

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

ДЛЯ КАЛЕНДАРНОГО ПЛАНУВАННЯ

SOFTWARE FOR CALENDAR PLANNING

Виконав(ла): студент(ка) 4 курсу, групи КНТ-131

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

КУЩЕНКО М.С.

(ПРИЗВИЩЕ та ініціали)

Керівник ЗАЙКО Т.А.

(ПРИЗВИЩЕ та ініціали)

Рецензент КОЦУР М.І.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2025 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

КУЩЕНКА Микити Сергійовича

(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення для календарного планування. Software for Calendar Planning

керівник проєкту (роботи) к.т.н., доцент, ЗАЙКО Тетяна Анатоліївна,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 28 ” квітня 2025 року № 209

2. Строк подання студентом проєкту (роботи) 03 червня 2025 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів) _____

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ЗАЙКО Т.А., професор		
Нормоконтроль	КАМІНСЬКА Ж.К., асистент		

7. Дата видачі завдання “ 28 ” квітня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

_____ Микита КУЩЕНКО
(підпис) (Імя ПРИЗВИЩЕ)

Керівник проєкту (роботи)

_____ Тетяна ЗАЙКО
(підпис) (Імя ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
100 с., 3 табл., 29 рис., 3 дод., 18 джерел.

C#, VISUAL STUDIO, ЗАСТОСУНОК, КАЛЕНДАРНЕ
ПЛАНУВАННЯ, МОВА ПРОГРАМУВАННЯ, ПРОГРАМНЕ
ЗАБЕЗПЕЧЕННЯ.

Об'єкт дослідження – процес розробки програмного забезпечення для календарного планування.

Предмет дослідження – програмні засоби для календарного планування.

Мета роботи – розробка програмного забезпечення для календарного планування.

Матеріали, методи та технічні засоби: мова програмування C#, середовище розробки Visual Studio.

Результати. Розроблено проєктні рішення для створення програмного забезпечення для календарного планування. Створено програмне забезпечення для підтримки процесу календарного планування. Проведено тестування розробленої програмної системи для календарного планування.

Висновки. Мету роботи досягнуто. Розроблено програмне забезпечення для календарного планування за допомогою мови програмування C# та середовища розробки Visual Studio.

Галузь використання – управління проєктами.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 100 pages, 3 tables, 29 figures, 3 appendixes, 18 sources.

C#, VISUAL STUDIO, APPLICATION, CALENDAR PLANNING, PROGRAMMING LANGUAGE, SOFTWARE.

The object of research is the process of developing software for calendar planning.

The subject of the research is software for calendar planning.

The purpose of this work is to develop the software for calendar planning.

Materials, methods and technical tools: C# programming language, Visual Studio development environment.

Results. Project solutions for the creation of software for calendar planning has been designed. The software to support the process of calendar planning has been developed. The developed software system for calendar planning has been tested.

Conclusions. The goal of the work has been achieved. The software tools for calendar planning using the C# programming language and the Visual Studio development environment have been developed.

Scope of use – project management.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ	9
1 Аналіз предметної області	11
1.1 Програмне забезпечення для календарного планування	11
1.2 Аналіз програмного забезпечення для календарного планування	21
1.3 Висновки за розділом 1	32
2 Матеріали і методи	33
2.1 Вибір мови програмування.....	33
2.2 Вибір середовища розробки для створення програмного забезпечення для календарного планування	36
2.3 Висновки за розділом 2	39
3 Опис програми	40
3.1 Структура програмного забезпечення для календарного планування....	40
3.2 Функціонування програмного забезпечення для календарного планування	45
3.3 Розробка бази даних програмного забезпечення для календарного планування	48
3.4 Проектування інтерфейсу програмного забезпечення для календарного планування	51
3.5 Висновки за розділом 3	54
4 Експлуатація, тестування та експериментальне дослідження програми.....	56
4.1 Призначення й умови застосування програми	56
4.2 Характеристики програми для календарного планування	57
4.3 Інструкція по експлуатації програми.....	58
4.3.1 Звернення до програми.....	58
4.3.2 Вхідні й вихідні дані.....	58
4.3.3 Повідомлення.....	58
4.4 Виконання програмного забезпечення для календарного планування ...	58

4.5 Тестування програмного забезпечення для календарного планування ..	63
4.6 Висновки за розділом 4	63
Висновки.....	64
Перелік джерел посилань.....	67
Додаток А Технічне завдання.....	69
Додаток Б Фрагмент тексту програми	73
Додаток В Слайди презентації	93

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API – application programming interface;

FK – foreign key;

PK – primary key;

БД – база даних;

КП – календарне планування;

ПЗ – програмне забезпечення.

ВСТУП

Актуальність програмного забезпечення для календарного планування обумовлена сучасними потребами управління часом у динамічному світі. У наш час як у бізнесі, так і в особистому житті люди стикаються з великою кількістю завдань та зустрічей, і програмне забезпечення допомагає структурувати ці активності, уникати конфліктів у розкладі та ефективно використовувати час. Особливо важливою стала роль таких інструментів у період поширення віддаленої та гібридної роботи, коли необхідно синхронізувати розклади команд, що працюють у різних місцях, організовувати онлайн зустрічі та інтегрувати календарі з іншими системами управління завданнями [1].

Календарне планування сьогодні виходить за межі простого запису подій, адже сучасне програмне забезпечення інтегрується з електронною поштою та іншими програмними інструментами, створюючи єдиний інформаційний простір. Новітні програмні рішення також використовують автоматизацію та штучний інтелект, пропонуючи можливість автоматичного вибору часу зустрічей з урахуванням зайнятості учасників та пріоритетів завдань. Зростає й потреба в мобільності, адже користувачі хочуть мати постійний доступ до своїх розкладів з будь-яких пристроїв, що стимулює розвиток кросплатформених рішень [2], [3].

Попри свою корисність і популярність, програмне забезпечення для календарного планування має низку недоліків, які обмежують його використання або створюють додаткові виклики. Однією з проблем використання таких програмних засобів є питання конфіденційності та захисту даних, оскільки багато сервісів для календарного планування зберігають дані користувачів у хмарі, що підвищує ризик витоку інформації, особливо якщо календар містить особисті чи корпоративно важливі дані. Крім того, складність інтерфейсу або надмірна кількість функцій у деяких програмах для календарного планування може стати бар'єром для

користувачів, які не мають достатнього досвіду роботи з такими системами, або призводити до перевантаження інформацією. Ще одним недоліком є обмежена сумісність між різними програмними платформами та сервісами. Не всі програмні засоби для календарного планування легко інтегруються з іншими програмами, через що виникають проблеми з синхронізацією даних між різними пристроями чи застосунками [4], [5].

Тому актуальною є розробка програмного забезпечення для календарного планування. У дипломній кваліфікаційній роботі бакалавра розв'язується актуальне завдання розробки програмного забезпечення для календарного планування.

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для календарного планування;
- здійснити проєктування програмного забезпечення для календарного планування;
- створити програмне забезпечення для календарного планування;
- виконати тестування розробленого програмного забезпечення для календарного планування.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Програмне забезпечення для календарного планування

Календарне планування є важливим аспектом для ефективного управління часом як у професійній, так і в особистій діяльності. Різні програмні рішення, що включають функції календаря, здатні допомогти з організацією та плануванням подій, зустрічей і завдань. Однак створення та впровадження календарних програмних систем може бути досить складним процесом, що потребує певних ресурсів та знань у галузі розробки програмного забезпечення [1], [2].

Існує велика кількість програмних рішень, що використовуються для управління календарем у різних сферах діяльності. У мобільному середовищі, де використовуються численні мобільні телефони та планшети, календарні додатки набули величезного поширення. Такі програми стали важливим інструментом для управління повсякденними завданнями, оскільки вони дозволяють організувати час і нагадують користувачам про важливі події та зустрічі. З розвитком технологій і збільшенням попиту на цифрові платформи, календарі перетворилися на основний інструмент для багатьох користувачів, що бажають максимально оптимізувати своє планування [1], [2].

Календарні програми використовуються в широкому спектрі індустрій. У сфері освіти такі програми дозволяють планувати уроки, іспити та інші важливі події для студентів і викладачів. Вони забезпечують своєчасні нагадування, що є важливим для підтримки дисципліни та організованості. У туристичній сфері календарі допомагають зберігати актуальність розкладів подорожей і організовувати зустрічі з клієнтами. Подібні рішення дозволяють туристичним компаніям точно планувати та координувати свої дії [1], [2].

Отже, програмне забезпечення для календарного планування розглядається як інструмент, призначений для впорядкування та координації заходів, зустрічей і завдань у межах встановленого розкладу. Його використання дозволяє автоматизувати процес управління часом, полегшуючи

створення, редагування та відстеження подій. Завдяки такому програмному забезпеченню забезпечується зручність доступу до графіків, що дає змогу своєчасно реагувати на зміни та уникати накладок. Застосування цього типу рішень сприяє підвищенню організованості, зменшенню кількості пропущених заходів і кращій координації діяльності як на індивідуальному, так і на командному рівнях. У результаті забезпечується більш ефективне використання робочого часу та ресурсів, що позитивно впливає на загальну продуктивність. Чи планується використання такого програмного забезпечення для особистих потреб чи в професійному середовищі, воно залишається важливою складовою процесу планування та управління завданнями [1], [2].

Важливість календарного планування також відзначається в медичній сфері. Програми, що інтегрують календарні функції, дозволяють користувачам планувати візити до лікаря, а також нагадувати про прийом ліків або проведення медичних процедур. Ці програмні інструменти допомагають забезпечити дотримання режиму лікування, що особливо важливо для підтримки здоров'я [1], [2].

У транспортній галузі календарне планування використовуються для організації бронювань і відстеження різних подій, пов'язаних з доставкою товарів або перевезеннями. Відправники та ті, хто отримують посилки, можуть вибирати найбільш зручний час для здійснення своїх операцій [1], [2].

Календарі також активно інтегруються у соціальні мережі, дозволяючи користувачам планувати особисті зустрічі або публічні події через платформи на кшталт Facebook, Instagram і Twitter. Це дозволяє організовувати заходи, запрошувати людей і координувати зустрічі, значно полегшуючи комунікацію між учасниками [1], [2].

Таким чином, календарне планування в цифровому форматі є незамінним інструментом у різних сферах життя, допомагаючи людям ефективно організовувати свій час і залишатися на зв'язку з усіма важливими подіями [1], [2].

Розробка програмного забезпечення для календарного планування передбачає вибір відповідного технологічного набору, що забезпечує ефективну роботу додатку та можливість його розвитку. Для створення мобільних версій програмного забезпечення календарного типу важливо правильно підібрати інструменти, які підтримуватимуть потреби як користувачів, так і розробників [1], [2].

Для інтеграції функцій календаря в мобільні додатки можна використовувати різноманітні інтерфейси програмування додатків (API). Наприклад, компанії Apple і Google пропонують власні рішення для роботи з календарями. Інструмент EventKit від Apple дозволяє створювати та редагувати події в календарі, надаючи користувачам можливість організовувати свою діяльність з урахуванням усіх необхідних параметрів. Крім того, існує інтерфейс EventKitUI, який надає готові елементи інтерфейсу для додавання в програми. З іншого боку, Google пропонує API для свого календаря, що дозволяє інтегрувати функції створення, перегляду та редагування подій як в мобільних, так і в вебзастосунках. Важливою перевагою цього API є наявність інструкцій для використання з різними мовами програмування, такими як Ruby, JavaScript, PHP та інші [2], [3].

Окрім стандартних API, існують і додаткові інструменти, які можуть допомогти в розробці більш специфічних функцій календаря. Наприклад, API Nylas Cloud надає можливість додавати нагадування, а SuperSaaS дозволяє планувати онлайн зустрічі. Також доступні інші API для інтеграції державних свят або фінансових подій у додаток [2], [3].

Що стосується бібліотек для розробки календаря, існують різні варіанти для платформ Android та iOS. Для Android доступні такі інструменти, як Calendar Cosmo та Agenda Calendar View, які допомагають реалізувати основні функції календаря. Для iOS існують бібліотеки CalendarLib і MBCalendarKit, що надають готові рішення для створення календаря. Для вебзастосунків можна інтегрувати функції за допомогою програмних інструментів Webix або jQuery UI [2], [3].

Для розробки програмного забезпечення для календарного планування важливо використовувати відповідні технології, такі як мови програмування Java, Kotlin для Android, Swift та Objective-C для iOS. Щодо баз даних, розробники можуть вибирати між MongoDB, PostgreSQL або MySQL, залежно від вимог проекту. Також для створення зручного інтерфейсу користувача можна використовувати інструменти, як Illustrator або Sketch [2], [3].

Для тестування програмного забезпечення для календарного планування важливо використовувати ефективні інструменти для забезпечення високої якості продукту. Серед таких інструментів є Postman для тестування API, TestFlight для бета-тестування додатків на iOS, а також Firebase і Crashlytics для відстеження помилок та стабільності додатків після запуску [2], [3].

