків.
5. Велика стандартна бібліотека та Nuget: C# має широку стандартну бібліотеку класів, яка забезпечує широкий набір функцій і інструментів для

розробки додатків, включаючи роботу з мережевими протоколами, базами даних, графічними елементами і багато іншого. Крім того, існує велика кількість пакетів Nuget, які легко додаються в проєкт.

6. Інтеграція з платформою .NET: C# є основною мовою програмування платформи .NET, що дозволяє розробляти додатки для різних операційних систем, таких як Windows, MacOS і Linux [17].

3.1.2 .NET Core

.NET Core[18] – це безкоштовна платформа з відкритим вихідним кодом для розробки кросплатформних додатків. Цей фреймворк широко використовується для створення десктопних додатків, веб-додатків, мікросервісів, проектів SaaS тощо. .NET Core підтримує розробку додатків на мовах програмування C#, Visual Basic і F#. Для роботи з платформою .NET Core використовуються такі інтегровані середовища, як Visual Studio і JetBrains Rider. Це вдосконалена версія .NET Framework, яка поступово виходить з ужитку.

3.1.3 ASP.NET Core

ASP.NET Core [19]– це технологія від Microsoft, призначена для створення різноманітних веб-додатків і веб-серверів. Вона має власний легкий контейнер для впровадження залежностей і може працювати незалежно від кросплатформенного середовища .NET Core на Windows, Mac OS і Linux. ASP.NET Core є відкритим фреймворком, доступним на GitHub, що робить його зручним для розробки веб-сервісів, які повинні постійно приймати, обробляти та відправляти дані. Це також дозволяє зручно розгортати розроблені сервіси на серверах, налаштовуючи підключення до інших компонентів, таких як бази даних, брокери повідомлень і кеш-сховища.

Основні особливості ASP.NET Core:

1. Мультиплатформність: Технологія дозволяє розробляти веб-додатки для різних платформ, включаючи Windows, macOS і Linux, що робить можливим створення додатків, які працюють на різних серверних платформах.

2. Ефективність роботи: ASP.NET Core забезпечує швидку обробку запитів і відповідей завдяки оптимізації та використанню асинхронної обробки.

3. Підтримка API: Фреймворк має вбудовану підтримку створення веб-сервісів і API, що дозволяє легко створювати REST API, використовуючи вбудовані інструменти та атрибути маршрутизації, що спрощує розробку комунікації між мікросервісами.

4. Безпека: ASP.NET Core включає механізми для забезпечення безпеки додатків, такі як аутентифікація, авторизація та захист від атак.

5. Інструменти розробки: ASP.NET Core інтегрується з різними середовищами розробки, такими як Visual Studio і JetBrains Rider, надаючи потужні інструменти для написання, відладки і тестування коду.

3.1.4. MySQL

MySQL – це популярна система управління реляційними базами даних з відкритим вихідним кодом. Вона широко використовується для зберігання і управління даними в різних додатках, включаючи веб-додатки, корпоративні системи та мобільні додатки. MySQL[20] підтримує SQL (Structured Query Language) для доступу та управління даними, що робить її потужним інструментом для розробки баз даних.

Основні переваги MySQL:

1. Висока продуктивність: MySQL оптимізована для швидкої обробки запитів і забезпечує високу продуктивність навіть при великих обсягах даних.

2. Надійність: MySQL забезпечує високу надійність і цілісність даних завдяки підтримці транзакцій та різних механізмів реплікації.

3. Гнучкість: MySQL підтримує різні типи таблиць і механізми зберігання даних, що дозволяє розробникам вибирати найбільш підходящі рішення для своїх потреб.

4. Масштабованість: MySQL легко масштабується, що дозволяє обробляти великі обсяги даних і підтримувати високе навантаження на систему.

5. Інтеграція: MySQL легко інтегрується з різними мовами програмування та фреймворками, включаючи PHP, Java, Python та інші.

3.1.5. DBeaver

DBeaver – це універсальний інструмент для роботи з базами даних, який підтримує різні системи управління базами даних, включаючи MySQL, PostgreSQL, Oracle, SQL Server і багато інших. DBeaver пропонує зручний графічний інтерфейс, що дозволяє розробникам та адміністраторам баз даних легко керувати даними, виконувати SQL-запити, створювати та змінювати структури баз даних.

Основні особливості DBeaver:

1. Підтримка різних баз даних: DBeaver підтримує роботу з великою кількістю різних систем управління базами даних, що робить його універсальним інструментом для розробників.

2. Зручний інтерфейс: Інструмент пропонує інтуїтивно зрозумілий графічний інтерфейс, що спрощує виконання SQL-запитів, управління даними та налаштування баз даних.

3. Інтеграція з іншими інструментами: DBeaver легко інтегрується з іншими інструментами для розробки та управління базами даних, такими як Git, JIRA, Jenkins та інші.

4. Розширюваність: DBeaver підтримує плагіни та розширення, що дозволяє додавати нові функції та адаптувати інструмент до своїх потреб.

5. Безкоштовність і відкритий код: DBeaver є безкоштовним інструментом з відкритим вихідним кодом, що робить його доступним для широкого кола користувачів.

3.1.6. WinForms

WinForms (Windows Forms) – це бібліотека класів від Microsoft, яка використовується для створення графічних інтерфейсів користувача (GUI) для додатків, що працюють на платформі Windows. WinForms є частиною платформи .NET і забезпечує розробникам можливість швидко створювати потужні та інтерактивні додатки з графічним інтерфейсом.

Основні особливості WinForms:

1. Легкість використання: WinForms надає зручний і зрозумілий інтерфейс для розробки GUI-додатків, що робить його доступним як для початківців, так і для досвідчених розробників.
2. Широкий набір компонентів: WinForms включає великий набір вбудованих компонентів, таких як кнопки, текстові поля, таблиці, списки та інші елементи управління, що дозволяє створювати різноманітні інтерфейси користувача.
3. Подієво-орієнтована модель програмування: WinForms використовує подієво-орієнтовану модель, що дозволяє розробникам легко обробляти взаємодію користувача з додатком шляхом підписки на події компонентів.
4. Інтеграція з Visual Studio: WinForms інтегрується з Visual Studio, надаючи потужні інструменти для розробки, відладки та тестування додатків. Візуальний дизайнер в Visual Studio дозволяє розробникам створювати інтерфейси шляхом простого перетягування компонентів на форму.
5. Розширюваність: WinForms підтримує створення власних компонентів і елементів управління, що дозволяє розробникам розширювати функціональність своїх додатків відповідно до специфічних вимог.
6. Сумісність з іншими технологіями .NET: WinForms легко інтегрується з іншими технологіями платформи .NET, такими як ADO.NET для роботи з базами даних, WCF для створення сервісів і багато інших, що робить його універсальним інструментом для розробки настільних додатків.

WinForms залишається популярним вибором для створення простих і середньо складних настільних додатків, що працюють на Windows, завдяки своїй простоті, ефективності та широким можливостям інтеграції.

3.2 Структура бази даних

Для створення, редагування, відслідковування та адміністрування бази даних було використано програмне рішення DBeaver. Для бази даних було обрано технологію MySQL через її високу поширеність та підтримку. База даних має наступні таблиці:

1. **Таблиця users_list.** Ця таблиця використовується для зберігання списку користувачів та їх паролів. Використовуючи захешований пароль збережений в цій таблиці застосунок проводить валідацію(співставлення) паролю, який введено користувачем при авторизації, до його захешованого варіанту.
2. **Таблиця ToDoData.** Ця таблиця використовується для зберігання усіх задач усіх користувачів. Для прив'язки задач до користувачів використовується логін користувача, завдяки якому відбувається фільтрація задач по логіну авторизованого у застосунку користувача.

На рисунку 3.1 зображено структуру вищенаведених таблиць:

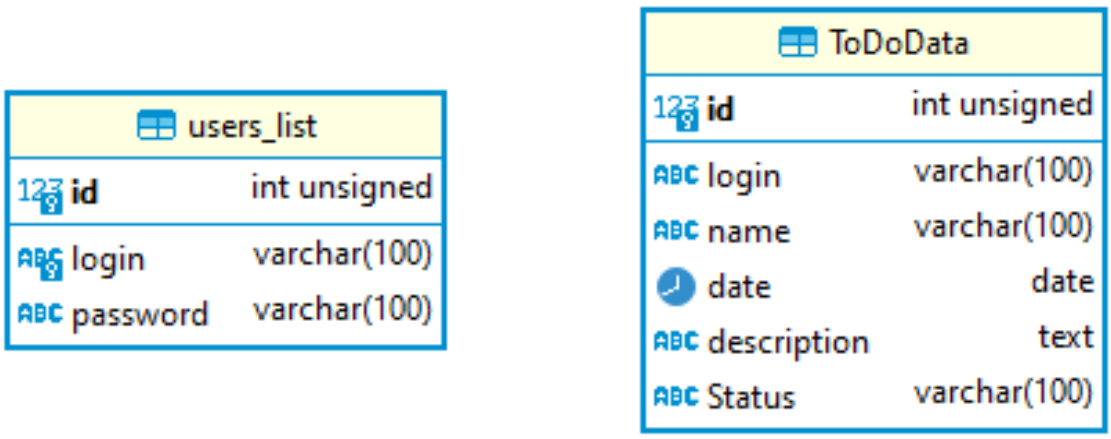


Рис. 3.1 Структура таблиць ToDoData та users_list

3.3 Підключення до бази даних MySQL

Для підключення до бази даних було створено клас DataBase який відповідає за підключення застосунка до бази даних. У класі було використано бібліотеку підключення MySQLConnector а саме файл MySQL.Data. Код цього класу зображено на рисунку 3.2.

```
1   using MySql.Data.MySqlClient;
2   using System;
3   using System.Collections.Generic;
4   using System.Linq;
5   using System.Text;
6   using System.Threading.Tasks;
7
8   namespace Test_project
9   {
        16 references
10      internal class DataBase
11      {
12          MySqlConnection connection = new("server=todoplatform-todoplatform.c.aivencloud.com;" +
13              "port=14884;" +
14              "username=avnadmin;" +
15              "password=AVNS_WepTmB-DbOfQAgdaJUR;" +
16              "database=users");
17
18
            4 references
19          public void openConnection()
20          {
21              if (connection.State == System.Data.ConnectionState.Closed)
22              {
23                  connection.Open();
24              }
25          }
            4 references
26          public void closeConnection()
27          {
28              if (connection.State == System.Data.ConnectionState.Open)
29              {
30                  connection.Close();
31              }
32          }
33
            9 references
34          public MySqlConnection getConnection()
35          {
36              return connection;
37          }
38      }
39  }
40
```

Рис. 3.2 Клас DataBase

Пояснення коду:

1. MySql.Data.MySqlClient – бібліотека, яка забезпечує роботу з MySQL базами даних.

2. MySqlConnection connection – об'єкт, який зберігає параметри підключення до бази даних, а саме: інформацію про сервер, порт, ім'я користувача, пароль та базу даних.