При створенні програмного забезпечення для календарного планування важливо врахувати ряд функцій, які зроблять використання додатка максимально зручним і ефективним для користувачів. Однією з ключових можливостей є налаштування нагадувань, яке допомагає не забути важливі події або зустрічі, що значно зручніше за традиційні паперові записи [2], [3].

Завдання користувачів мають бути організовані в календарі, що дає змогу легко відстежувати їх виконання та позначати завершені. Це сприяє кращій організації роботи і командній взаємодії. Крім того, важливою є функція програмного забезпечення для календарного планування для блокування часу, що дозволяє виділяти певні періоди для важливих завдань, попереджаючи про свою недоступність для інших осіб [2], [3].

Для тих, хто працює за неповним робочим днем або має обмежений час, варто передбачити можливість резервування часу для зустрічей. Це дозволяє зручно планувати зустрічі, даючи можливість ділитися доступними інтервалами часу через посилання на календар [2], [3].

Наявність функції перегляду сповіщень дає змогу бути в курсі всіх змін, таких як нові учасники, скасування подій чи зміни в розкладі. Також

важливою є підтримка повторюваних подій, що дозволяє без труднощів організовувати регулярні зустрічі або наради [2], [3].

Управління ресурсами через програмне забезпечення для календарного планування дозволяє ефективно розподіляти та контролювати їх використання, що особливо важливо для автоматизації процесів бронювання та планування. Для зручності планування важливо також передбачити можливість перегляду доступності контактів, щоб організатори могли швидко визначити зручні для всіх учасників дати та часи зустрічей [2], [3].

Інтеграція цих функцій дозволяє створити потужний інструмент для організації часу, що відповідає потребам різних категорій користувачів і покращує ефективність їхньої діяльності. Розуміння значення кожної з цих функцій є ключовим при розробці або вдосконаленні додатку календаря, що може значно підвищити його популярність серед користувачів [2], [3].

Для розробки програмного забезпечення календаря важливо чітко визначити його функціональні можливості та цілі. Перед початком розробки потрібно обрати ключові елементи, які забезпечать зручність і ефективність роботи, зокрема функції для створення подій, налаштування нагадувань та синхронізацію між пристроями. Важливим є також інтегрування з популярними платформами, що дозволить користувачам отримати доступ до календаря з різних пристроїв [2], [3].

Перш ніж розпочати розробку, необхідно сформулювати чіткий план, в якому буде визначено обсяг робіт та терміни для кожного етапу проекту. Такий підхід дозволяє розробникам краще орієнтуватися в процесі, контролювати виконання завдань і вчасно вносити корективи, що значно підвищує ефективність роботи команди [2], [3].

Вибір правильного технологічного стеку має вирішальне значення для майбутнього успіху додатку. Потрібно ретельно обирати технології, що найкраще підходять для ваших вимог, з урахуванням масштабованості, безпеки та користувацького досвіду. Використання кросплатформених рішень,

таких як React Native або Flutter, допоможе забезпечити зручність роботи на різних пристроях [2], [3].

При наймі розробників варто звертати увагу на досвід команди в створенні мобільних додатків та її здатність реалізувати функції, необхідні для календаря. Окрім технічних спеціалістів, важливо залучити дизайнерів інтерфейсу, які зможуть створити інтуїтивно зрозумілий інтерфейс [2], [3].

Процес розробки має бути гнучким, щоб зберігати адаптивність до змінних вимог. Впровадження методології гнучкої розробки дозволяє постійно вдосконалювати продукт, враховуючи зворотний зв'язок користувачів і підтримуючи високий рівень комунікації серед членів команди [2], [3].

Оскільки програмне забезпечення для календарного планування часто працює з конфіденційною інформацією, важливо дотримуватись найвищих стандартів безпеки. Всі дані повинні бути захищені на рівні шифрування та автентифікації користувачів, а також регулярно проводитись перевірки на відповідність вимогам конфіденційності [2], [3].

Перед запуском програмного забезпечення для календарного планування важливо провести всебічне тестування програми, включаючи модульне, інтеграційне та користувацьке тестування. Це дозволяє оцінити його функціональність, продуктивність та зручність для користувачів. Врахування відгуків з тестування дозволить усунути можливі проблеми до випуску фінальної версії програми [3], [4].

Створення програмного забезпечення для календарного планування вимагає врахування кількох важливих аспектів, щоб забезпечити зручний і ефективний користувацький досвід. Одним із ключових елементів є інтерфейс програми, який має бути інтуїтивно зрозумілим і легким у використанні. Важливо, щоб інтерфейс не лише був простим і привабливим, а й забезпечував безперешкодний доступ до всіх функцій, створюючи комфортне використання програми з першого запуску. Для цього варто звертати увагу на сучасні тенденції дизайну, включаючи використання яскравих кольорів та простих форм, які допомагають зробити додаток приємним для користувача [3], [4].

Ключовим аспектом є синхронізація даних, яка дозволяє користувачам без проблем використовувати додаток на різних пристроях, таких як мобільні телефони, планшети та комп'ютери. Це забезпечує безперебійний доступ до інформації, незалежно від того, на якому пристрої працює користувач. Використання хмарних технологій для зберігання даних допомагає уникнути втрати важливої інформації [3], [4].

Щоб виділити розроблене програмне забезпечення для календарного планування серед конкурентів, важливо впроваджувати передові функції та можливості. Це може бути різноманітність кольорових схем, можливість налаштування режимів перегляду та додавання нагадувань. Крім того, додаткові функції програмного забезпечення для календарного планування, такі як інтеграція з іншими сервісами або соціальними мережами, допомагають підвищити привабливість програми для користувачів [3], [4].

Ще одним важливим аспектом є використання можливостей апаратного забезпечення пристроїв для покращення функціональності та безпеки додатка. Наприклад, можна реалізувати підтримку технологій, таких як 3D Touch або Force Touch, що дозволить користувачам зручніше взаємодіяти з додатком. Крім того, можна додати можливість відображення інформації про місце подій, що також підвищить зручність планування [3], [4].

Розробка додатка для календарного планування потребує детального планування та стратегічного підходу, оскільки ціни на розробку можуть варіюватися залежно від складності додатка, платформи, на якій він працюватиме, та додаткових функцій. Простий додаток може бути створений за порівняно низьку вартість, але більш складні функції, такі як нагадування на основі гео локації або інтеграція з соціальними мережами, можуть значно підвищити вартість розробки. Крім того, важливо враховувати витрати на підтримку, маркетинг і оновлення додатка, що також можуть вплинути на загальну вартість [3], [4].

Вартість та тривалість розробки програмного забезпечення для календарного планування можуть залежати від багатьох факторів, включаючи

вибір платформи, вимоги до дизайну, складність серверної частини та інші функції. Структура команди розробників, а також її місце розташування також впливають на кінцеву вартість і терміни розробки. Врахування цих аспектів дозволить створити якісний програмний продукт, що відповідає всім вимогам користувачів [3], [4].

У сучасному цифровому середовищі програмне забезпечення для календарного планування виконує роль інструменту для організації часу та координації дій. Такі програмні продукти дозволяють користувачам створювати події, встановлювати нагадування та синхронізувати розклади між різними пристроями, забезпечуючи безперервний доступ до актуальної інформації [3], [4].

Завдяки можливості інтеграції з іншими платформами, такими як електронна пошта чи інші календарні сервіси, користувачі можуть централізовано керувати своїми зобов'язаннями. Це особливо корисно для тих, хто працює в команді або має численні зустрічі та події [3], [4].

Функція спільного доступу до календаря дозволяє кільком користувачам переглядати та редагувати розклад, що сприяє ефективній співпраці та уникненню конфліктів у плануванні. Крім того, можливість імпорту та експорту подій забезпечує гнучкість у використанні різних інструментів та платформ [3], [4].

Для забезпечення безпеки та конфіденційності даних, сучасні календарні додатки впроваджують різноманітні заходи захисту, включаючи шифрування даних та автентифікацію користувачів. Це гарантує, що особиста інформація залишається захищеною від несанкціонованого доступу [3], [4].

Таким чином, програмне забезпечення для календарного планування стає незамінним інструментом для організації повсякденного життя, забезпечуючи зручність, ефективність та безпеку у керуванні часом [3], [4].

Програмне забезпечення для календарного планування виконує функцію системи, що автоматизує організацію подій, зустрічей та завдань, сприяючи впорядкуванню особистих і професійних справ. За допомогою

такого програмного продукту здійснюється створення записів, редагування даних та налаштування нагадувань, що дозволяє зберігати контроль над розкладом [4], [5].

Завдяки інтеграції з іншими цифровими сервісами забезпечується обмін даними між різними платформами, що відкриває можливість синхронізації інформації на кількох пристроях. Це підвищує гнучкість використання й дозволяє уникати дублювання даних або конфліктів під час планування [4], [5].

У програмному середовищі подібного типу передбачено функціонал спільного доступу, завдяки якому декілька користувачів можуть мати право перегляду або редагування календаря, що сприяє покращенню командної взаємодії. Такий підхід полегшує координацію планів у колективних проєктах і мінімізує ризик накладання зустрічей [4], [5].

Впровадження заходів захисту даних, як-от шифрування та багаторівнева ідентифікація, підвищує рівень безпеки збережених відомостей. Це дозволяє зберігати конфіденційність записів, запобігаючи несанкціонованому доступу до приватної інформації. [4], [5].

У підсумку, програмне забезпечення для календарного планування виступає як комплексне рішення для управління часом, поєднуючи простоту використання, ефективність організації та захист даних у єдиній системі. Розробка програмного забезпечення для календарного планування часто сприймається як складне завдання, хоча насправді цей процес піддається освоєнню й організації. Такий вид програмних рішень створюється для полегшення керування подіями, зустрічами та завданнями, що дає змогу ефективніше розподіляти ресурси й уникати непотрібних помилок. Його застосування сприяє економії часу, підвищує рівень впорядкованості та забезпечує безперервність роботи як для окремих користувачів, так і для організацій різного масштабу [4], [5].

Програмне забезпечення для календарного планування виконує роль цифрового інструмента, який автоматизує процеси координації подій. Його

впровадження дає можливість контролювати розклади, забезпечувати доступність інформації з будь-якого пристрою та підтримувати синхронізацію між учасниками процесів. Використання таких систем допомагає підтримувати організованість, зменшує ризик накладання заходів та полегшує обмін інформацією в команді [4], [5].

Основними характеристиками програмного забезпечення для календарного планування вважаються інтуїтивно зрозумілий інтерфейс, функціонал для внесення змін до подій, а також можливість візуалізації розкладів за допомогою календарних уявлень. Також важливою є наявність нагадувань і повідомлень, які допомагають користувачам не пропускати важливі події. Інтеграція з іншими цифровими інструментами, наприклад, електронною поштою чи месенджерами, робить програмне забезпечення більш функціональним і зручним у використанні. Крім того, можливість налаштування під індивідуальні потреби користувачів підвищує гнучкість системи [4], [5].

Перед створенням програмного забезпечення для календарного планування важливо провести дослідження ринку, щоб отримати уявлення про очікування та вимоги потенційних користувачів. Таке дослідження допомагає зрозуміти, які функціональні можливості є найбільш затребуваними, а також виявити, що можна вдосконалити або запропонувати як інновацію. Вивчення ринку передбачає не лише виявлення цільової аудиторії, але й аналіз програмних рішень, що вже представлені на ринку [4], [5].

Ідентифікація потенційних користувачів дозволяє зрозуміти, які саме завдання вони очікують вирішити за допомогою програмного забезпечення для календарного планування. Залежно від сфери діяльності можуть знадобитися різні функціональні акценти, наприклад, можливість формування графіків для співробітників, бронювання зустрічей або управління кількома подіями одночасно. Вивчення потреб потенційних користувачів за допомогою опитувань та інтерв'ю дозволяє сформулювати перелік ключових функцій, що дійсно відповідають реальним викликам [4], [5].

Аналіз програмних продуктів конкурентів дає змогу оцінити існуючі пропозиції, зрозуміти їх сильні сторони, недоліки та відгуки користувачів. Такий підхід дозволяє знайти незаповнені ніші та визначити, як новий продукт може вигідно відрізнитися від аналогів. Ознайомлення з оцінками користувачів конкурентних продуктів допомагає уникнути поширених помилок і створити більш конкурентоспроможне рішення. Результати цього аналізу можна узагальнити у вигляді порівняльних даних, що слугуватимуть орієнтиром для формування функціоналу та підготовки стратегії просування продукту [4], [5].

Визначено, що програмне забезпечення для календарного планування є інструментом, призначеним для впорядкування та координації заходів, зустрічей і завдань у межах встановленого розкладу. Його використання дозволяє автоматизувати процес управління часом, полегшуючи створення, редагування та відстеження подій. Завдяки такому програмному забезпеченню забезпечується зручність доступу до графіків, що дає змогу своєчасно реагувати на зміни та уникати накладок. Застосування цього типу програмних рішень сприяє підвищенню організованості, зменшенню кількості пропущених заходів і кращій координації діяльності як на індивідуальному, так і на командному рівнях. У результаті забезпечується більш ефективне використання робочого часу та ресурсів, що позитивно впливає на загальну продуктивність.