3. openConnection – перевіряє, чи стан підключення є закритим. Якщо так, відкриває підключення до бази даних.

4. closeConnection – перевіряє, чи стан підключення є відкритим. Якщо так, закриває підключення до бази даних.

5. getConnect – повертає об'єкт MySqlConnection, що дозволяє іншим частинам програми отримати доступ до підключення і використовувати його для виконання SQL-запитів.

3.4 Опис головної сторінки

Головна сторінка представляє собою головну форму, на якій буде розміщений основний функціонал застосунку. Головна сторінка може також бути названа формою відображення всіх задач. На рисунку 3.3 зображено пусту форму головної сторінки без списків задач з англійською локалізацією та з українською на рисунку 3.4.

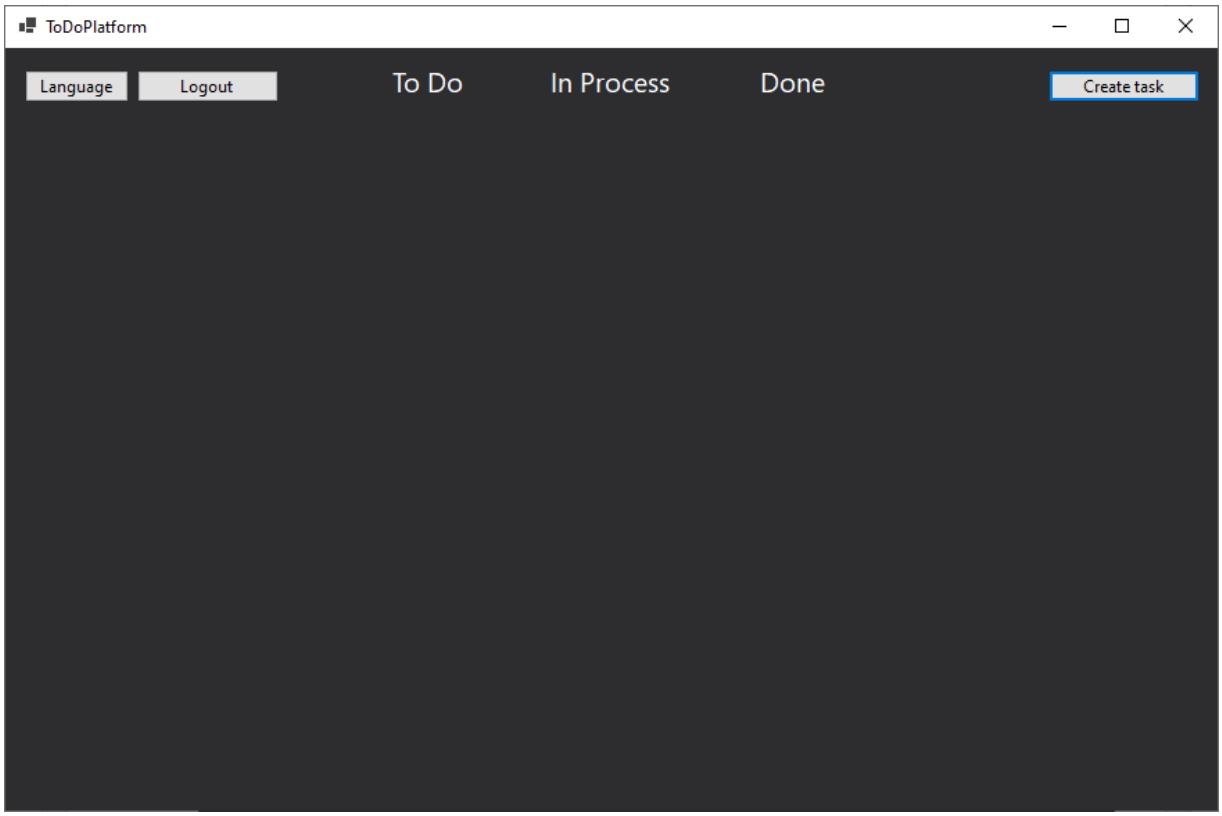


Рис. 3.3 Вікно головної сторінки з англійською локалізацією

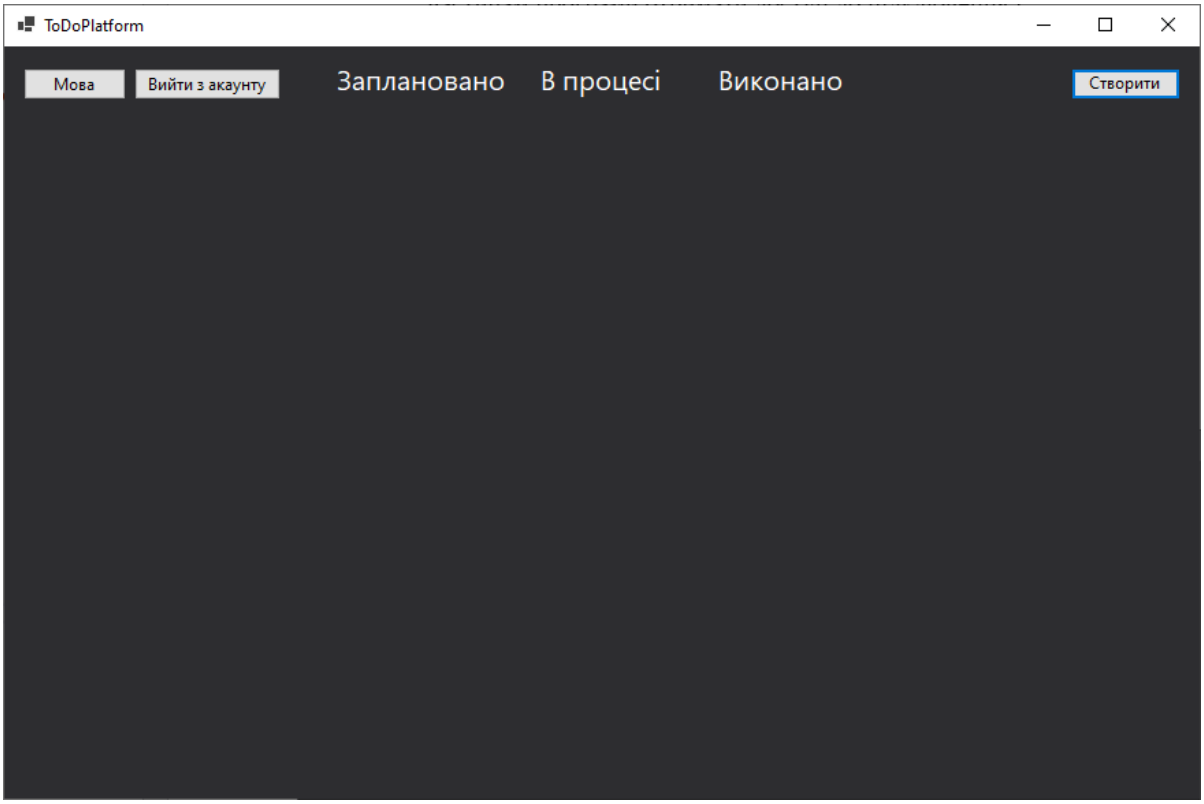


Рис. 3.4 Вікно головної сторінки з українською локалізацією

На рисунках 3.3 та 3.4 зображено основні елементи які створені та розміщенні на формі статично, а саме:

- Кнопка зміни мови

Викликає форму для вибору мови та по вибору мови оновлює в системі застосунку вибрану користувачем мову.

- Кнопка виходу з акаунту

Видаляє збережений в системі застосунку логін авторизованого користувача та переходить на форму авторизації.

- Кнопка створення нової задачі

Переходить на форму створення нової задачі.

- Мітка «Заплановано»

При натисненні на мітку відбувається фільтрація та оновлення списку задач за статусом виконання. Вмикає відображення лише задач зі статусом Заплановано.

- Мітка «В процесі»

При натисненні на мітку відбувається фільтрація та оновлення списку задач за статусом виконання. Вмикає відображення лише задач зі статусом В процесі.

- Мітка «Виконано»

При натисненні на мітку відбувається фільтрація та оновлення списку задач за статусом виконання. Вмикає відображення лише задач зі статусом Виконано.

Для міток також організоване правило, яке дозволяє перемикати фільтрацію відображення одним кліком на іншу мітку. При повторному натисненні на вже обрану мітку (див. рисунок 3.5), вимикає фільтрацію та перемикає на відображення усіх задач, які належать авторизованому в застосунку користувачу.

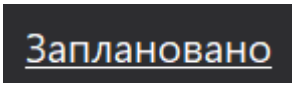


Рис. 3.5 Приклад відображення обраної мітки

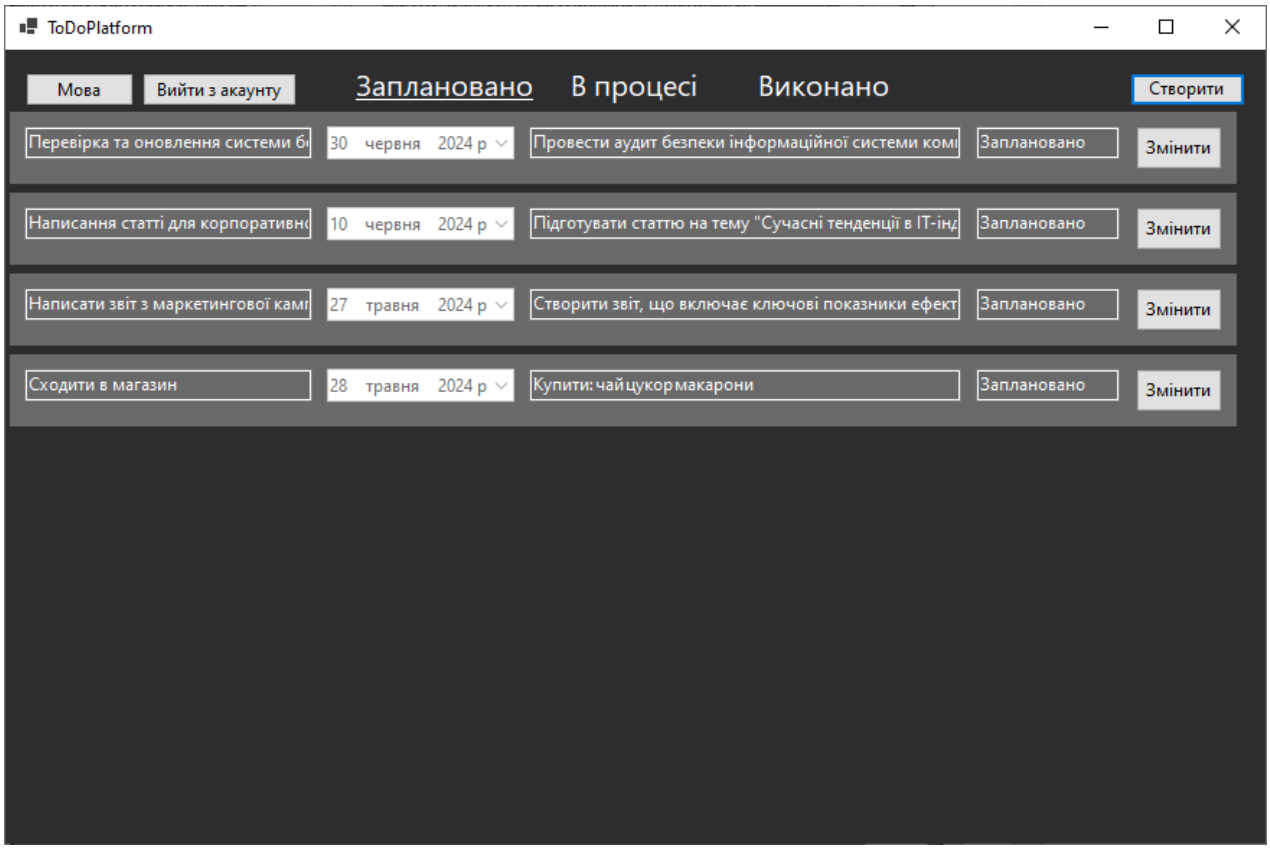


Рис. 3.6 Приклад фільтрації по статусу «Заплановано»

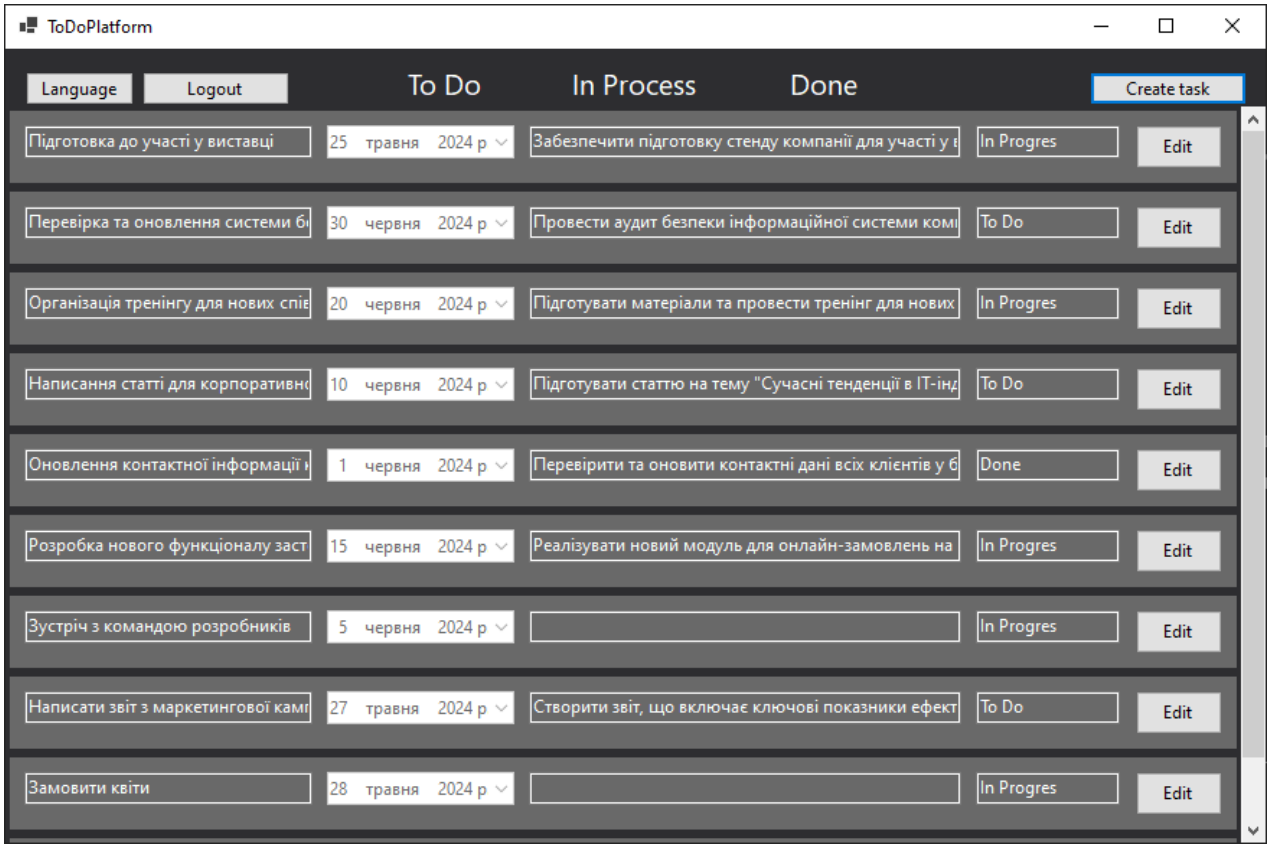


Рис. 3.7 Форма головної сторінки з англійською локалізацією та задачами

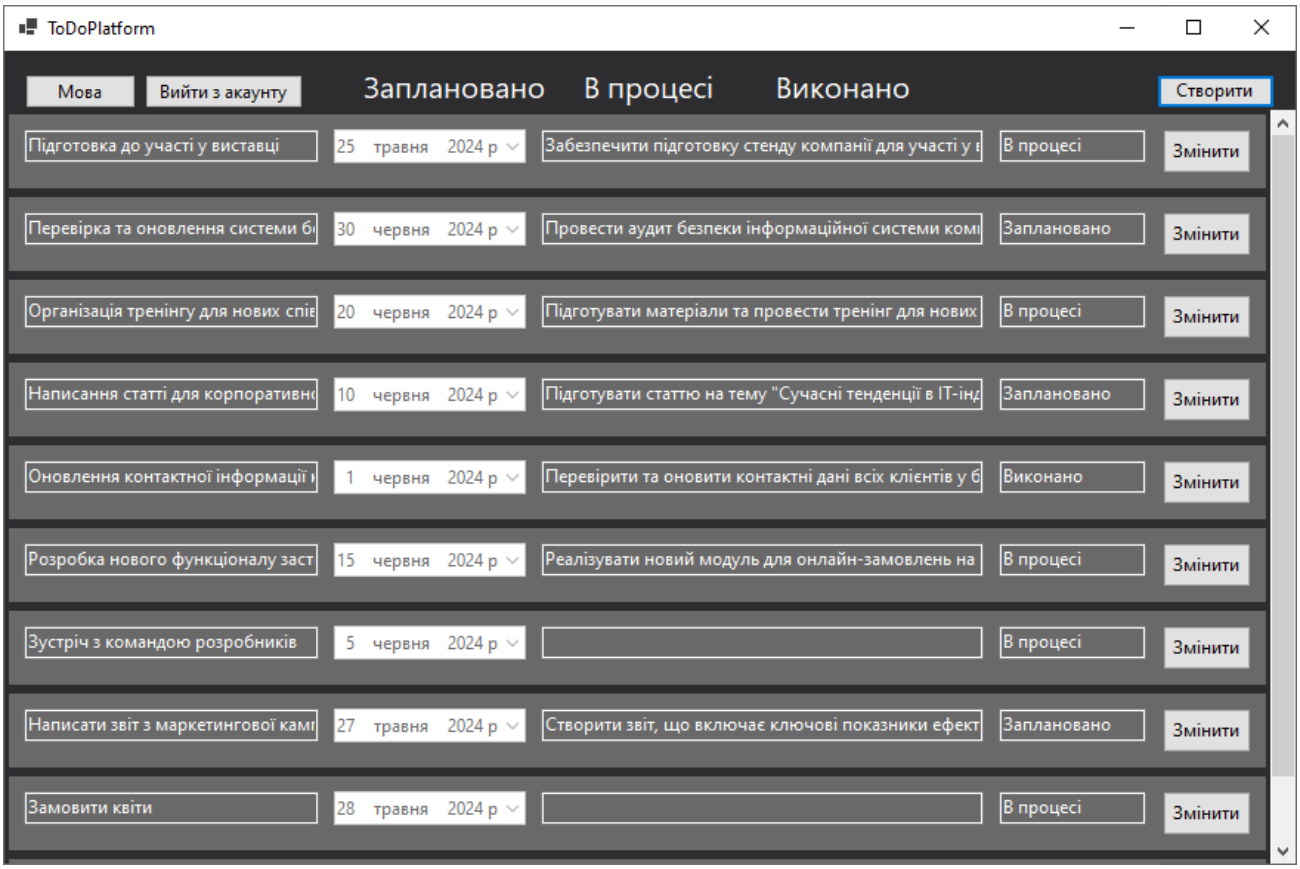


Рис. 3.8 Форма головної сторінки з українською локалізацією та задачами

На рисунках 3.7 та 3.8 зображено головна сторінка з доданими динамічними об’єктами, а саме:

- Панель компонування потоку (FlowLayoutPanel)

Використовується як поле для створення на ньому панелі з задачами. Це полегшує очищення динамічних об’єктів при оновленні екранної форми.

- Панель задачі

Використовується для групування інформації про задачу у один компонент та відображення у горизонтальному порядку. Розміщується на FlowLayoutPanel. Містить в собі наступні об’єкти:

1. Текстове поле з назвою

Містить в собі назву задачі.

2. Вибірник дати та часу (DateTimePicker)

Використовується для позначення дати дедлайну.

3. Текстове поле з описом задачі

Містить в собі опис задачі.

4. Текстове поле зі статусом задачі

Містить в собі статус задачі.

5. Кнопка редагування задачі

Використовується для переходу на форму редагування та видалення задачі.

Кількість динамічних об’єктів визначається кількістю рядків в таблиці задач, які відповідають умовам запиту до хмарної бази даних, а саме співпадіння логіну користувача та статусу задачі при ввімкненій фільтрації.

3.5 Опис форми реєстрації

Форма реєстрації використовується для створення та внесення акаунта у базу даних. На рисунку 3.9 зображено форму реєстрації.