1.2 Аналіз програмного забезпечення для календарного планування

Проаналізуємо програмне забезпечення для календарного планування [6]-[14].

Програмне забезпечення nTask (рис. 1.1) належить до категорії програмних рішень, які забезпечують зручну організацію робочих процесів і підтримку структури щоденної діяльності за допомогою календарного

планування. Це застосування вирізняється широким спектром функціональних можливостей, що дозволяє ефективно координувати завдання, планувати зустрічі та стежити за виконанням запланованих дій. Забезпечується високий рівень керованості робочим середовищем, що позитивно впливає на дисципліну виконання завдань та загальний рівень продуктивності [6], [12].

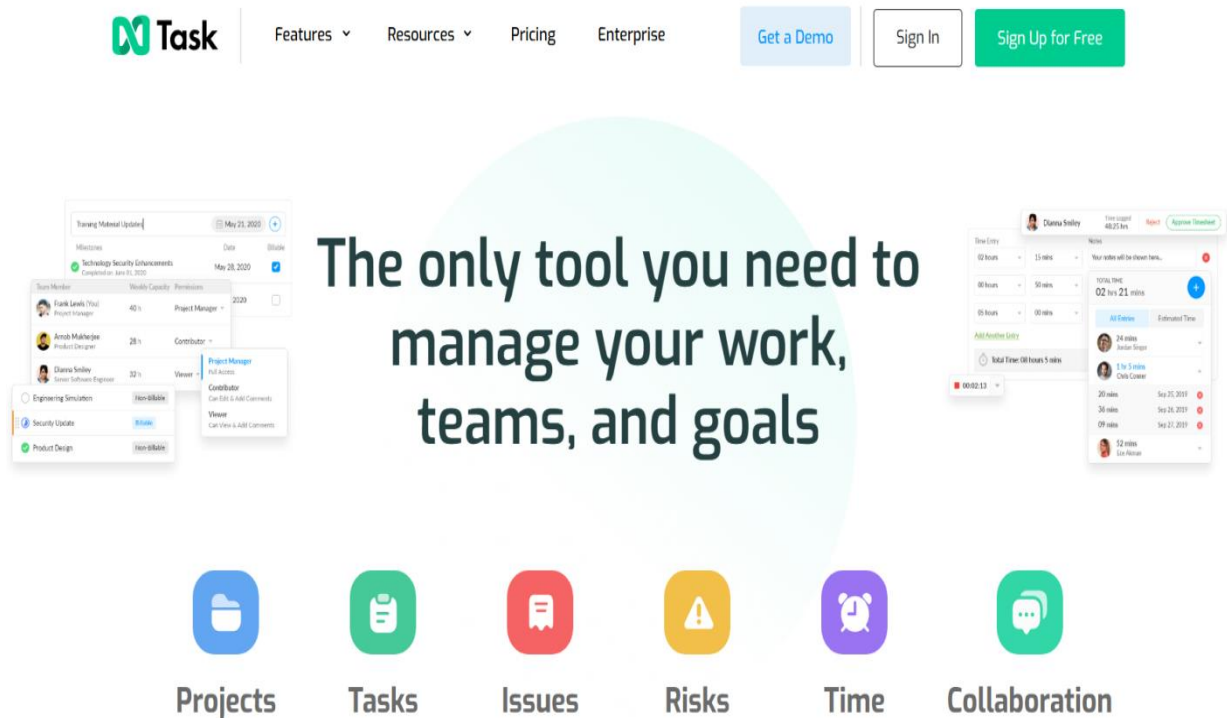


Рисунок 1.1 – Програмне забезпечення nTask [12]

Завдяки підтримці на різних мобільних платформах, таких як ті, що встановлюються на більшості мобільних телефонів і планшетів, забезпечується зручний доступ до функціоналу незалежно від використовуваного пристрою. Особливістю програмного забезпечення nTask є розширені можливості управління зустрічами: організатор отримує повний контроль над процесом запрошення учасників та може визначати, хто з них матиме право створювати командні сесії. Платформа nTask підтримує інтеграцію з популярними сервісами відео зв'язку, що значно спрощує проведення дистанційних нарад [6], [12].

У рамках середовища nTask також передбачено обмін інформацією, що стосується проєктів та конкретних завдань, а також використання інструментів для фіксації тривалості роботи над ними. Можливе формування порядку денного, управління часом проведення зустрічей та швидке поширення пов'язаної інформації серед учасників. Платформа nTask підтримує синхронізацію з основними поштовими та календарними сервісами, що дозволяє забезпечити цілісну взаємодію між запланованими подіями та іншими елементами цифрової організації робочого дня [6], [12].

Основними характеристиками програмного забезпечення nTask є такі [6], [12]:

- універсальність платформи: програмне забезпечення nTask підтримує роботу на різних мобільних операційних системах, що дозволяє користуватися програмою з більшості сучасних пристроїв [6], [12];

- управління завданнями: програмне забезпечення nTask дозволяє створювати, планувати та контролювати завдання, а також відстежувати їх виконання в рамках календаря [6], [12];

- контроль зустрічей: програмне забезпечення nTask надає організатору можливість вибирати учасників, запрошувати їх та делегувати права на створення командних сесій [6], [12];

- інтеграція з сервісами: програмне забезпечення nTask підтримує підключення до популярних систем відеозв'язку, електронної пошти та календарів, що полегшує синхронізацію подій [6], [12];

- функція обміну даними: програмне забезпечення nTask дає змогу ділитися посиланнями на проєкти, завдання та іншу інформацію безпосередньо між учасниками команди [6], [12];

- інструменти для тайм-менеджменту: програмне забезпечення nTask включає можливість встановлення таймерів для завдань, а також планування часу початку й завершення зустрічей [6], [12];

- формування порядку денного: програмне забезпечення nTask дозволяє створювати структуровані списки тем для обговорення під час зустрічей і поширювати їх серед учасників [6], [12];

- безкоштовний тариф: програмне забезпечення nTask надає базовий функціонал безкоштовно, але з обмеженим доступом до деяких розширених можливостей [6], [12].

Серед недоліків програмного забезпечення nTask можна відзначити обмеження безкоштовної версії, що не дозволяє користуватися всіма функціями без переходу на платний тариф, а також складність для нових користувачів через велику кількість налаштувань і можливостей, які потребують часу для освоєння. Деколи можуть спостерігатися затримки у синхронізації між різними пристроями, що створює певні незручності під час роботи з календарем у реальному часі. Також інтерфейс не завжди виглядає інтуїтивно зрозумілим, особливо для тих, хто звик до простіших рішень для планування [6], [12].

Програмне забезпечення Calendly (рис. 1.2) для календарного планування розглядається як ефективний інструмент, призначений переважно для автоматизації процесу організації зустрічей, що особливо корисно у сфері підбору персоналу. Його функціональність орієнтована на інтеграцію процесів призначення співбесід у загальну систему управління, що дозволяє позбутися необхідності ручного погодження часу та уникнути супутніх помилок. Завдяки цьому скорочуються витрати часу й зусиль, а також підвищується точність запланованих заходів. Програмний продукт Calendly вирізняється лаконічним і сучасним інтерфейсом, який забезпечує простоту сприйняття, залишаючись водночас привабливим і функціональним. Однією з важливих можливостей програми Calendly є надсилання автоматичних нагадувань про майбутні події, що мінімізує ризик пропусків [6], [13].

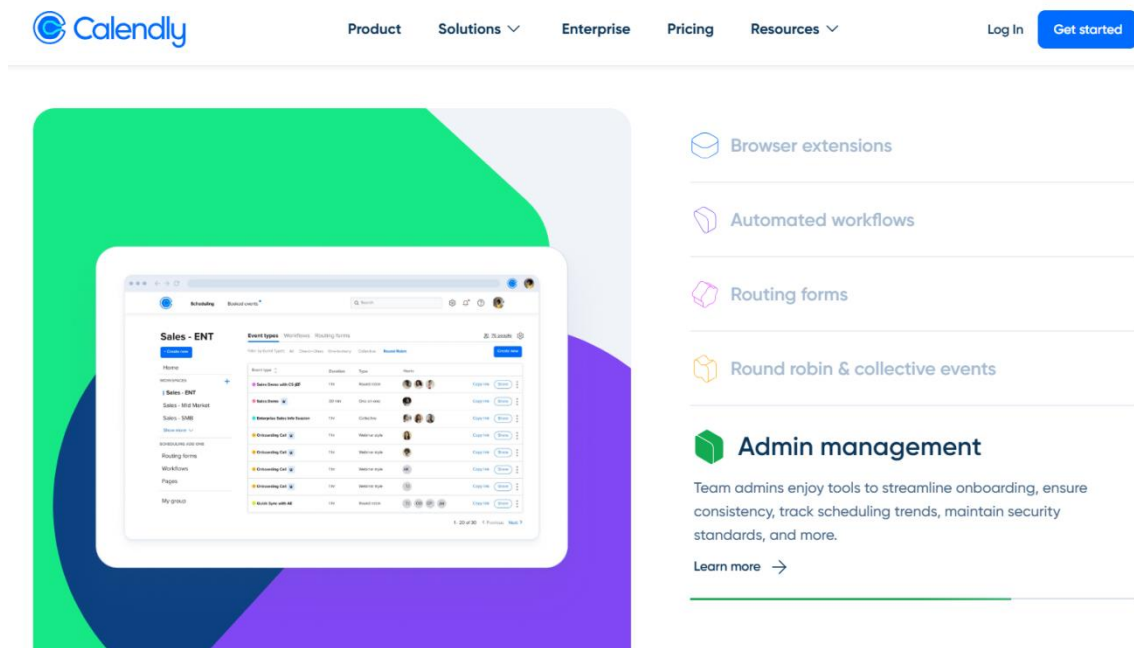


Рисунок 1.2 – Програмне забезпечення Calendly [13]

Програмне забезпечення Calendly підтримує роботу на різних пристроях і платформах, що робить його доступним незалежно від технічного середовища. Передбачено функцію організації співбесід за участю кількох представників компанії, що дозволяє залучати команду до процесу прийняття рішень. Система Calendly забезпечує автоматичне створення зустрічей, моніторинг діяльності, синхронізацію з іншими календарями, управління подіями, організацію групових заходів, а також використання шаблонів, які можна адаптувати під власні потреби [6], [13].

Програмне забезпечення Calendly має такі особливості [6], [13]:

- автоматичне призначення зустрічей: програмне забезпечення Calendly забезпечує самостійне планування подій без потреби у втручанні користувача, що економить час [6], [13];
- інтеграція з календарями: програмне забезпечення Calendly підтримує синхронізацію з Google Calendar, Outlook та іншими популярними сервісами для узгодженості розкладу [6], [13];
- нагадування про події: програмне забезпечення Calendly надсилає автоматичні сповіщення учасникам подій для зниження ризику пропуску зустрічі [6], [13];

- панель доступності команди: програмне забезпечення Calendly дозволяє переглядати вільний час членів команди для зручного вибору оптимального моменту для зборів [6], [13];
- групові події: програмне забезпечення Calendly надає можливість організовувати заходи за участю кількох людей одночасно, що корисно для командної роботи [6], [13];
- налаштовувані шаблони: програмне забезпечення Calendly пропонує варіанти шаблонів подій, які можна адаптувати під індивідуальні потреби користувачів або бізнесу [6], [13];
- відстеження активності: програмне забезпечення Calendly фіксує інформацію про заплановані та проведені зустрічі, що полегшує аналіз участі та активності [6], [13];
- простий інтерфейс: програмне забезпечення Calendly вирізняється мінімалістичним дизайном, що спрощує використання та забезпечує швидке освоєння новими користувачами [6], [13];
- доступність на різних платформах: програмне забезпечення Calendly працює на мобільних і настільних пристроях, що гарантує зручність роботи в будь-якому середовищі [6], [13].

Серед недоліків програмного забезпечення Calendly можна відзначити обмеженість деяких функцій у безкоштовній версії, що змушує користувачів переходити на платні тарифи для доступу до розширених можливостей. Також можна звернути увагу на складнощі з інтеграцією програмного забезпечення Calendly в окремі корпоративні системи через специфічні вимоги або обмеження сумісності. Інколи виникає потреба у додатковому налаштуванні інтерфейсу для повної відповідності індивідуальним бізнес-процесам, що потребує часу та технічної обізнаності [6], [13].

Програмне забезпечення Notion (рис. 1.3) виступає універсальним цифровим простором, що поєднує можливості ведення нотаток, календарного планування, створення документів, управління проєктами та організації спільної роботи [7], [8], [11], [14].

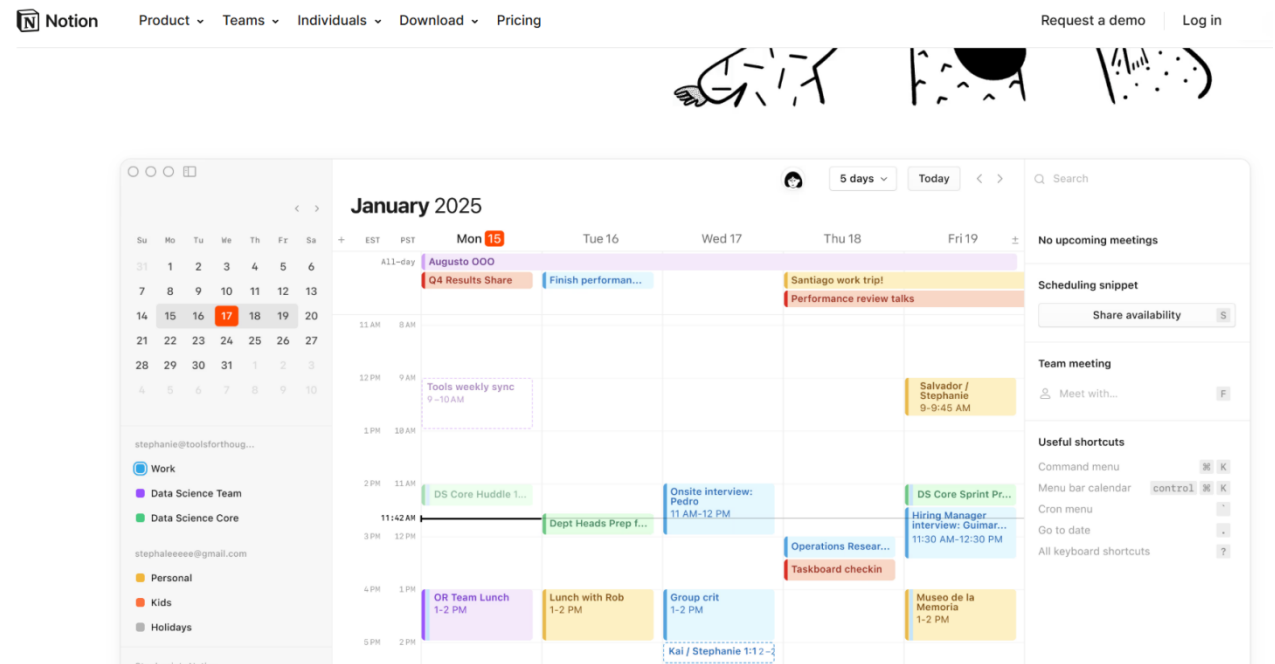


Рисунок 1.3 – Програмне забезпечення Notion [14]

Програмне забезпечення Notion можна використовувати індивідуально або разом із командою, поступово розширюючи коло користувачів. У середовищі Notion доступний готовий шаблон для щоденного планування, який охоплює календарні функції, ведення щоденника та інші елементи організації часу [7], [8], [11], [14].

Цей інструмент підходить для тих, хто надає перевагу гнучкості та творчості, створюючи цифровий планувальник з нуля відповідно до власних потреб. Серед його можливостей — система керування завданнями, перегляд розкладу у форматі календаря, а також функціонал для налаштування списків завдань у зручній формі [7], [8], [11], [14].

Дані, що зберігаються в Notion, можна синхронізувати між різними пристроями, а також поширювати їх з іншими користувачами. Програмне забезпечення Notion підтримує інтеграцію з різними зовнішніми сервісами, що сприяє підвищенню ефективності роботи [7], [8], [11], [14].

Серед ключових можливостей програмного забезпечення Notion є інтуїтивне управління завданнями, використання вкладених і пов'язаних баз даних для впорядкування інформації, а також інтеграція з іншими

платформами для створення єдиного робочого середовища. Крім того, є опції вбудовування зовнішніх матеріалів, що забезпечує легкий доступ до потрібного контенту прямо з панелі інструментів [7], [8], [11], [14].

Водночас програмне забезпечення Notion має певні обмеження: йому бракує розвинених інструментів автоматизації та системи нагадувань, а мобільний додаток поступається за функціональністю деяким аналогам. Для новачків процес освоєння програмного забезпечення Notion може видатися дещо складнішим через різноманіття функцій [7], [8], [11], [14].

Попри це, програмне забезпечення Notion добре підходить для централізованого зберігання інформації компанії. Додатково підтримується можливість вбудовування відео та завантаження файлів у межах певних обмежень. Структурування тексту відбувається через створення окремих блоків, що можна легко переміщати та комбінувати [7], [8], [11], [14].

Програмне забезпечення Notion дозволяє налаштовувати кожен елемент під власні потреби, від зовнішнього вигляду сторінок до вибору шрифтів, зображень і анімацій, що робить процес планування більш персоналізованим. Варто врахувати, що для ефективної роботи з програмним забезпеченням Notion на початку потрібно приділити час налаштуванню платформи відповідно до своїх завдань [7], [8], [11], [14].

Програмне забезпечення Notion має такі особливості [7], [8], [11], [14]:

- робочий простір для нотаток: програмне забезпечення Notion забезпечує створення, редагування та зберігання текстових записів у зручному форматі [7], [8], [11], [14];

- керування завданнями: програмне забезпечення Notion дозволяє створювати, організовувати та відстежувати завдання в межах одного інтерфейсу [7], [8], [11], [14];

- перегляд календаря: програмне забезпечення Notion підтримує календарний формат для планування подій і кінцевих термінів виконання завдань та робіт [7], [8], [11], [14];

- налаштовувані бази даних: програмне забезпечення Notion надає можливість створювати вкладені й пов'язані бази даних для структурованого зберігання інформації [7], [8], [11], [14];

- гнучкі шаблони: програмне забезпечення Notion пропонує готові шаблони для швидкого початку роботи з програмою та дозволяє створювати власні шаблони з нуля [7], [8], [11], [14];

- інструменти співпраці: програмне забезпечення Notion підтримує спільну роботу над проектом з іншими користувачами програми в режимі реального часу [7], [8], [11], [14];

- інтеграція з сервісами: програмне забезпечення Notion дає змогу підключати сторонні сервіси, як-от Slack або Gmail, для полегшення робочих процесів [7], [8], [11], [14];

- кросплатформність: програмне забезпечення Notion синхронізує дані між різними пристроями користувачів для доступу до контенту з будь-якого місця [7], [8], [11], [14];

- можливість вбудовування: програмне забезпечення Notion дозволяє додавати мультимедійний контент, як-от відео, GIF-файли та зображення, прямо на сторінки [7], [8], [11], [14];

- мінімальні обмеження безкоштовної версії: безкоштовний тариф програмного забезпечення Notion підтримує додавання файлів невеликого розміру та основних функцій [7], [8], [11], [14];

- адаптивність інтерфейсу: інтерфейс програмного забезпечення Notion можна налаштовувати під власні потреби користувачів через зручну систему налаштувань [7], [8], [11], [14];

- обмежена автоматизація: відсутність розвинених функцій автоматизації програмного забезпечення Notion та нагадувань порівняно з конкурентами [7], [8], [11], [14];

- складність для новачків: нові користувачі програмного забезпечення Notion можуть зіткнутися з труднощами під час освоєння функцій і налаштування [7], [8], [11], [14];

– обмежений мобільний функціонал: мобільна версія програмного забезпечення Notion має менший набір можливостей та функцій порівняно з настільною [7], [8], [11], [14].

Серед недоліків програмного забезпечення Notion можна відзначити обмежені можливості автоматизації процесів і відсутність вбудованих функцій нагадування, що ускладнює управління строками виконання завдань без використання сторонніх інструментів. Додатково варто звернути увагу на дещо урізаний функціонал мобільної версії, яка не завжди забезпечує такий самий рівень зручності та швидкості, як настільна платформа. Для нових користувачів процес освоєння може бути тривалим через необхідність розуміння великої кількості опцій та налаштувань, що потребує додаткового часу для адаптації [7], [8], [11], [14].

Порівняльну характеристику програмного забезпечення для календарного планування наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння програмного забезпечення для календарного планування

Критерій порівняння	nTask [12]	Calendly [13]	Notion [14]
Функції календаря та планування завдань	+	+	+—
Співпраця в команді	+	—	+
Організація зустрічей	+—	+	—
Інтеграція з іншими сервісами	+—	+	+
Простота інтерфейсу	+—	+	+—
Гнучкість налаштування	+—	—	+
Побудова робочих процесів	+	—	+—

За результатами проведеного аналізу можна зробити висновок, що у наш час існує досить багато програмних засобів для календарного планування. Проте деякі програмні засоби для календарного планування можуть мати складний інтерфейс або надмірну кількість функцій, що може стати бар'єром для користувачів, які не мають достатнього досвіду роботи з такими програмними системами, або призводити до перевантаження інформацією. Іншою проблемою використання таких програмних засобів є питання конфіденційності та захисту даних, оскільки багато сервісів для календарного планування зберігають дані користувачів у хмарі, що підвищує ризик витоку інформації, особливо якщо календар містить особисті чи корпоративно важливі дані. Ще одним недоліком є обмежена сумісність між різними програмними платформами та сервісами. Не всі програмні засоби для календарного планування легко інтегруються з іншими програмами, через що виникають проблеми з синхронізацією даних між різними пристроями чи застосунками.

Тому актуальною є розробка програмного забезпечення для календарного планування.

При розробці програмного забезпечення для календарного планування необхідно забезпечити такі функціональні вимоги:

- підтримка можливості відображення інформації про події користувача у хронологічному порядку;
- підтримка можливості календарного планування шляхом додавання інформації про події користувача;
- можливість корегування інформації про події користувача, зокрема, інформації про початок та завершення події;
- можливість відображення переліку подій користувача на день, тиждень та місяць;
- підтримка можливості переміщення між списками подій за допомогою календаря.

1.3 Висновки за розділом 1

Визначено, що програмне забезпечення для календарного планування є інструментом, призначеним для впорядкування та координації заходів, зустрічей і завдань у межах встановленого розкладу. Його використання дозволяє автоматизувати процес управління часом, полегшуючи створення, редагування та відстеження подій. Завдяки такому програмному забезпеченню забезпечується зручність доступу до графіків, що дає змогу своєчасно реагувати на зміни та уникати накладок. Застосування цього типу програмних рішень сприяє підвищенню організованості, зменшенню кількості пропущених заходів і кращій координації діяльності як на індивідуальному, так і на командному рівнях.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для календарного планування. Проте деякі програмні засоби для календарного планування можуть мати складний інтерфейс або надмірну кількість функцій, що може стати бар'єром для користувачів, які не мають достатнього досвіду роботи з такими програмними системами, або призводити до перевантаження інформацією. Іншою проблемою використання таких програмних засобів є питання конфіденційності та захисту даних, оскільки багато сервісів для календарного планування зберігають дані користувачів у хмарі, що підвищує ризик витоку інформації, особливо якщо календар містить особисті чи корпоративно важливі дані. Ще одним недоліком є обмежена сумісність між різними програмними платформами та сервісами. Не всі програмні засоби для календарного планування легко інтегруються з іншими програмами, через що виникають проблеми з синхронізацією даних між різними пристроями чи застосунками. Тому актуальною є розробка програмного забезпечення для календарного планування.

Сформульовано функціональні вимоги до програмного забезпечення для календарного планування.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

При розробці програмного забезпечення для календарного планування було використано мову програмування C# [15], [16].

C# розглядається як ефективне рішення для створення програмного забезпечення, що поєднує сучасний підхід до програмування з простотою використання. Цю мову було створено компанією Microsoft як складову екосистеми .NET, що забезпечує взаємодію застосунків із різними компонентами платформи. Основу мови C# складає об'єктно-орієнтована парадигма, що дозволяє структурувати програмні рішення таким чином, аби забезпечити гнучкість архітектури, можливість повторного застосування коду та легкість масштабування системи [15], [16].

Завдяки підтримці таких можливостей, як асинхронне програмування, багатопотокова обробка й сучасні концепції розробки, мова програмування C# дає змогу розробляти ефективні та продуктивні програми, здатні виконувати кілька завдань одночасно. Синтаксис мови C# нагадує інші популярні мови сімейства C, що полегшує її вивчення тим, хто вже має досвід роботи з C++ або Java [15], [16].

Мова програмування C# постійно вдосконалюється, отримуючи нові функції та розширення, що відповідають сучасним вимогам розробників. Використання C# у поєднанні з Visual Studio забезпечує комплексний підхід до програмування, надаючи інструменти для написання коду, тестування, налагодження та розгортання застосунків [15], [16].

Підтримка різноманітних платформ, серед яких Windows, Linux, macOS, Android і iOS, дозволяє застосовувати C# для створення як звичайних, так і мобільних та вебрішень. Мова відкриває широкі можливості для інтеграції з вебсервісами, API й базами даних, що робить її універсальною для розробки застосунків із різними функціональними вимогами [15], [16].

Завдяки поєднанню високого рівня абстракції, гнучкості та підтримки сучасних технологій, C# залишається одним із найбільш придатних інструментів для реалізації масштабованих, надійних і зручних програмних рішень [15], [16].

Для створення програмного забезпечення для календарного планування доцільно застосовувати мову програмування C#, оскільки завдяки своїм характеристикам вона забезпечує високу продуктивність і багатфункціональність. Незважаючи на глибоку інтеграцію в екосистему .NET, ця мова програмування C# відкриває можливість реалізувати рішення, що працюють на різних платформах, включаючи настільні, веб- і мобільні додатки. Використання фреймворків, таких як Xamarin чи MAUI, сприяє розробці застосунків, здатних функціонувати на різних пристроях, що особливо важливо для програм, орієнтованих на управління подіями й завданнями [15], [16].