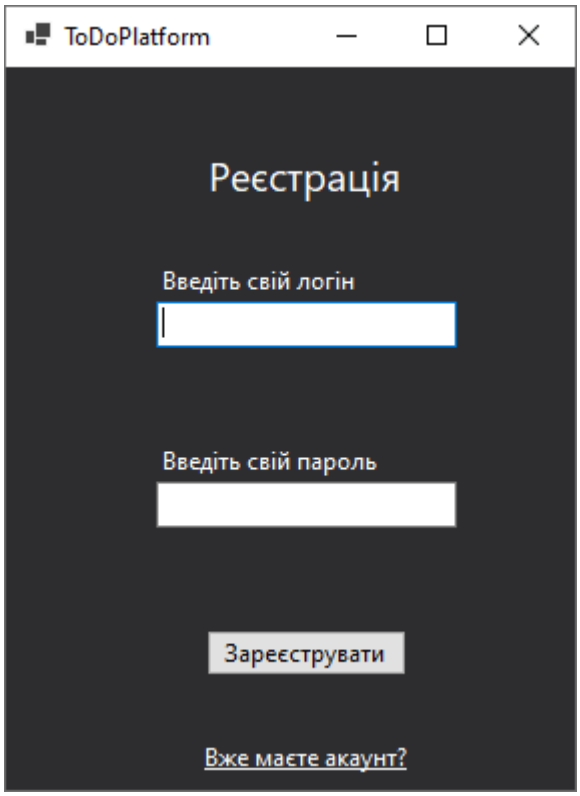


Рис. 3.9 Форма реєстрації в українській локалізації

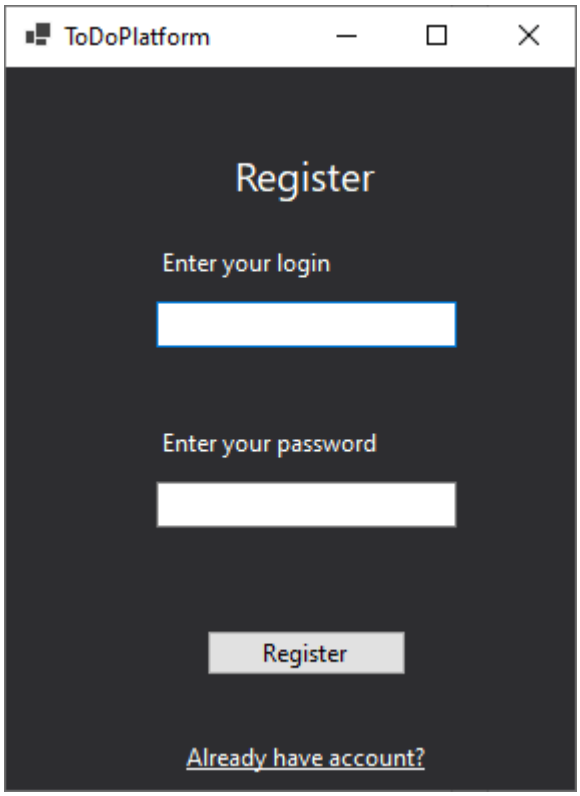


Рис. 3.10 Форма реєстрації в англійській локалізації

На формі розташовані такі об’єкти:

- Поле для логіну

Приймає в себе текст логіну.

- Поле для паролю

Приймає в себе текст паролю. Введені символи візуально змінює на зірочки.

- Кнопка «Зареєструватися»

При натисненні перевіряє наявність введеного логіну та паролю. Якщо логін не зайнятий, створюється акаунт. Для забезпечення безпеки пароль хешується за допомогою бібліотеки BCrypt.Net.

- Текст-посилання на форму авторизації

При натисненні на напис відбувається перехід на форму авторизації.

3.6 Опис форми авторизації

Форма авторизації відповідає за авторизацію користувача та подальший запуск застосунку. На рисунку 3.10 зображено форму авторизації.

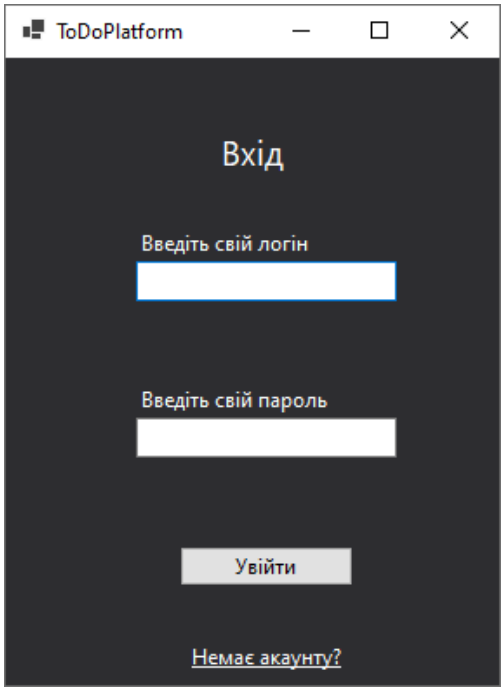


Рис. 3.11 Форма авторизації в українській локалізації

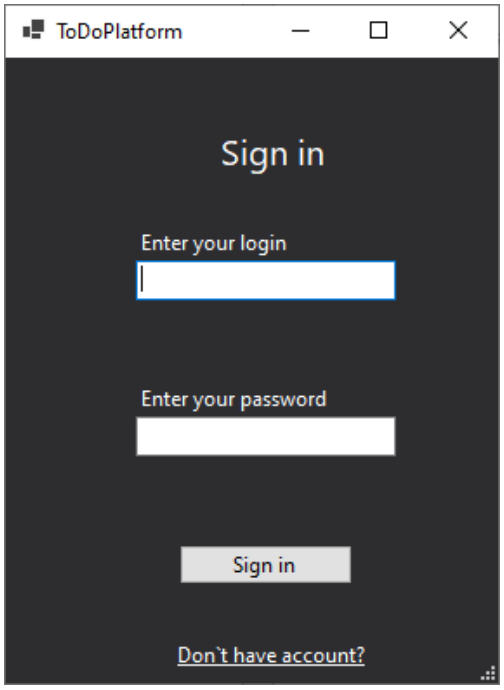


Рис. 3.12 Форма авторизації в англійській локалізації

На формі розташовані такі об’єкти:

- Поле для логіну

Приймає в себе текст логіну.

- Поле для паролю

Приймає в себе текст паролю. Введені символи візуально змінює на зірочки.

- Кнопка «Увійти»

При натисненні перевіряє наявність введеного логіну та паролю. При наявності такого логіну у таблиці бази даних зрівнює захешований пароль з бази даних та введений в поле паролю текст за допомогою верифікації бібліотекою BCrypt.Net. При співпадінні переходить на головну форму застосунку.

- Текст-посилання на форму реєстрації

При натисненні на напис відбувається перехід на форму реєстрації.

Якщо користувач успішно пройшов авторизацію його логін зберігається в застосунку та використовується при усіх подальших запусках застосунку запускаючи одразу головну сторінку. При виході з акаунту з головної сторінки цей логін видаляється з пам’яті застосунку до нової авторизації.

3.7 Опис форми створення нової задачі

Форма створення нових задач створює та зберігає задачі у базі даних.

На формі яка зображена на рисунку 3.11 зображено такі об’єкти:

- Текстове поле для назви

Містить в собі назву задачі.

- Вибірник дати та часу (DateTimePicker)

Використовується для вибору дати дедлайну.

- Текстове поле для опису задачі

Містить в собі опис задачі.

- Випадний список зі статусом задачі

Список з 3 елементів які обираються.

- Кнопка створення задачі

Використовується для збереження задачі та переходу на основну сторінку.

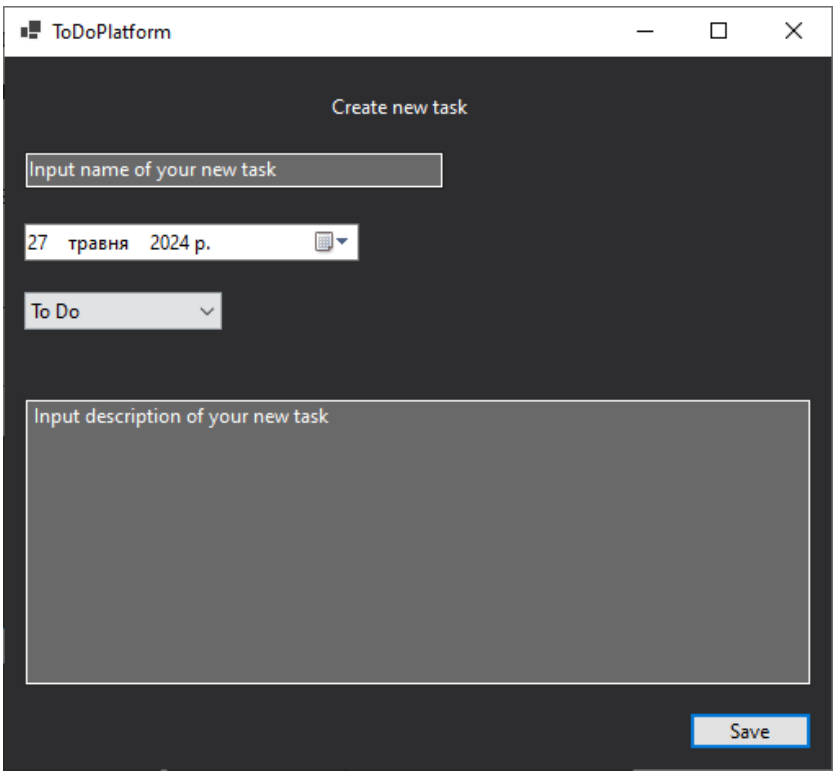


Рис. 3.13 Форма створення нової задачі в англійській локалізації

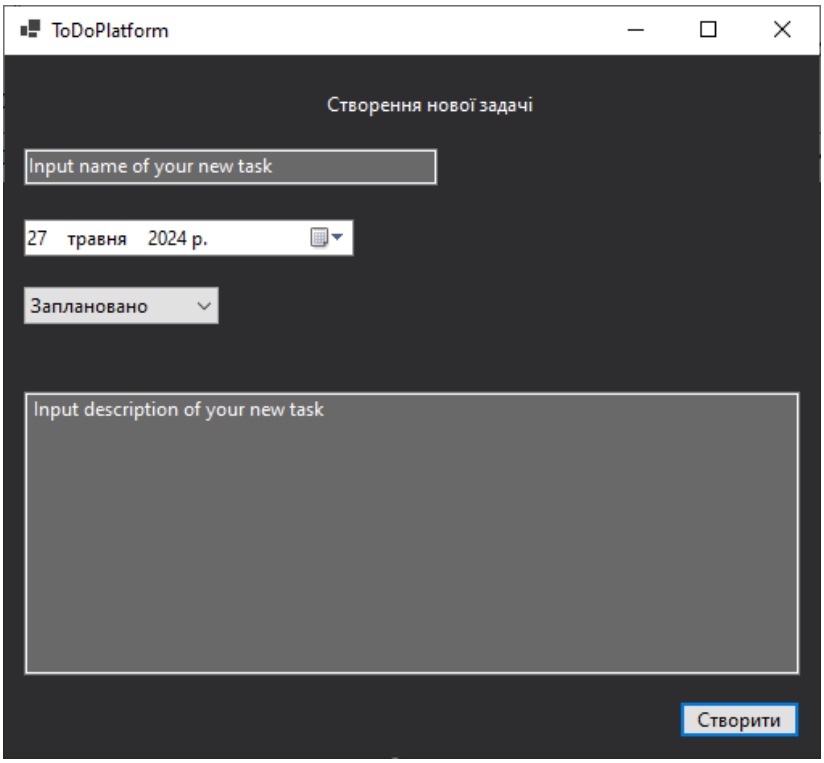


Рис. 3.14 Форма створення нової задачі в українській локалізації

Для вибору дати дедлайну стоїть обмеження, що можна обрати лише не раніше місяця який є сьогодні. Для збереження стоїть обмеження, що створити задачу без вказання назви не можна, але можна залишати пустим поле для опису.