Мова програмування C# надає доступ до таких інструментів, як Entity Framework та інші ORM-рішення, що спрощує зберігання й обробку даних про зустрічі, користувачів і події. Мову програмування C# характеризує висока ступінь узгодженості та наявність великої кількості документації, що допомагає забезпечити прозорість, масштабованість і надійність розробки програмного забезпечення, призначеного для роботи з особистою та корпоративною інформацією [15], [16].

Крім того, використання мови програмування C# дозволяє інтегрувати календарні системи з іншими програмними службами, такими як Google Calendar API або Microsoft Outlook, а також реалізувати управління часом, повідомленнями й іншими функціями без потреби у складних зовнішніх рішеннях. Завдяки знанню таких технологій, як Windows Forms, Windows Presentation Foundation або Blazor, відкривається шлях до створення привабливих і функціональних інтерфейсів [15], [16].

Підтримка з боку Microsoft, активна спільнота розробників і доступність детальної документації полегшують процес навчання, розробки й

усунення помилок, а також сприяють ефективному управлінню проектами. Використання C# забезпечує баланс між швидкістю розробки й якістю програмного продукту, що є важливою умовою для вчасної реалізації складних проєктів [15], [16].

Таким чином, застосування мови програмування C# у розробці програмного забезпечення для календарного планування розглядається як оптимальний варіант завдяки підтримці багатоплатформенності, гнучкості інтеграцій, розширеним можливостям у створенні інтерфейсу, підвищеній безпеці й простоті експлуатації [15], [16].

Обґрунтування вибору мови програмування для розробки програмного забезпечення для календарного планування наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для розробки програмного забезпечення для календарного планування

Критерій порівняння мов програмування	Мова програмування		
	C#	Python	Ruby
Продуктивність виконання	+	+—	+
Розробка графічного інтерфейсу	+	+—	+—
Інтеграція з Windows-сервісами	+	+—	—
Зручність для створення ПЗ для календарного планування	+	—	—
Швидкість розробки	+	+	+—

Отже, для реалізації програмного забезпечення для календарного планування обрано мову програмування C#, яка добре підходить для створення інтерфейсів користувача та має великий набір готових бібліотек і компонентів для роботи з календарями, обробки часу, генерації повідомлень, а також інтеграції з іншими сервісами. Крім того, C# підтримується Microsoft, має велику спільноту розробників та якісну документацію, що спрощує процес

навчання, розробки, пошуку рішень і підтримки проєкту. Також важливо, що ця мова забезпечує хороший баланс між продуктивністю й швидкістю розробки, що дозволяє реалізувати навіть складні функціональні вимоги у прийнятні терміни.

2.2 Вибір середовища розробки для створення програмного забезпечення для календарного планування

В якості середовища розробки для створення програмного забезпечення для календарного планування було обрано середовище Visual Studio [17], [18].

Середовище розробки Visual Studio є ефективним середовищем, що використовується для створення, тестування та вдосконалення програмного забезпечення різного призначення. Це рішення передбачає роботу з багатьма мовами програмування, серед яких можна знайти C#, C++, JavaScript, Python, що дозволяє охоплювати широкий спектр проєктів і рівнів підготовки. Завдяки можливості адаптації інтерфейсу, включаючи різноманітні віджети та функціональні елементи, середовище можна налаштовувати відповідно до конкретних завдань і уподобань користувача [17], [18].

Однією з важливих характеристик цього інструменту є його інтеграція з хмарними платформами, зокрема Azure, що відкриває доступ до управління хмарною інфраструктурою та застосування сучасних практик DevOps. У процесі роботи забезпечується надання рекомендацій, автоматизованого аналізу коду та оцінювання якості без затримок, що сприяє підтримці високого стандарту розробки та полегшує усунення недоліків [17], [18].

Також середовище передбачає наявність корисних засобів для створення інтерфейсів, включаючи інструменти для графічного редагування форм і елементів візуалізації, що є особливо актуальним під час проєктування систем із різноплановими функціями, наприклад, програм для керування календарями. Крім того, Visual Studio підтримує взаємодію із системами

управління проєктами, що сприяє координації колективної роботи, забезпечує обмін ресурсами та контроль за виконанням завдань [17], [18].

Завдяки постійній підтримці з боку Microsoft, ця платформа слугує сучасним і потужним інструментом для реалізації програмних рішень. Вона забезпечує необхідну гнучкість і оперативність як для окремих фахівців, так і для масштабних команд, що займаються розробкою комплексних програмних продуктів [17], [18].

Застосування Visual Studio для створення програмного забезпечення, призначеного для організації щоденних справ, відкриває можливість опанування мови програмування C#, що відзначається простотою у використанні, гнучкістю та високою ефективністю. Завдяки підтримці платформи .NET Framework стає можливим проєктування програм для настільних комп'ютерів, вебсередовища та мобільних пристроїв. Робота в середовищі передбачає використання інструментів на зразок Windows Forms, WPF або MAUI, що сприяє створенню інтерфейсів і робочих процесів у чітко структурованому форматі [17], [18].

Крім розробки, Visual Studio забезпечує процеси тестування, налагодження, моніторингу продуктивності, що дозволяє полегшити всі етапи розробки програмного забезпечення. Інструмент взаємодіє із системами контролю версій, такими як Git, завдяки чому зберігається порядок у проєктах і спрощується їхня координація. Додатково передбачено можливість використання розширень і плагінів, що надає змогу адаптувати середовище під конкретні потреби, зокрема при роботі з інтерфейсами прикладного програмування та репозиторіями [17], [18].

Особливо важливою є функція автоматичного завершення коду та виявлення помилок під час написання, що суттєво прискорює розробку та підвищує її результативність. Інтеграція із хмарними сервісами, такими як Azure, відкриває можливість швидкого розміщення й запуску проєктів безпосередньо у хмарному середовищі [17], [18].

Завдяки таким характеристикам Visual Studio слугує платформою, що підтримує взаємодію різних технологій, об'єднує етапи розробки в єдиний процес і надає необхідні засоби для роботи з комплексними програмами, серед яких — системи планування й календарні застосунки [17], [18].

Обґрунтування вибору середовища розробки для створення програмного забезпечення для календарного планування наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору середовища розробки для створення програмного забезпечення для календарного планування

Критерій порівняння середовищ розробки	Середовища розробки		
	Visual Studio	Eclipse	PyCharm
Зручність засобів налагодження	+	+	—
Інструменти для роботи з базами даних	+	+—	+—
Інтелектуальні підказки	+	+—	+—
Швидкість налаштування середовища	+	+—	+
Розширення функціоналу через плагіни	+	+—	+

Таким чином, для створення програмного забезпечення для календарного планування обрано середовище розробки Visual Studio завдяки його широким можливостям, зручності та інтеграції з мовою програмування C#. Це середовище має потужні вбудовані інструменти для розробки інтерфейсів користувача, що є важливим для створення зручних та функціональних календарних програмних застосунків. Крім того, Visual Studio пропонує засоби автоматизованого тестування, відлагодження, профілювання

продуктивності та інші інструменти, що дозволяють контролювати якість програмного продукту на всіх етапах розробки.

2.3 Висновки за розділом 2

Для реалізації програмного забезпечення для календарного планування обрано мову програмування C#, яка добре підходить для створення інтерфейсів користувача та має великий набір готових бібліотек і компонентів для роботи з календарями, обробки часу, генерації повідомлень, а також інтеграції з іншими сервісами. Крім того, C# підтримується Microsoft, має велику спільноту розробників та якісну документацію, що спрощує процес навчання, розробки, пошуку рішень і підтримки проєкту. Також важливо, що ця мова забезпечує хороший баланс між продуктивністю й швидкістю розробки, що дозволяє реалізувати навіть складні функціональні вимоги у прийнятні терміни.

Для створення програмного забезпечення для календарного планування обрано середовище розробки Visual Studio завдяки його широким можливостям, зручності та інтеграції з мовою програмування C#. Це середовище має потужні вбудовані інструменти для розробки інтерфейсів користувача, що є важливим для створення зручних та функціональних календарних програмних застосунків. Крім того, Visual Studio пропонує засоби автоматизованого тестування, відлагодження, профілювання продуктивності та інші інструменти, що дозволяють контролювати якість програмного продукту на всіх етапах розробки.

3 ОПИС ПРОГРАМИ

3.1 Структура программного обеспечения для календарного планирования

Структуру программного обеспечения для календарного планирования наведено на рис. 3.1.

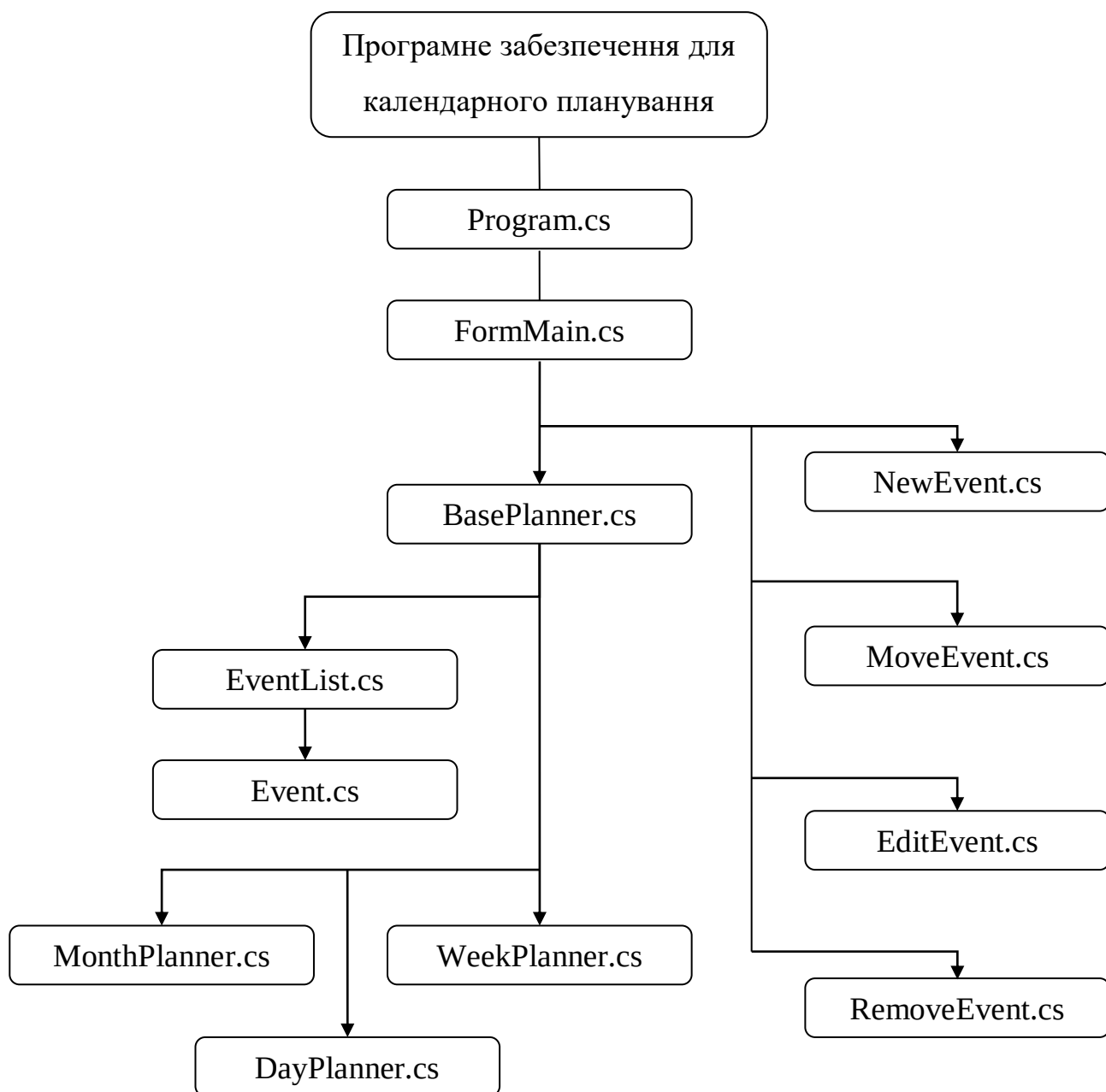


Рисунок 3.1 – Структура программного обеспечения для календарного планирования

Програмне забезпечення для календарного планування побудовано навколо концепції інтерактивної взаємодії користувача із системою, що дозволяє зручно управляти подіями у межах обраного часового проміжку. Фундаментальна архітектура цього програмного продукту базується на принципі централізації управління через головний модуль, який служить відправною точкою для роботи всієї системи. Саме модуль Program.cs відповідає за ініціалізацію програми, передаючи керування головній формі інтерфейсу, втілений у функціоналі FormMain.cs. Ця основна форма виступає як інтерфейсний хаб, через який користувачі взаємодіють з усіма можливостями системи.