Для повернення до головної сторінки потрібно закрити форму створення.

3.8 Опис форми редагування задачі

Форма редагування задачі отримує інформацію про задачу за допомогою передачі унікального ідентифікатора задачі при створенні нової форми редагування. Далі, за допомогою цього id, застосунок звертаючись до бази даних отримує інформацію про задачу та при завантаженні форми заповнює поля отриманою інформацією. Приклад форми зображено на рисунку 3.12.

На формі розташовані такі об’єкти:

- Текстове поле з назвою

Містить в собі назву задачі.

- Вибірник дати та часу (DateTimePicker)

Використовується для відображення дати дедлайну.

- Текстове поле для опису задачі

Містить в собі опис задачі.

- Випадний список зі статусом задачі

Список з 3 елементів які обираються.

- Кнопка збереження задачі

Використовується для збереження задачі та переходу на основну сторінку.

- Кнопка видалення задачі

При натисненні задача буде видалена з бази даних. Відбудеться перехід до головної сторінки.

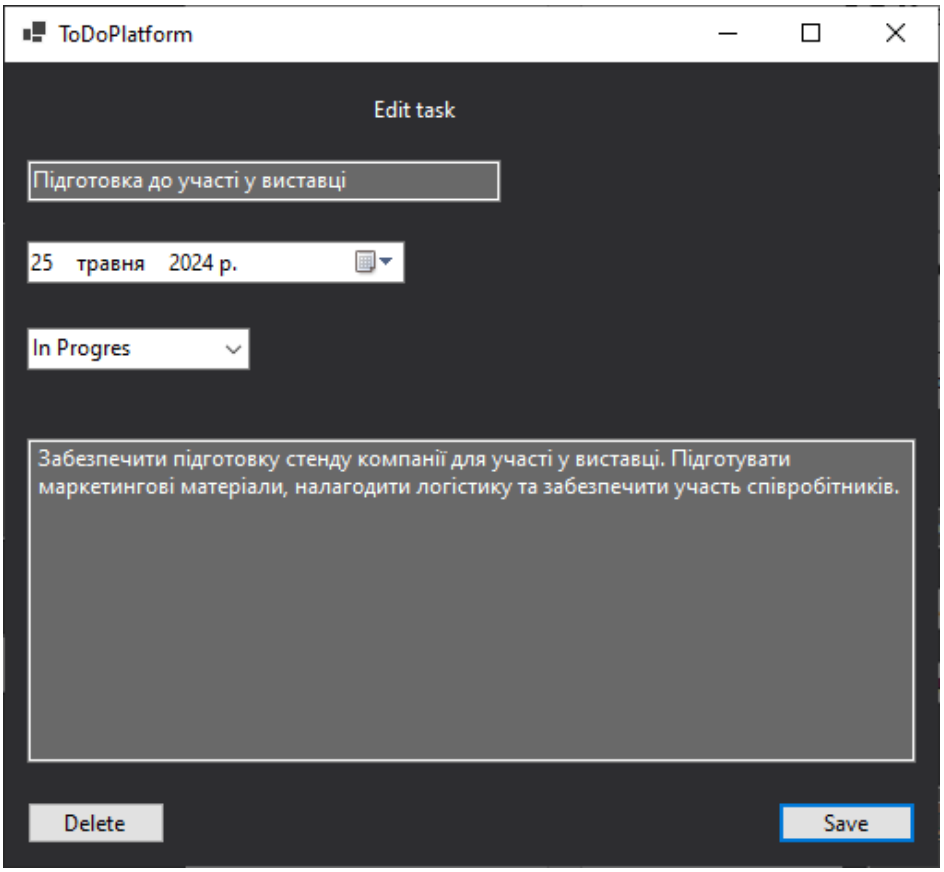


Рис. 3.15 Форма для редагування задачі в англійській локалізації

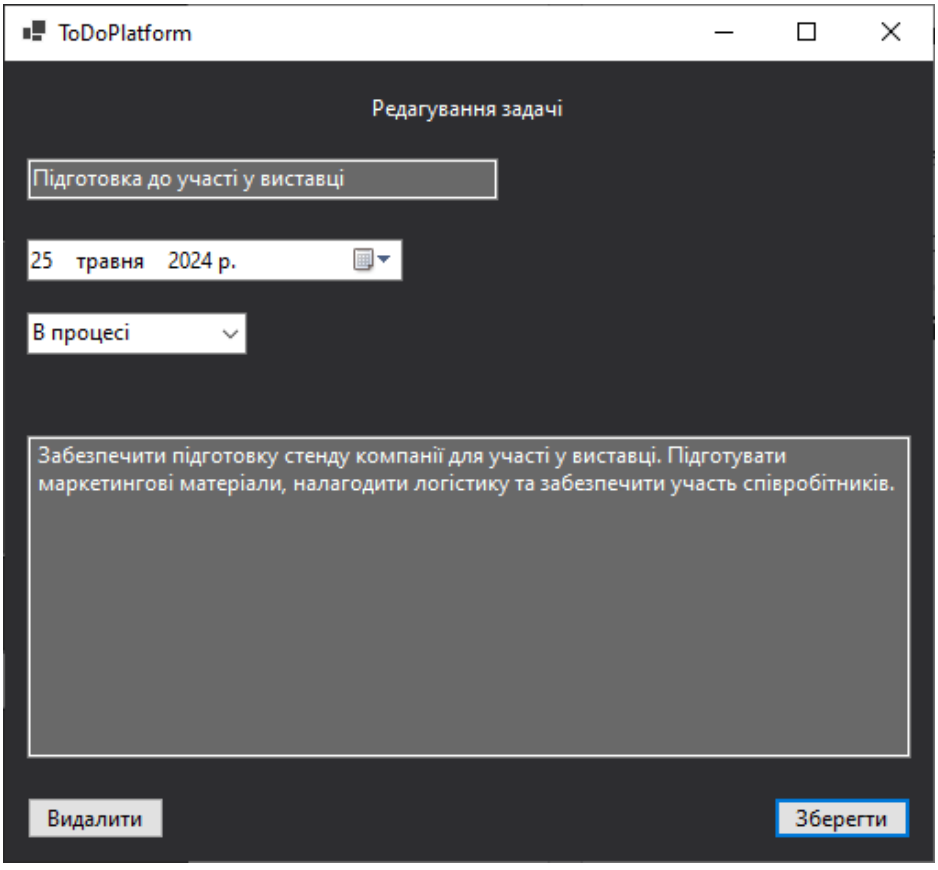


Рис. 3.16 Форма для редагування задачі в українській локалізації

Для скасування змін потрібно закрити форму редагування. При цьому відбудеться перехід до головної сторінки.

3.8 Опис форми вибору мови

Ця форма запускається при першому запуску застосунку та при зміні мови з головної сторінки застосунку. Вибрана мова зберігається в глобальних налаштуваннях застосунку та переноситься між сесіями. На рисунку 3.13 зображено форму вибору мови.

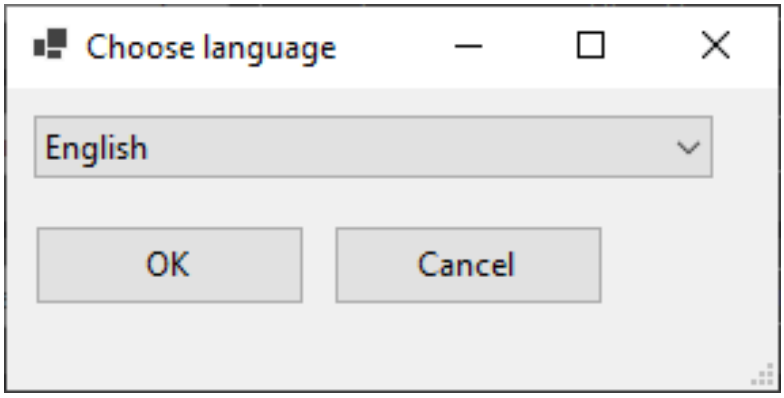


Рис. 3.17 Форма вибору мови

3.9 Тестування застосунку

Юніт-тестування відіграє ключову роль у забезпеченні якості програмного забезпечення, перевіряючи окремі частини коду, такі як функції, методи або класи, для підтвердження їхньої коректної роботи. Метою юніт-тестів є виявлення помилок у найменших функціональних одиницях застосунку, що дозволяє розробникам впевнитися у правильній роботі компонентів та швидко знаходити і виправляти помилки на ранніх стадіях розробки.

Для початку юніт-тестування необхідно підготувати середовище і налаштувати залежності, використовуючи спеціалізовані фреймворки або бібліотеки, як-от NUnit для C#. Кожен компонент програми має свої відповідні тести, які перевіряють його функціональність.

Структура юніт-тесту зазвичай включає такі етапи:

1. Підготовка (Setup): Підготовка до тесту, яка передбачає створення необхідних об'єктів, ініціалізацію змінних та налаштування середовища.

2. Виконання (Execution): Тестування компонента, що зазвичай включає виклик функції або методу.

3. Перевірка (Assertion): Перевірка результату виконання компонента за допомогою операторів перевірки, які порівнюють отриманий результат з очікуваним.

4. Прибирання (Cleanup): Очищення після виконання тесту, що включає звільнення ресурсів, видалення тимчасових об'єктів та скасування змін у тестовому середовищі.

На рисунку 3.14 зображено один з тестів для перевірки підключення до баз даних.

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using MySql.Data.MySqlClient;
using Mysqlx.Crud;
using Test_project;

[TestClass]
0 references
public class DataBaseTests
{
    private DataBase db;

    [TestInitialize]
    0 references
    public void Setup()
    {
        db = new DataBase();
    }

    [TestMethod]
    0 references
    public void Test_OpenConnection()
    {
        db.openConnection();
        Assert.AreEqual(System.Data.ConnectionState.Open, db.getConnection().State);
    }

    [TestMethod]
    0 references
    public void Test_CloseConnection()
    {
        db.openConnection();
        db.closeConnection();
        Assert.AreEqual(System.Data.ConnectionState.Closed, db.getConnection().State);
    }

    [TestMethod]
    0 references
    public void Test_GetConnection()
    {
        MySqlConnection connection = db.getConnection();
        Assert.IsNotNull(connection);
    }

    [TestCleanup]
    0 references
    public void Cleanup()
    {
        if (db.getConnection().State == System.Data.ConnectionState.Open)
        {
            db.closeConnection();
        }
    }
}
```

Рис. 3.18 Приклад юніт-тесту застосунку

ВИСНОВКИ

Під час розробки додатку ToDo, було проведено детальний аналіз існуючих аналогів, зокрема Taskwarrior та Simple Sticky Notes. При проведені аналізу було визначено основні функції та характеристики майбутнього застосунку, які можуть задовольнити потреби користувачів і забезпечити високу конкурентоспроможність.

На основі цього аналізу були сформульовані вимоги до функціоналу, інтерфейсу користувача, продуктивності, безпеки та інших важливих аспектів. Вимоги включають можливість легко створювати, редагувати та видаляти списки справ і їх дедлайни, а також можливість фільтрувати списки за статусом виконання та інші.

В процесі розробки вивчалися та оцінювалися різні інструменти та технології, серед яких мова програмування C#, середовище розробки Visual Studio 2022, система керування базами даних DBeaver та MySQL, а також бібліотека WinForms. Було прийнято рішення використовувати ці технології, оскільки вони забезпечують необхідну функціональність та стабільність для реалізації проекту.

Створення та налаштування бази даних було проведено для забезпечення надійного зберігання інформації про завдання та їх параметри. Було використано MySQL для організації структури даних, забезпечуючи ефективне управління та доступ до інформації.

Було розроблено застосунок відповідно до розробленої архітектури та вимогам. Для кожної з форми було розроблено та реалізовано унікальний функціонал для ефективного вирішення поставлених вимог.

Проведено тестування застосунку з метою виявлення та виправлення помилок та недоліків. Було застосовано функціональне та юніт- тестування для перевірки коректності роботи всіх компонентів системи та їх взаємодії. Виявлені проблеми були оперативно виправлені, що дозволило забезпечити високу якість кінцевого продукту.

Крім того, було приділено увагу питанням безпеки, щоб забезпечити захист даних користувачів та запобігти несанкціонованому доступу до системи. Було впроваджено механізми аутентифікації та авторизації користувачів, а також забезпечено шифрування конфіденційної інформації.

Таким чином, розроблений додаток ToDo є зручним і функціональним інструментом для організації завдань і управління часом, що відповідає сучасним вимогам користувачів і забезпечує високу продуктивність та безпеку.

ПЕРЕЛІК ПОСИЛАНЬ

1. Клусенко Д.М. Залива В.В. Порівняльний аналіз хмарних баз даних для покращення продуктивності веб сервісів. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях». 24 квітня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024. С.114-117.

2. MARDAN, Azat; MARDAN, Azat. Todo App. Pro Express. js, 2014, 223-248.

3. Taskwarrior - Color Themes - Taskwarrior. Taskwarrior. URL: https://taskwarrior.org/docs/themes/ (дата звернення: 25.04.2024).

4. Screenshots - Simple Sticky Notes. Simple Sticky Notes - Free Sticky Notes Software. URL: https://www.simplestickynotes.com/screenshots (дата звернення: 25.04.2024).

5. PRICE, Mark J. C# 9 and. NET 5–Modern Cross-Platform Development: Build intelligent apps, websites, and services with Blazor, ASP. NET Core, and Entity Framework Core using Visual Studio Code. Packt Publishing Ltd, 2020.

6. DBeaver Community | Free Universal Database Tool. DBeaver Community | Free Universal Database Tool. URL: https://dbeaver.io/ (дата звернення: 25.04.2024).

7. BIN UZAYR, Sufyan. Mastering MySQL for Web: A Beginner's Guide. CRC Press, 2022.

8. High-Performance .NET MySQL Driver - MySqlConnector. High-Performance .NET MySQL Driver - MySqlConnector. URL: https://mysqlconnector.net/ (дата звернення: 25.04.2024).

9. SELLS, Chris; GEHTLAND, Justin. Windows forms programming in Visual Basic. NET. Addison-Wesley Professional, 2004.

10. ПОПОВА, В. Р.; БОБРІКОВА, І. С. Шифрування даних як один з методів захисту інформації. СТАН, ДОСЯГНЕННЯ ТА ПЕРСПЕКТИВИ ІНФОРМАЦІЙНИХ СИСТЕМ І ТЕХНОЛОГІЙ, 2022, 70.

11. МІЩЕНКО, Володимир Іванович. ВПЛИВ ЦИФРОВИХ ТЕХНОЛОГІЙ НА РОЗВИТОК ПРОЦЕСІВ ЕКОНОМІЧНОЇ ГЛОБАЛІЗАЦІЇ ТА ЛОКАЛІЗАЦІЇ. ББК 65.9 (4Укр) 26, 2022, 27.

12. МАЗУР, Олена. Локалізація як явище перекладу. 2021

13. ARIFIN, Mohammad Nazir; SIAHAAN, Daniel. Structural and Semantic Similarity Measurement of UML Use Case Diagram. Lontar Komputer: Jurnal Ilmiah Teknologi Informasi, 2020, 11.2: 88.

14. ABBAS, Messaoud, et al. Formal modeling and verification of UML Activity Diagrams (UAD) with FoCaLiZe. Journal of Systems Architecture, 2021, 114: 101911.

15. NAGEL, Christian. Professional C# and. Net. John Wiley & Sons, 2021.

16. CIESLA, Robert; CIESLA, Robert. Object-Oriented Programming (OOP). Programming Basics: Getting Started with Java, C#, and Python, 2021, 43-62.

17. C# docs - get started, tutorials, reference. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/uk-ua/dotnet/csharp/tour-of-csharp/ (дата звернення: 25.04.2024).

18. Introduction to .NET - .NET. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/uk-ua/dotnet/core/introduction (дата звернення: 25.04.2024).

19. PRICE, Mark J. C# 10 and. NET 6–Modern Cross-Platform Development: Build apps, websites, and services with ASP. NET Core 6, Blazor, and EF Core 6 using Visual Studio 2022 and Visual Studio Code. Packt Publishing Ltd, 2021.

20. ŠUŠTER, Ivan; RANISAVLJEVIĆ, Tamara. Optimization of MySQL database. Journal of process management and new technologies, 2023, 11.1-2: 141-151.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ
(Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО -КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО -НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка застосунку To Do для складання списків справ та планування їх дедлайнів мовою C# з використанням баз даних

Виконав студент 4 курсу

групи ПД-41
Клусенко Данило Миколайович
Керівник роботи

Доктор філософії, старший викладач кафедри ІПЗ Залива Віталій Вікторович

Київ – 2024

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** - спрощення зберігання, відслідковування справ та їх термінів виконання за допомогою desktop-застосунку, створеного мовоюC#.
- **Об’єкт дослідження** - процес створення та відслідковування поставлених задач.
- **Предмет дослідження** - desktop-застосунок для створення, зберігання та відслідковування списку справ.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Проаналізувати існуючі аналоги та визначити головні функції та характеристики майбутнього застосунку.
2. Сформулювати вимоги до функціоналу, інтерфейсу користувача, продуктивності, безпеки та інших важливих аспектів.
3. Вивчити та оцінити різні інструменти та технології, які можуть бути використані для розробки застосунку включаючи мову програмування, середовище розробки, системи керування базами даних та бібліотеки
4. Розробити інтерфейс користувача, реалізувати функціонал для створення, редагування та видалення списків справ та їх дедлайнів, а також для фільтрації списків справ. Створити та налаштувати базу даних для зберігання інформації.
5. Провести тестування застосунку. Виконати функціональне тестування, виявити та виправити помилки та недоліки.

3

АНАЛІЗ АНАЛОГІВ

	Переваги	Недоліки
Simple Sticky Notes	• Не вимагає реєстрації або встановлення додаткового програмного забезпечення • Кастомізація тексту	• Відсутність можливості встановлювати терміни виконання задач • Відсутність можливості синхронізації задач між пристроями • Відсутня можливість трекінгу виконання задач
TaskWarrior	• Можливість працювати офлайн • Підтримує ієрархічну структуру задач • Система фільтрації та пошуку задач • Функція трекінгу виконання задач	• Для використання та керування потрібно знати спеціальні команди керування • Відсутність графічного інтерфейсу
ToDoPlatform	• Збереження в хмарному сховищі • Присутня українська локалізація • Функція трекінгу виконання задач	• Відсутність можливості створення підзадач

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги

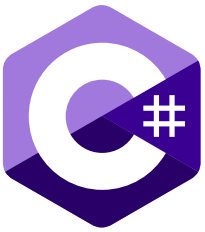
1. Реєстрація та авторизація користувача.
2. Створення нових задач.
3. Відображення задач користувача.
4. Редагування та видалення існуючих задач користувача.
5. Зберігання задач у базі даних.
6. Зміна мови застосунку.

Нефункціональні вимоги

1. Локалізація (Українська, Англійська).
2. Безпека (хешування паролю).

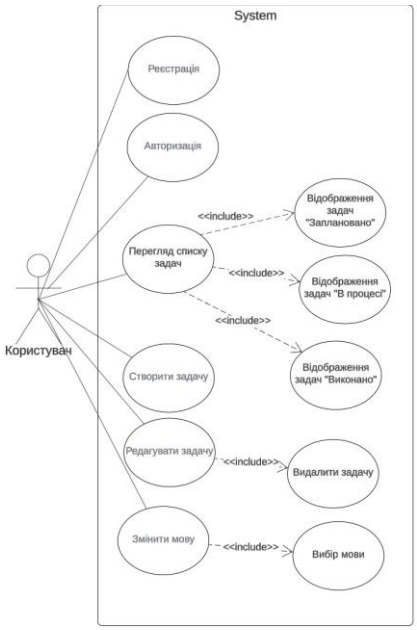
5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ



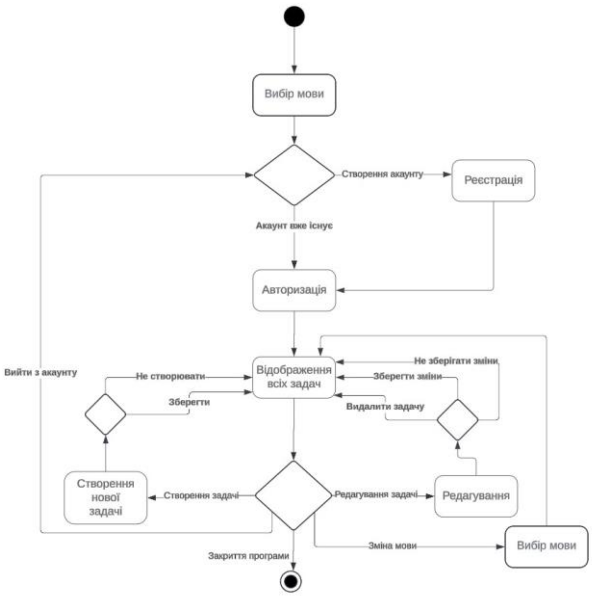
6

Діаграма варіантів використання



7

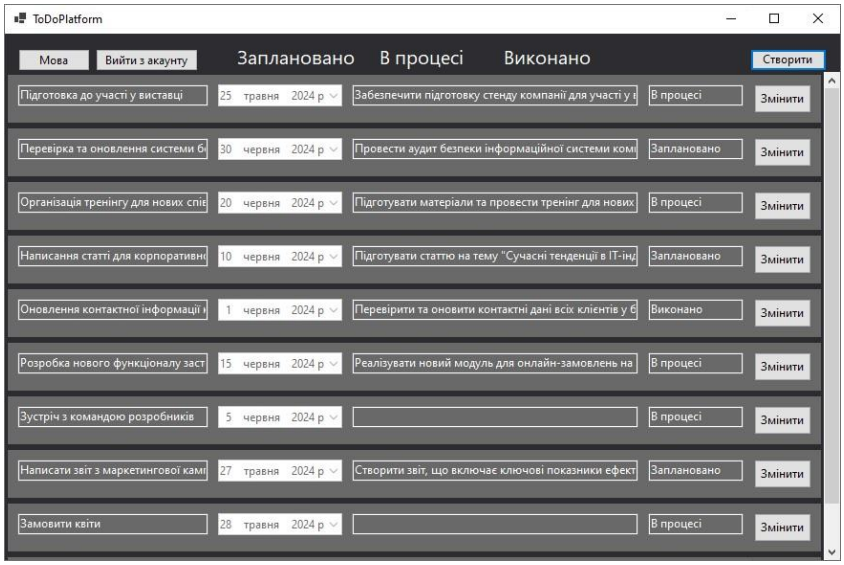
Блок-схема функціонування застосунку



8

ЕКРАННІ ФОРМИ

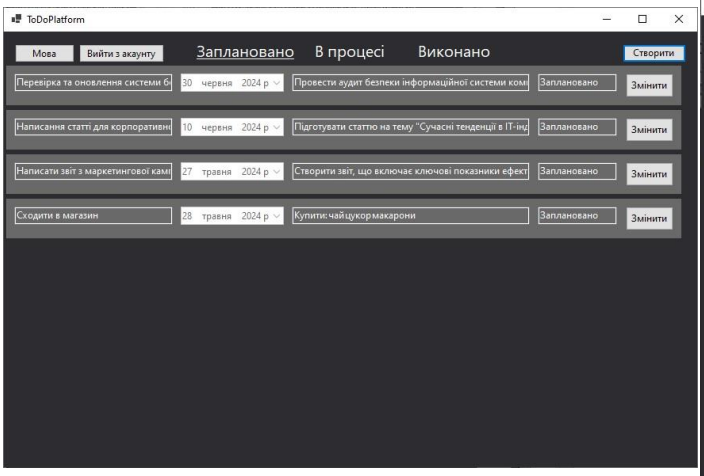
Авторизація



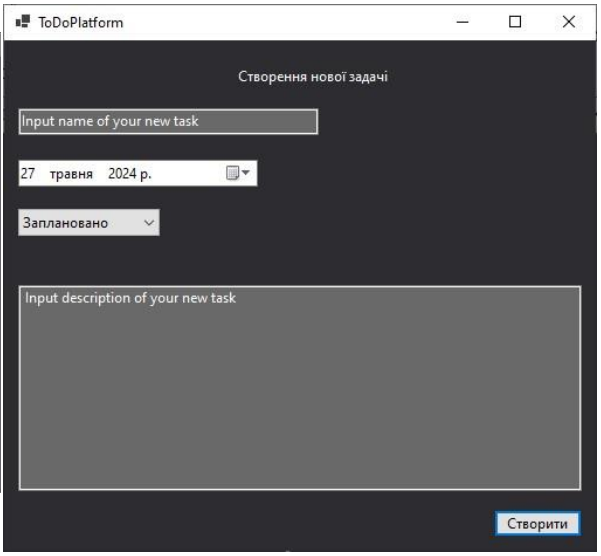
Відображення усіх задач

9

ЕКРАННІ ФОРМИ



Фільтрація по статусу задач



Створення задачі

10

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Клусенко Д. М. Порівняльний аналіз хмарних баз даних для покращення продуктивності веб сервісів. Актуальні проблеми забезпечення інформаційної та кібернетичної безпеки: Матеріали Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях». Збірник тез. 24.04.2024, ДУТ, м. Київ. К.: ДУІКТ, 2024. С. 114.

11

ВИСНОВКИ

1. Під час розробки додатку ToDo, було проведено детальний аналіз існуючих аналогів, зокрема Taskwarrior та Simple Sticky Notes. При проведені аналізу було визначено основні функції та характеристики майбутнього застосунку, які можуть задовольнити потреби користувачів і забезпечити високу конкурентоспроможність.
2. На основі цього аналізу були сформульовані вимоги до функціоналу, інтерфейсу користувача, продуктивності, безпеки та інших важливих аспектів. Вимоги включають можливість легко створювати, редагувати та видаляти списки справ і їх дедлайни, а також можливість фільтрувати списки за статусом виконання та інші.