Форма головного вікна тісно пов'язана з іншими логічними компонентами, що реалізують ключові операції управління подіями. Наприклад, створення нових записів у календарі забезпечується окремим функціональним блоком, представленим модулем NewEvent.cs, що обробляє введення даних користувача та додає новий запис у загальний перелік подій. Для редагування вже наявних записів призначено модуль EditEvent.cs, завданням якого є відкриття існуючих подій для внесення змін і подальше збереження оновленої інформації. У випадках, коли виникає потреба змінити дату проведення запланованої події, управління переходить до модуля MoveEvent.cs, що відповідає за зміну прив'язки події до календарної дати.

Ключову роль у структурі програмного забезпечення виконує модуль BasePlanner.cs, який забезпечує основну логіку управління списком подій. Саме на його базі вибудовується механізм формування переліків подій відповідно до обраного періоду – чи то одного дня, чи то цілого тижня, чи навіть місяця. Залежно від конкретного запиту користувача, управління делегується відповідному підмодулю: для перегляду подій за день застосовується функціонал BasePlanner.cs, для роботи з подіями протягом тижня – використовується WeekPlanner.cs, а для формування календаря на місяць – модуль MonthPlanner.cs. Таким чином, загальна структура системи

зберігає гнучкість та масштабованість, дозволяючи деталізовано керувати подіями в різних часових масштабах.

Особливе місце у проєкті відведено модулю EventList.cs, який виступає своєрідним сховищем і контролером переліку всіх подій. Цей модуль реалізує клас, що відповідає за додавання, видалення, пошук та оновлення записів у загальній базі подій, забезпечуючи швидкий доступ та маніпуляцію даними. Кожен окремий запис у системі представлений класом, описаним у файлі Event.cs, де визначаються всі атрибути події, включно з датою, часом, описом, категорією тощо.

Узгоджена робота цих модулів створює цілісну систему, в якій процеси створення, редагування, переміщення та перегляду подій інтегруються в єдиний функціональний простір. Така архітектура дозволяє забезпечити як простоту використання, так і розширюваність, що робить програмне забезпечення гнучким інструментом для щоденного планування та управління календарними завданнями. Завдяки чітко визначеним ролям окремих компонентів забезпечується логічний поділ відповідальності між модулями, що сприяє легшому супроводу, оновленню та масштабуванню системи в майбутньому. Програмний продукт, побудований на такій основі, здатен задовольняти потреби користувачів у плануванні подій на різних рівнях деталізації, забезпечуючи при цьому стабільну роботу та інтуїтивно зрозумілий інтерфейс.

Програмне забезпечення для календарного планування базується на складній, багатокомпонентній архітектурі, що охоплює низку взаємопов'язаних модулів, кожен із яких виконує чітко визначені завдання в рамках загальної функціональності системи. Центральним елементом цієї структури виступає модуль Program, що відіграє роль стартового механізму всієї програми. Саме через головний метод класу, Main, відбувається передача керування далі – безпосередньо до модуля FormMain, який, у свою чергу, стає своєрідним диспетчером взаємодії між інтерфейсом користувача й іншими функціональними компонентами системи.

FormMain використовує можливості класу `MonthCalendar`, що належить простору імен `System.Windows.Forms`. Цей компонент відповідає за відображення візуального календаря на місяць і забезпечує інтуїтивний вибір дати через інтерфейс. Саме тут реалізуються ключові методи, які дозволяють здійснювати завантаження подій на основі обраної дати, обробляти їхнє редагування, переміщення та створення. Зокрема, метод `LoadEvents` забезпечує вибірку й завантаження списку подій, що відповідають конкретному дню, використовуючи параметр дати для фільтрації. Обробники `EventEditProcessor`, `EventMoveProcessor` та `EventAddProcessor` реалізують захоплення відповідних подій взаємодії користувача з календарем – редагування, переміщення та додавання, що супроводжується запуском спеціальних форм для редагування або створення. Окремий метод `MonCalDateSelected` відповідає за реакцію програми на вибір дати в календарі, оновлюючи інтерфейс відповідно до вибраного дня.

Фундаментальну роль у логіці представлення даних відіграє клас `BasePlanner`. Він побудований на базі елемента `HiddenGrid`, який є модифікованим варіантом стандартного `DataGridView`. Цей компонент функціонує як невидима сітка, яка забезпечує внутрішню логіку взаємодії із даними, залишаючись при цьому прихованою від користувача. Серед основних методів `BasePlanner` можна виокремити `RefreshEvents`, що здійснює оновлення відображення подій у разі їхньої зміни; `HitTestEvent`, який дозволяє визначити, яка саме подія відповідає координатам курсора; а також `ParametrizeDateTime`, що виконує трансляцію координат у відповідні часові відрізки. Інші методи, такі як обробники `grid_SelectionChanged`, `grid_KeyDown`, `OnMouseDoubleClick` та `DragDropStart`, забезпечують реакцію системи на взаємодію з елементами сітки, включаючи вибір комірки, натискання клавіш, подвійне клацання миші та початок операцій перетягування.

Зберігання структурованої інформації про окремі дні здійснюється завдяки класу `DayPart`, що дозволяє фіксувати назви та часові межі днів. Для

збереження й керування колекціями подій застосовується `EventList`, оснащений методом `SortEvents`, що впорядковує події за часом їхнього початку, гарантуючи правильність хронологічного порядку відображення.

Кожна окрема подія описується об'єктами класу `EventPart`, що включають як основні атрибути – назву події та межі її відображення на вкладці, представлені через властивість `Positions` типу `Rectangle`. Ця властивість має інтерфейс для встановлення та зчитування координат меж. Крім того, атрибут `OutName` та його відповідний `get`-метод забезпечують формування текстового опису події, що включає дату, час початку та описовий рядок.

Для роботи з подіями конкретного дня застосовується клас `DayPlanner`, що розширює `BasePlanner`, додаючи специфічні методи й атрибути. Клас містить приватний атрибут `date`, який фіксує поточну дату, для якої формується список подій. У межах `DayPlanner` метод `ParametrizeDateTime` трансформує координати, отримані від натискання миші, у часові інтервали, базуючись на 15-хвилинному кроці. Метод `FindTimePPositions` відповідає за побудову часової сітки для візуалізації інтервалів, а `FindEventPositions` – за визначення меж, в яких відображатимуться події на сітці, також із 15-хвилинною деталізацією. Малювання інтерфейсу реалізується через метод `OnPaint`, який обробляє подію перерисовування форми. Додатково реалізований метод `findCrosses`, що виконує аналіз перетинів подій за часом, забезпечуючи їхнє коректне розміщення через зміну ширини й горизонтальне зміщення прямокутників, аби уникнути накладання.

Для відображення подій протягом тижня застосовується клас `WeekPlanner`, який також є нащадком `BasePlanner`. Він вводить атрибути `begindate` (початок тижня) та `Date` (цільова дата) і дозволяє за допомогою `get/set`-методів встановлювати й змінювати дату, автоматично коригуючи початковий день тижня. `WeekPlanner` має конструктор, що ініціалізує перелік днів тижня, а також власні версії методів `FindTimePPositions`,

FindEventPositions та OnPaint, адаптовані для відображення подій у тижневому масштабі.

Клас MonthPlanner, подібно до WeekPlanner, наслідує BasePlanner та має свої власні реалізації ключових методів, налаштовані для місячного інтервалу. Завдяки цим адаптаціям MonthPlanner здатний відображати події в межах місяця, враховуючи їхнє просторове розташування на координатній сітці.

Окрему функціональну категорію складають модулі для взаємодії з подіями – NewEvent, EditEvent, MoveEvent та RemoveEvent. Кожен із них відповідає за ініціалізацію відповідних форм і обробку натискання кнопок для збереження змін або скасування дій користувача. Вони слугують інтерфейсним містком між користувачем і системою, дозволяючи додавати, редагувати, переміщати чи видаляти події з календарного плану через діалогові вікна, забезпечуючи при цьому цілісність і актуальність даних.

Загалом, програмне забезпечення для календарного планування являє собою злагоджену систему взаємодії багатьох модулів, де кожен елемент виконує конкретну роль у процесі відображення, редагування, створення та управління подіями у календарному середовищі, а їхня сукупність формує функціонально завершений і користувацько-орієнтований продукт

3.2 Функціонування програмного забезпечення для календарного планування

Функціонування програмного забезпечення для календарного планування подамо за допомогою схеми, зображеної на рис. 3.2.

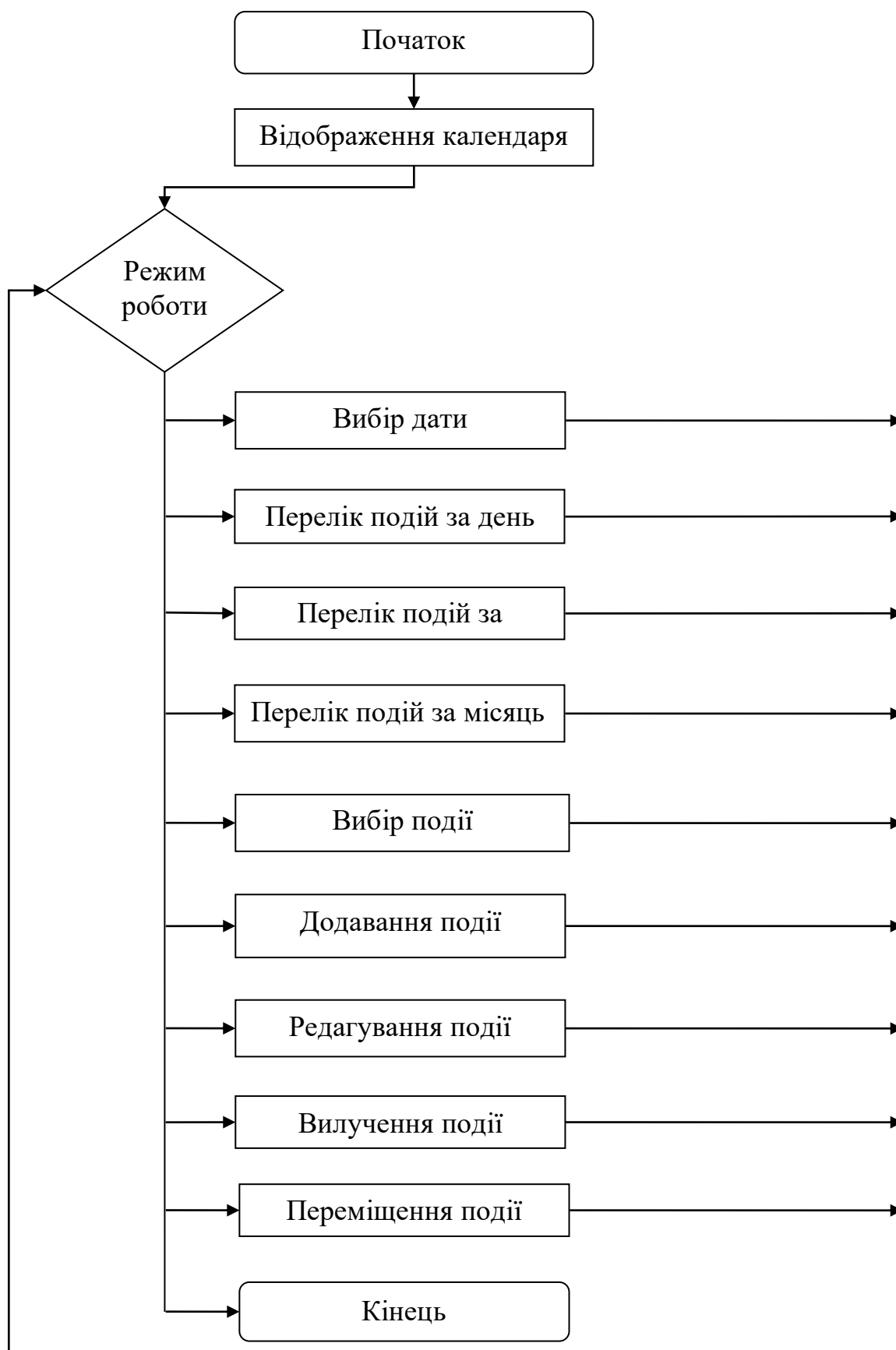


Рисунок 3.2 – Схема функціонування програмного забезпечення для календарного планування

Функціонування програмного забезпечення для календарного планування ґрунтується на інтерактивному підході, що передбачає постійну взаємодію користувача з візуальним інтерфейсом календаря. Від самого запуску системи перед користувачем розгортається графічне відображення календаря, що стає головною платформою для здійснення всіх подальших дій. Через цей календар користувач отримує можливість управляти своїми подіями, переходячи між різними режимами відображення інформації або безпосередньо обираючи окрему подію для детальної роботи. У момент завантаження програми система автоматично фокусує увагу на першій із доступних подій, що дозволяє уникнути обов'язкового попереднього вибору елемента користувачем і миттєво приступити до взаємодії.