3. В процесі розробки вивчалися та оцінювалися різні інструменти та технології, серед яких мова програмування C#, середовище розробки Visual Studio 2022, система керування базами даних DBeaver та MySQL, а також бібліотека WinForms. Було прийнято рішення використовувати ці технології, оскільки вони забезпечують необхідну функціональність та стабільність для реалізації проекту.
4. Було розроблено застосунок відповідно до розробленої архітектури та вимогам. Для кожної з форми було розроблено та реалізовано унікальний функціонал для ефективного вирішення поставлених вимог.
5.Проведено тестування застосунку та виправлені виявлені помилки та недоліки.

12

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

Форма головної сторінки

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Configuration;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Test_project.Properties;

namespace Test_project
{
   public partial class MainForm : Form
   {
      private string userLogin;
      private int status_view = 1;
      private bool show_all = true;
      private FlowLayoutPanel flowLayoutPanel;

      public MainForm(string login)
      {
         InitializeComponent();
         this.userLogin = login;
         flowLayoutPanel = new FlowLayoutPanel();
         flowLayoutPanel.Dock = DockStyle.Fill;
         flowLayoutPanel.AutoScroll = true;
         flowLayoutPanel.FlowDirection =
FlowDirection.TopDown;
         flowLayoutPanel.WrapContents = false;
         this.Controls.Add(flowLayoutPanel);
         //this.FormClosing += MainForm_FormClosing;
         SetLabelPosition();
      }

      private static bool exiting;

      private void MainForm_FormClosing(object sender,
FormClosingEventArgs e)
      {
         if (e.CloseReason == CloseReason.UserClosing
&& !exiting)
         {
            if (MessageBox.Show("Are you sure want to
exit?",
                  "ToDoPlatform",
                   MessageBoxButtons.OKCancel,
                   MessageBoxIcon.Information) ==
DialogResult.OK)
            {
               exiting = true;
               Environment.Exit(0);
            }
            else
            {
               e.Cancel = true;
            }
         }
      }

      private void MainForm_Load(object sender,
EventArgs e)
      {
         UpdateData();
      }

      private void NewTaskButton_Click(object sender,
EventArgs e)
```

```
    {
        this.Hide();
        NewTaskForm newTaskForm = new(userLogin);
        newTaskForm.Show();
    }

    private void editButton_Click(int id)
    {
        this.Hide();
        EditForm editform = new(id, userLogin);
        editform.Show();
    }

    private void ToDoLable_Click(object sender,
EventArgs e)
    {
        if (status_view == 1)
            InProgresLable.Font = new
Font(InProgresLable.Font, FontStyle.Regular);
        if (status_view == 2)
            DoneLable.Font = new Font(DoneLable.Font,
FontStyle.Regular);
        if (status_view == 0)
        {
            show_all = true;
            ToDoLable.Font = new Font(ToDoLable.Font,
FontStyle.Regular);
            UpdateData();
            status_view = 3;
            return;
        }
        show_all = false;
        status_view = 0;
        ToDoLable.Font = new Font(ToDoLable.Font,
FontStyle.Underline);
        UpdateData();
    }

    private void InProgresLable_Click(object sender,
EventArgs e)
    {
        if (status_view == 0)
            ToDoLable.Font = new Font(ToDoLable.Font,
FontStyle.Regular);
        if (status_view == 2)
            DoneLable.Font = new Font(DoneLable.Font,
FontStyle.Regular);
        if (status_view == 1)
        {
            show_all = true;
            InProgresLable.Font = new
Font(InProgresLable.Font, FontStyle.Regular);
            UpdateData();
            status_view = 3;
            return;
        }
        show_all = false;
        status_view = 1;
        UpdateData();
        InProgresLable.Font = new
Font(InProgresLable.Font, FontStyle.Underline);
    }

    private void DoneLable_Click(object sender,
EventArgs e)
    {
        if (status_view == 0)
            ToDoLable.Font = new Font(ToDoLable.Font,
FontStyle.Regular);
        if (status_view == 1)
            InProgresLable.Font = new
Font(InProgresLable.Font, FontStyle.Regular);
        if (status_view == 2)
        {
            show_all = true;
            DoneLable.Font = new Font(DoneLable.Font,
FontStyle.Regular);
            UpdateData();
            status_view = 3;
```

```
            return;
        }
        show_all = false;
        status_view = 2;
        UpdateData();
        DoneLable.Font = new Font(DoneLable.Font,
FontStyle.Underline);
    }

    private void UpdateData()
    {
        flowLayoutPanel.Controls.Clear();

        DataBase db = new DataBase();
        DataTable dt = new DataTable();

        MySqlDataAdapter adapter = new
MySqlDataAdapter();
        if (!show_all)
        {
            MySqlCommand command = new
MySqlCommand("SELECT * FROM `ToDoData`
WHERE `login` = @log AND `Status` = @stat ORDER
BY `id` DESC", db.getConnection());

            command.Parameters.Add("@log",
MySqlDbType.VarChar).Value = userLogin;
            command.Parameters.Add("@stat",
MySqlDbType.Enum).Value = status_view;

            adapter.SelectCommand = command;
            adapter.Fill(dt);
        }
        else
        {
            MySqlCommand command = new
MySqlCommand("SELECT * FROM `ToDoData`
WHERE `login` = @log ORDER BY `id` DESC",
db.getConnection());
            command.Parameters.Add("@log",
MySqlDbType.VarChar).Value = userLogin;

            adapter.SelectCommand = command;
            adapter.Fill(dt);
        }

        if (dt.Rows.Count > 0)
        {
            for (int i = 0; i < dt.Rows.Count; i++)
            {
                Panel rowPanel = new Panel();
                rowPanel.Size = new
Size(this.ClientSize.Width - 23, 50);
                rowPanel.BackColor =
SystemColors.ControlDarkDark;

                Button button = new Button();
                button.Name = "EditButton" + i;
                button.Location = new
Point(rowPanel.Width - 70, 10);
                button.Size = new Size(60, 30);
                if
(ConfigurationManager.AppSettings["Language"] ==
"en-US")
                {
                    button.Text = "Edit";
                }
                else if
(ConfigurationManager.AppSettings["Language"] ==
"uk-UA")
                {
                    button.Text = "Змінити";
                }
                int id = Convert.ToInt32(dt.Rows[i]["id"]);
                button.Click += (sender, e) =>
editButton_Click(id);
                button.UseVisualStyleBackColor = true;
                rowPanel.Controls.Add(button);

                TextBox nameBox = new TextBox();
```

```
            nameBox.Name = "NameText" + i;
            nameBox.Size = new Size(200, 30);
            nameBox.Text =
dt.Rows[i]["name"].ToString();
            nameBox.Location = new Point(10, 10);
            nameBox.ReadOnly = true;
            nameBox.BackColor =
SystemColors.ControlDarkDark;
            nameBox.ForeColor =
SystemColors.ButtonHighlight;
            rowPanel.Controls.Add(nameBox);


            DateTimePicker dateBox = new
DateTimePicker();
            dateBox.Name = "DateBox" + i;
            dateBox.Size = new Size(130, 30);
            dateBox.Text =
dt.Rows[i]["date"].ToString();
            dateBox.Location = new Point(220, 10);
            dateBox.Enabled = false;
            dateBox.BackColor =
SystemColors.ControlDarkDark;
            dateBox.ForeColor =
SystemColors.ButtonHighlight;
            rowPanel.Controls.Add(dateBox);


            TextBox descriptionBox = new TextBox();
            descriptionBox.Name = "DescriptionBox" +
i;
            descriptionBox.Size = new Size(300, 30);
            descriptionBox.Text =
dt.Rows[i]["description"].ToString();
            descriptionBox.Location = new Point(360,
10);
            descriptionBox.ReadOnly = true;
            descriptionBox.BackColor =
SystemColors.ControlDarkDark;
            descriptionBox.ForeColor =
SystemColors.ButtonHighlight;
            rowPanel.Controls.Add(descriptionBox);


            TextBox statusBox = new TextBox();
            statusBox.Name = "StatusBox" + i;
            statusBox.Size = new Size(100, 30);
            statusBox.Location = new Point(670, 10);
            statusBox.ReadOnly = true;
            statusBox.BackColor =
SystemColors.ControlDarkDark;
            statusBox.ForeColor =
SystemColors.ButtonHighlight;
            if
(ConfigurationManager.AppSettings["Language"] ==
"en-US")
            {
                switch
(Convert.ToInt32(dt.Rows[i]["Status"]))
                {
                    case 0: statusBox.Text = "To Do";
break;
                    case 1: statusBox.Text = "In Progres";
break;
                    case 2: statusBox.Text = "Done";
break;
                }
            }
            else if
(ConfigurationManager.AppSettings["Language"] ==
"uk-UA")
            {
                switch
(Convert.ToInt32(dt.Rows[i]["Status"]))
                {
                    case 0: statusBox.Text =
"Заплановано"; break;
                    case 1: statusBox.Text = "В процесі";
break;
                    case 2: statusBox.Text = "Виконано";
break;
                }
            }
            rowPanel.Controls.Add(statusBox);


            flowLayoutPanel.Controls.Add(rowPanel);
        }
    }
```

```
        dt.Clear();
    }

    private void LanguageChangeButton_Click(object
sender, EventArgs e)
    {
        LenguageChose lenguageChose = new
LenguageChose();
        lenguageChose.ChooseLanguage();
        this.Hide();
        MainForm mainForm = new
MainForm(ConfigurationManager.AppSettings["Login"]
);
        mainForm.Show();
    }

    private void LogoutBytton_Click(object sender,
EventArgs e)
    {
        Configuration config =
ConfigurationManager.OpenExeConfiguration(Configur
ationUserLevel.None);
        config.AppSettings.Settings["Login"].Value = "";
        config.Save(ConfigurationSaveMode.Modified);

ConfigurationManager.RefreshSection("appSettings");
        this.Hide();
        LoginForm loginForm = new LoginForm();
        if (loginForm.ShowDialog() ==
DialogResult.OK)
        {
            MainForm mainForm = new
MainForm(ConfigurationManager.AppSettings["Login"]
);
            mainForm.Show();
        }
    }
    private void MainForm_Resize(object sender,
EventArgs e)
    {
        // Зміна позиції Label при зміні розміру форми
        SetLabelPosition();
    }

    private void SetLabelPosition()
    {

        double todopercent = 35;
        double inprogres = 50;
        double donepercent = 65;

        int ToDoLableX = (int)(this.ClientSize.Width *
(todopercent / 100.0));
        int InProgresX = (int)(this.ClientSize.Width *
(inprogres / 100.0));
        int DoneX = (int)(this.ClientSize.Width *
(donepercent / 100.0));

        ToDoLable.Location = new
System.Drawing.Point(ToDoLableX - ToDoLable.Width
/ 2, ToDoLable.Location.Y);
        InProgresLable.Location = new
System.Drawing.Point(InProgresX -
InProgresLable.Width / 2, InProgresLable.Location.Y);
        DoneLable.Location = new
System.Drawing.Point(DoneX - DoneLable.Width / 2,
DoneLable.Location.Y);
    }
  }
}
```