Ключовим механізмом взаємодії виступає вибір дати через інтерфейс календаря. Користувач має змогу відмітити лише один конкретний день, але цей одиничний вибір має набагато ширший вплив. Обрана дата визначає не лише перелік подій, які мають місце саме в цей день, а й автоматично задає межі відповідного тижня та місяця, в які входить ця дата. Таким чином, система самостійно корелює вибір користувача із ширшими часовими рамками, забезпечуючи доступ до перегляду як щоденного, так і тижневого чи місячного планування без необхідності вручну змінювати режим.

Користувацький досвід у роботі з цим програмним продуктом вибудовується навколо простоти та логічності переходів між різними масштабами перегляду подій. Після вибору дати календар адаптує своє відображення, підсвічуючи події відповідного дня, при цьому залишаючи можливість одним натисканням миші перемкнутись на огляд тижня або всього місяця. Перемикання між цими режимами відбувається непомітно для користувача, адже програма самостійно підбирає межі тижня та місяця, виходячи з тієї дати, яку було вибрано.

Коли користувач переходить до управління окремою подією, йому надається змога виконувати такі операції, як перегляд деталей, редагування, видалення чи переміщення події на інший день. При цьому система підтримує

безперервний контекст роботи: усі зміни одразу відображаються в обраному режимі перегляду, що дозволяє бачити оновлення в реальному часі. Такий підхід забезпечує високий рівень інтуїтивності, адже користувач не потребує додаткових дій для підтвердження або оновлення списку подій після редагування.

Функціональність програми також підтримує сценарії, коли користувач переглядає не лише події обраного дня, але й бачить їх у контексті всього тижня або місяця, що допомагає ефективніше планувати завантаженість та уникати накладання справ. Система самостійно забезпечує зв'язок між вибраною датою та відповідними календарними межами, дозволяючи легко переключатися між різними рівнями деталізації без необхідності ручного коригування відображення. Таким чином, функціонування програмного забезпечення створює зручне середовище, де вибір однієї дати відкриває широкі можливості керування подіями як у короткостроковій, так і у довгостроковій перспективі, зберігаючи при цьому простоту, швидкість доступу та логічну структуру навігації.

3.3 Розробка бази даних програмного забезпечення для календарного планування

База даних, що лежить в основі програмного забезпечення для календарного планування, є ключовим елементом, який забезпечує збереження, організацію та доступ до інформації про події, користувачів та інші пов'язані дані. Її структура розроблена таким чином, щоб ефективно підтримувати функціональність програми, забезпечуючи швидкий доступ до необхідної інформації та можливість її модифікації.

Схему бази даних програмного забезпечення для календарного планування наведено на рис. 3.3.

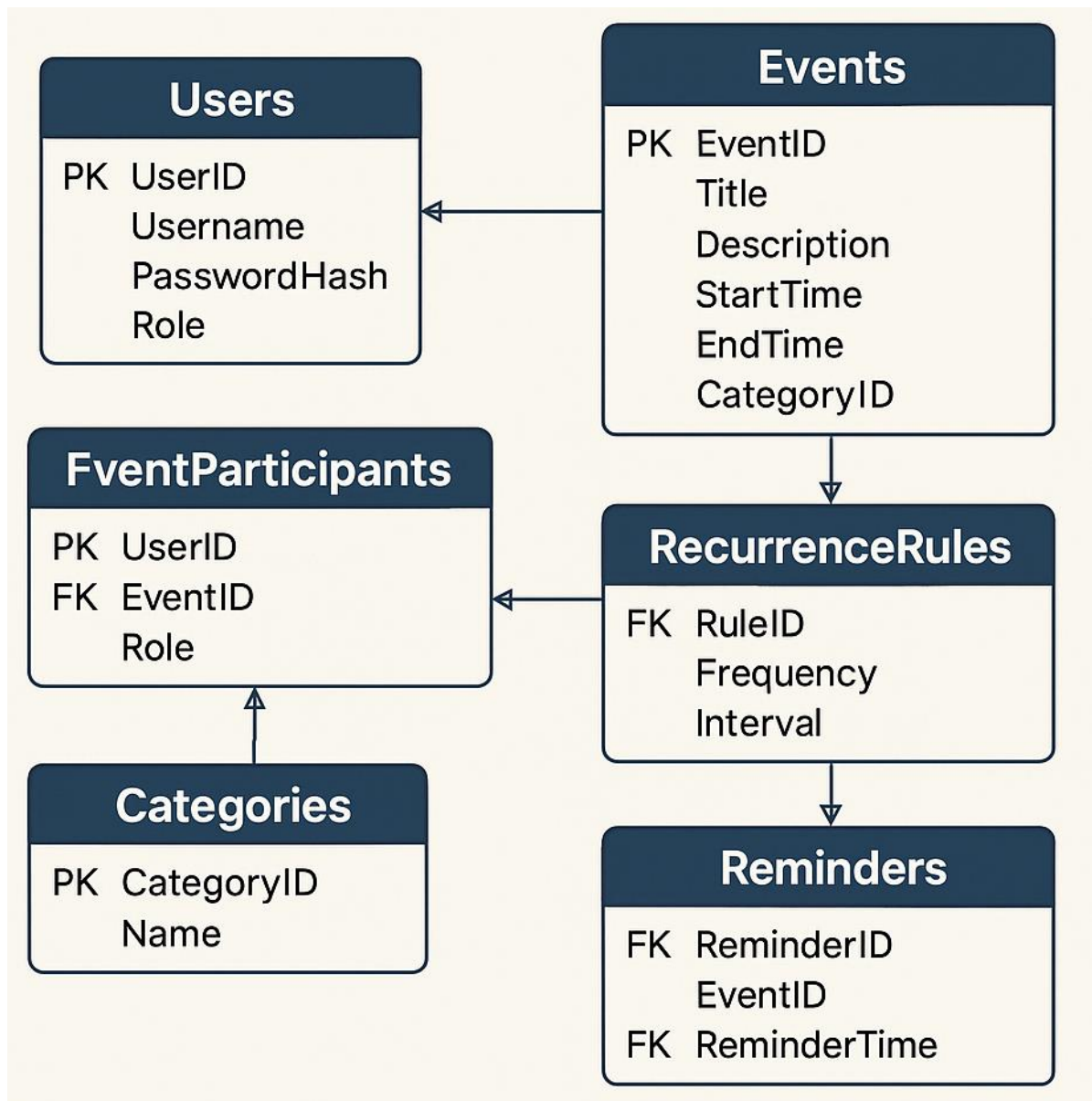


Рисунок 3.3 – Схема бази даних програмного забезпечення для календарного планування

Особливістю цієї бази є те, що вона орієнтована на роботу з часовими даними, а отже, потребує спеціалізованих механізмів для ефективної обробки запитів, пов'язаних із датами і часом. Вона підтримує пошук і фільтрацію подій за різними параметрами, що дозволяє швидко знаходити необхідну інформацію у межах обраного періоду — дня, тижня, місяця. Важливою властивістю є можливість обробки повторюваних подій: замість створення копій для кожної дати, база зберігає одне правило повторення, яке автоматично генерує відповідні події.

У центрі цієї бази даних знаходиться таблиця, яка містить записи про події. Кожен запис включає унікальний ідентифікатор події, назву, опис, дату та час початку і завершення, а також додаткові атрибути, такі як місце проведення, категорія та статус. Ця таблиця пов'язана з іншими таблицями, які містять інформацію про користувачів, дозволяючи відстежувати, хто створив або редагував подію.

Для підтримки різних режимів перегляду календаря, таких як денний, тижневий або місячний, база даних включає механізми, які дозволяють швидко фільтрувати та сортувати події за відповідними критеріями. Це забезпечується за допомогою індексів на полях дати та часу, що дозволяє ефективно виконувати запити до бази даних.

Крім того, база даних підтримує функціональність повторюваних подій, дозволяючи зберігати шаблони повторення та автоматично генерувати відповідні записи для кожного повторення. Це забезпечує гнучкість у плануванні регулярних подій, таких як щотижневі зустрічі або щомісячні звіти.

Безпека даних також є важливим аспектом цієї бази даних. Використовуються механізми контролю доступу, які обмежують можливість перегляду або редагування подій лише для авторизованих користувачів. Це забезпечує конфіденційність інформації та запобігає несанкціонованому доступу.

На схемі бази даних для програмного забезпечення календарного планування показані ключові таблиці та їх зв'язки. Таблиця Користувачі (Users) зберігає інформацією про користувачів (ID, ім'я, пароль). У таблиці Події (Events) зберігаються всі заплановані події. Вона має атрибути: ID події, заголовок, опис, дата початку, дата завершення, категорія. Таблиця Учасники події призначена для зберігання інформації про користувачів, які беруть участь у події. Таблиця Нагадування (Reminders) призначена для налаштування нагадувань про події. Таблиця Категорії (Categories) призначена для класифікації подій (ID категорії, назва категорії), яка пов'язується з

таблицею подій через ID категорії. На схемі позначення PK (Primary Key) являє собою первинний ключ, який однозначно ідентифікує кожен запис у таблиці. Позначення FK (Foreign Key) являє собою зовнішній ключ. Він використовується для встановлення зв'язку між таблицями. Зовнішній ключ у таблиці посилається на первинний ключ іншої таблиці. Наприклад, у таблиці Events є UserID (FK), який вказує на UserID (PK) у таблиці Users. Це означає, що кожна подія пов'язана з конкретним користувачем, який її створив.

Ця структура забезпечує гнучке керування користувачами, подіями, їх класифікацією та нагадуваннями, дозволяючи масштабувати систему та інтегрувати додатковий функціонал.

Загалом, база даних для календарного планування є складною системою, яка забезпечує ефективне зберігання, організацію та доступ до інформації про події, підтримуючи різноманітні функціональні можливості програми та забезпечуючи безпеку даних.

3.4 Проєктування інтерфейсу програмного забезпечення для календарного планування

Проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для календарного планування виконано з урахуванням вимог технічного завдання. При розробленні програмного забезпечення для календарного планування враховані функціональні вимоги до програми:

- підтримка можливості відображення інформації про події користувача у хронологічному порядку;
- підтримка можливості календарного планування шляхом додавання інформації про події користувача;
- можливість корегування інформації про події користувача, зокрема, інформації про початок та завершення події;
- можливість відображення переліку подій користувача на день, тиждень та місяць;

– підтримка можливості переміщення між списками подій за допомогою календаря.

Інтерфейс головної форми програмного забезпечення для календарного планування наведено на рис. 3.4.

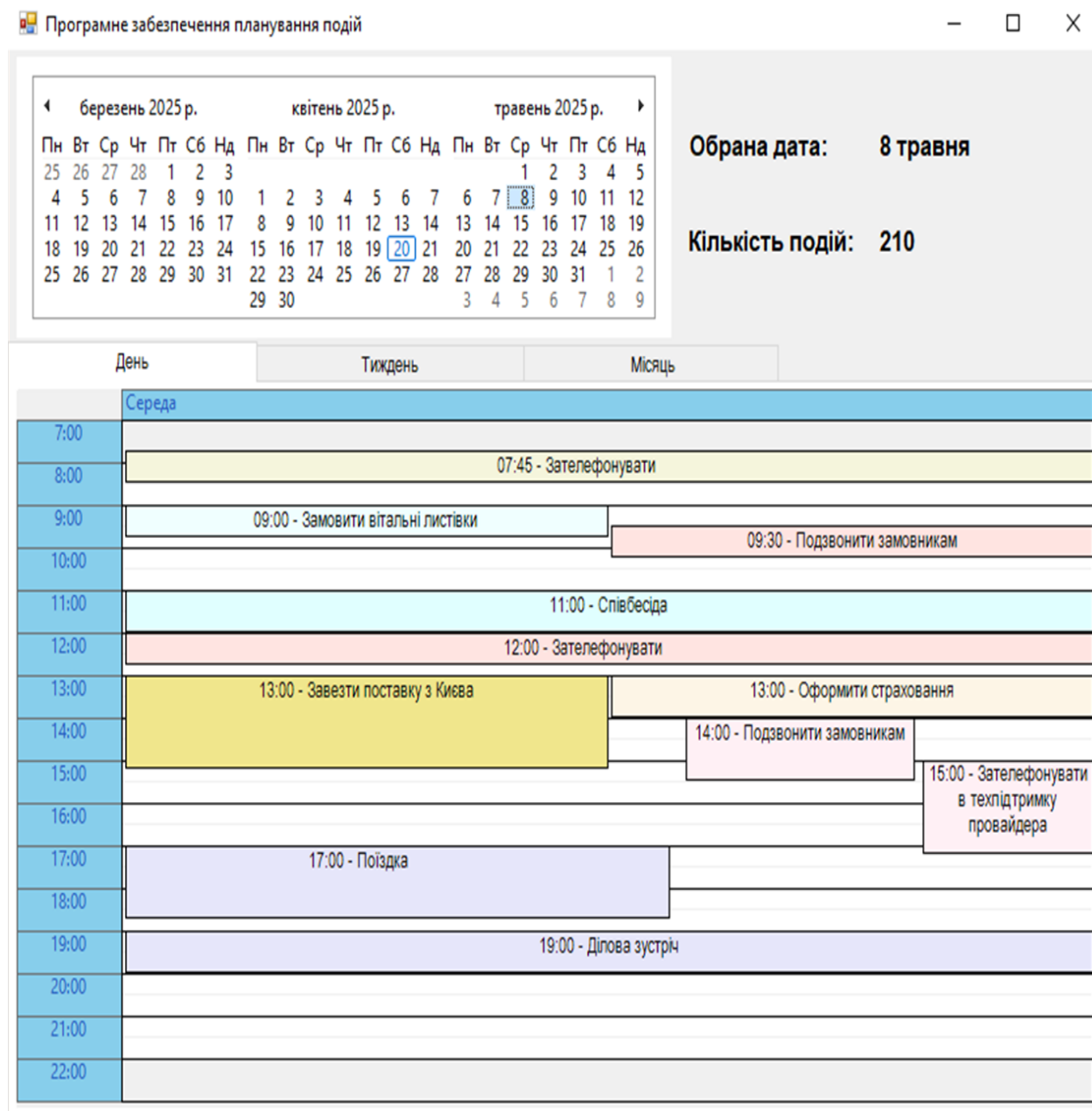


Рисунок 3.4 – Інтерфейс головної форми програмного забезпечення для календарного планування

На рис. 3.5 наведено форму редагування інформації про події.