Файл створення статичних об’єктів головної форми

```
namespace Test_project
{
    partial class MainForm
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private
System.ComponentModel.IContainer
components = null;

        /// <summary>
        /// Clean up any resources
being used.
        /// </summary>
```

```
        /// <param
name=»disposing»>true if managed
resources should be disposed;
otherwise, false.</param>
        protected override void
Dispose(bool disposing)
        {
            if (disposing &&
(components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer
generated code

        /// <summary>
        /// Required method for
Designer support - do not modify
        /// the contents of this method
with the code editor.
        /// </summary>
        private void
InitializeComponent()
        {

System.ComponentModel.ComponentResource
Manager resources = new
System.ComponentModel.ComponentResource
Manager(typeof(MainForm));
            NewTaskButton = new
Button();
            ToDoLable = new Label();
            InProgresLable = new
Label();
            DoneLable = new Label();
            LanguageChangeButton = new
Button();
            LogoutBytton = new
Button();
            SuspendLayout();
            //
            // NewTaskButton
            //

resources.ApplyResources(NewTaskButton,
«NewTaskButton»);
            NewTaskButton.Name =
«NewTaskButton»;

NewTaskButton.UseVisualStyleBackColor =
true;
            NewTaskButton.Click +=
NewTaskButton_Click;
            //
            // ToDoLable
            //

resources.ApplyResources(ToDoLable,
«ToDoLable»);
            ToDoLable.ForeColor =
SystemColors.ButtonHighlight;
            ToDoLable.Name =
«ToDoLable»;
            ToDoLable.Click +=
ToDoLable_Click;
            //
            // InProgresLable
            //

resources.ApplyResources(InProgresLable
, «InProgresLable»);
            InProgresLable.ForeColor =
SystemColors.ButtonHighlight;
            InProgresLable.Name =
«InProgresLable»;
            InProgresLable.Click +=
InProgresLable_Click;
            //
            // DoneLable
            //

resources.ApplyResources(DoneLable,
«DoneLable»);
            DoneLable.ForeColor =
SystemColors.ButtonHighlight;
            DoneLable.Name =
«DoneLable»;
            DoneLable.Click +=
DoneLable_Click;
            //
            // LanguageChangeButton
            //

resources.ApplyResources(LanguageChange
Button, «LanguageChangeButton»);
            LanguageChangeButton.Name =
«LanguageChangeButton»;

LanguageChangeButton.UseVisualStyleBack
Color = true;
            LanguageChangeButton.Click
+= LanguageChangeButton_Click;
            //
            // LogoutBytton
            //

resources.ApplyResources(LogoutBytton,
«LogoutBytton»);
            LogoutBytton.Name =
«LogoutBytton»;

LogoutBytton.UseVisualStyleBackColor =
true;
            LogoutBytton.Click +=
LogoutBytton_Click;
            //
            // MainForm
            //

resources.ApplyResources(this,
«$this»);
```

```
            AutoScaleMode =
AutoScaleMode.Font;
            BackColor =
Color.FromArgb(45, 45, 48);
            Controls.Add(LogoutBytton);

Controls.Add(LanguageChangeButton);
            Controls.Add(DoneLable);

Controls.Add(InProgresLable);
            Controls.Add(ToDoLable);

Controls.Add(NewTaskButton);
            Name = «MainForm»;
            FormClosing +=
MainForm_FormClosing;
            Load += MainForm_Load;
            ResizeEnd +=
MainForm_Resize;
            ResumeLayout(false);
            PerformLayout();
        }

        #endregion
        private Button NewTaskButton;
        private Label ToDoLable;
        private Label InProgresLable;
        private Label DoneLable;
        private Button
LanguageChangeButton;
        private Button LogoutBytton;
    }
}
```