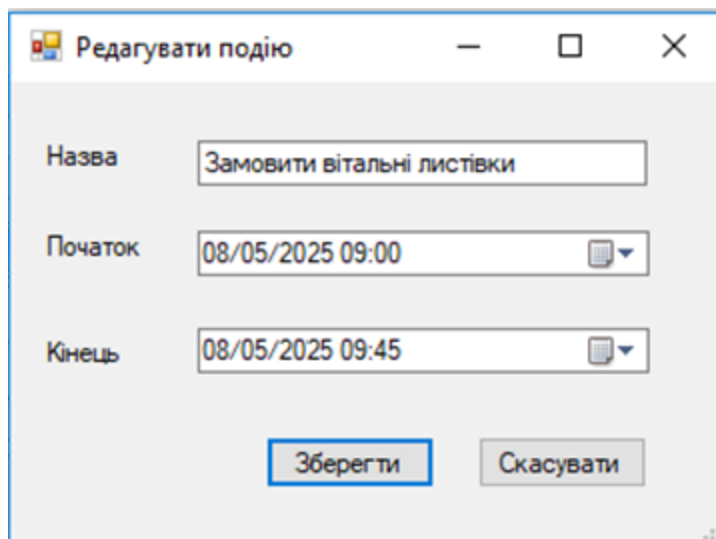


Рисунок 3.5 – Форма редагування інформації про подію у програмному забезпеченні для календарного планування

На рис. 3.6 наведено форму створення нової події.

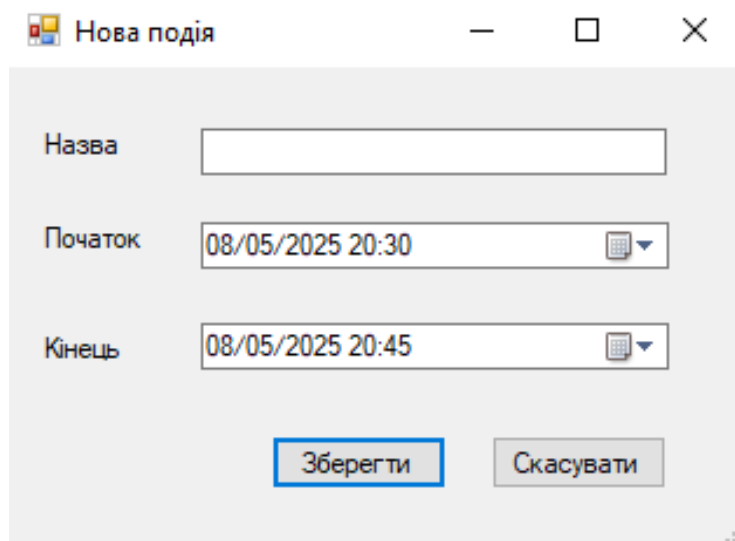


Рисунок 3.6 – Форма створення нової події

На рис. 3.7 наведено приклад вкладки для календарного планування подій на місяць. Аналогічною є вкладка для календарного планування подій на тиждень.

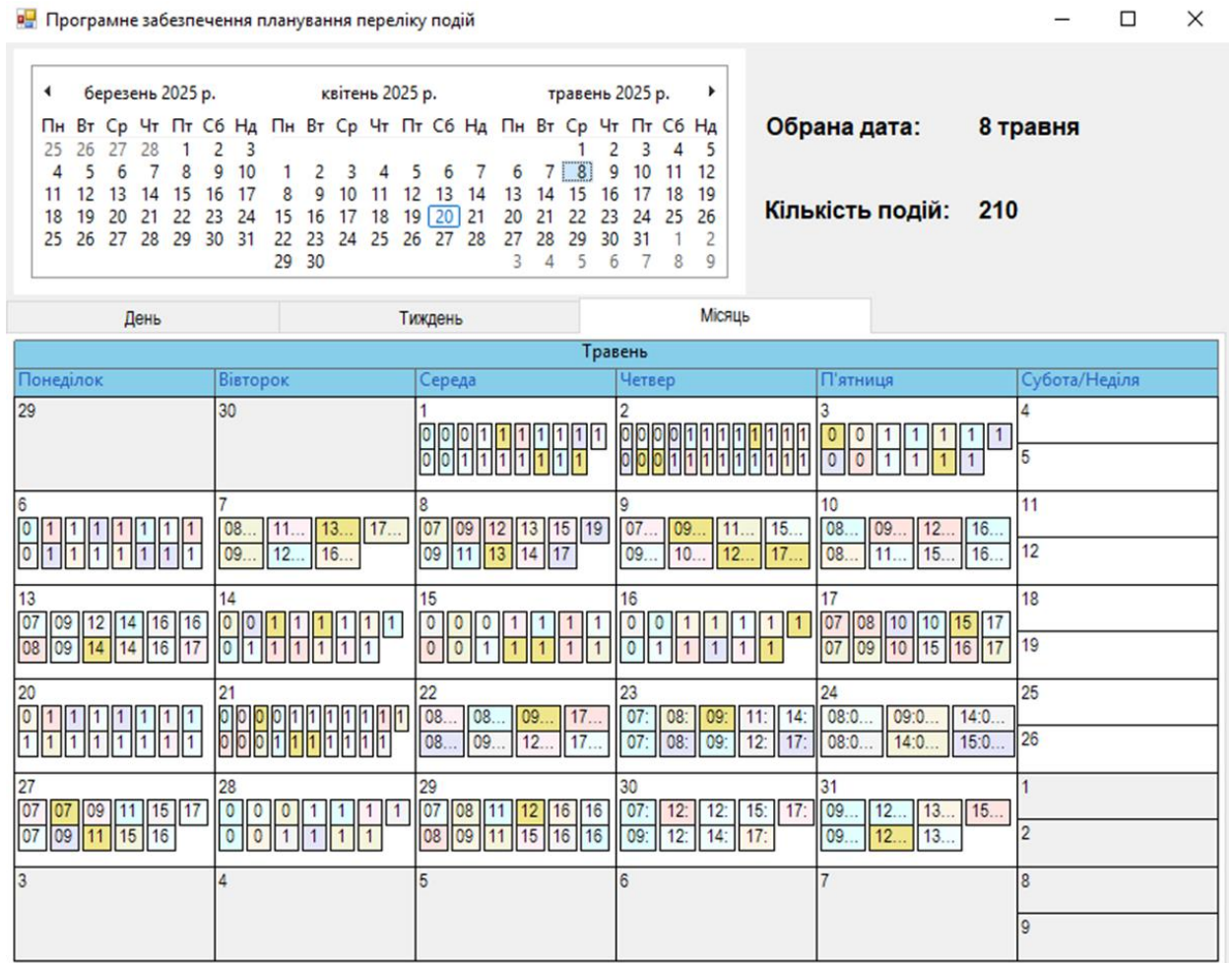


Рисунок 3.7 – Вкладка календарного планування подій на місяць

3.5 Висновки за розділом 3

Розроблено програмне забезпечення для календарного планування.

Запропоновано структуру програмного забезпечення для календарного планування. Визначено, що архітектура програмного продукту для календарного планування базується на принципі централізації управління через головний модуль, який служить відправною точкою для роботи всієї системи. Узгоджена робота програмних модулів створює цілісну систему, в якій процеси створення, редагування, переміщення та перегляду подій інтегруються в єдиний функціональний простір. Така архітектура дозволяє забезпечити як простоту використання, так і розширюваність, що робить

програмне забезпечення гнучким інструментом для щоденного планування та управління календарними завданнями.

Описано функціонування програмного забезпечення для календарного планування. Відзначено, що функціонування програмного забезпечення для календарного планування ґрунтується на інтерактивному підході, що передбачає постійну взаємодію користувача з візуальним інтерфейсом календаря.

Розроблено базу даних програмного забезпечення для календарного планування, яка є ключовим елементом, що забезпечує збереження, організацію та доступ до інформації про події, користувачів та інші пов'язані дані. Її структура розроблена таким чином, щоб ефективно підтримувати функціональність програми, забезпечуючи швидкий доступ до необхідної інформації та можливість її модифікації.

Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для календарного планування.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Програма призначена для календарного планування. Програма написана на мові C#, яка добре підходить для створення інтерфейсів користувача та має великий набір готових бібліотек і компонентів для роботи з календарями, обробки часу, генерації повідомлень, а також інтеграції з іншими сервісами. В якості середовища розробки для створення програмного забезпечення для календарного планування обрано Visual Studio завдяки його широким можливостям, зручності та інтеграції з мовою програмування C#, крім того, це середовище має потужні вбудовані інструменти для розробки інтерфейсів користувача, що є важливим для створення зручних та функціональних календарних програмних застосунків.

Програмне забезпечення для календарного планування забезпечує виконання таких функцій:

- підтримка можливості відображення інформації про події користувача у хронологічному порядку;
- підтримка можливості календарного планування шляхом додавання інформації про події користувача;
- можливість корегування інформації про події користувача, зокрема, інформації про початок та завершення події;
- можливість відображення переліку подій користувача на день, тиждень та місяць;
- підтримка можливості переміщення між списками подій за допомогою календаря.

Для роботи програмного забезпечення для календарного планування потрібні такі технічні та програмні ресурси: комп'ютер з процесором із тактовою частотою не нижче 2.0 ГГц, не менше ніж двох ядерним. Оперативна пам'ять має становити не менше 4 ГБ для базових завдань і не менше 8 ГБ для

одночасної роботи кількох користувачів або інтеграції з іншими системами. Вільне місце на жорсткому диску повинно бути не менше 500 МБ для встановлення програми та базових даних, а для зберігання великих обсягів даних рекомендується мати 1–2 ГБ додаткового простору. З програмного забезпечення необхідна операційна система не нижче Windows 10.

Функціональні характеристики розробленого програмного забезпечення для календарного планування наведено у технічному завданні (додаток А). Фрагмент тексту програмного забезпечення для календарного планування наведено у додатку Б.

4.2 Характеристики програми для календарного планування

Програмне забезпечення для календарного планування базується на принципі централізації управління через головний модуль, який служить відправною точкою для роботи всієї системи. Узгоджена робота програмних модулів створює цілісну систему, в якій процеси створення, редагування, переміщення та перегляду подій інтегруються в єдиний функціональний простір. Така архітектура дозволяє забезпечити як простоту використання, так і розширюваність, що робить програмне забезпечення гнучким інструментом для щоденного планування та управління календарними завданнями.

Програмне забезпечення для календарного планування характеризується наявністю зручного інтерфейсу для створення, редагування та перегляду подій. Підтримується функціонал сповіщень і нагадувань, що дозволяє користувачам своєчасно отримувати інформацію про заплановані заходи. Передбачено збереження даних у базі даних для забезпечення надійності та можливості резервного копіювання. Підтримується багатокористувацький режим для організації спільної роботи над плануванням подій. Програмне забезпечення забезпечує високу швидкість обробки запитів та стабільну роботу.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для запуску програмного забезпечення для календарного планування необхідно запустити виконуваний файл.

4.3.2 Вхідні й вихідні дані

Вхідними даними до програмного забезпечення для календарного планування є інформація, що вводиться користувачем під час створення або редагування подій, це можуть бути назви заходів, дати та час початку й завершення, місця проведення, описи подій, а також параметри повторення подій.

Вихідними даними програмного забезпечення для календарного планування є відображені розклади подій у вигляді календаря, списки запланованих заходів, повідомлення про майбутні події, сформовані сповіщення та нагадування.

4.3.3 Повідомлення

Користувач програмного забезпечення для календарного планування може отримати такі повідомлення: повідомлення про некоректно введені дані, неправильно введені дати, відсутність назви та інших обов'язкових полів даних.

4.4 Виконання програмного забезпечення для календарного планування

Після запуску програмного забезпечення для календарного планування відображається головна форма (рис. 4.1), де користувач може планувати свою

роботу. На головній формі відображено календар, а також перелік подій на поточну дату.

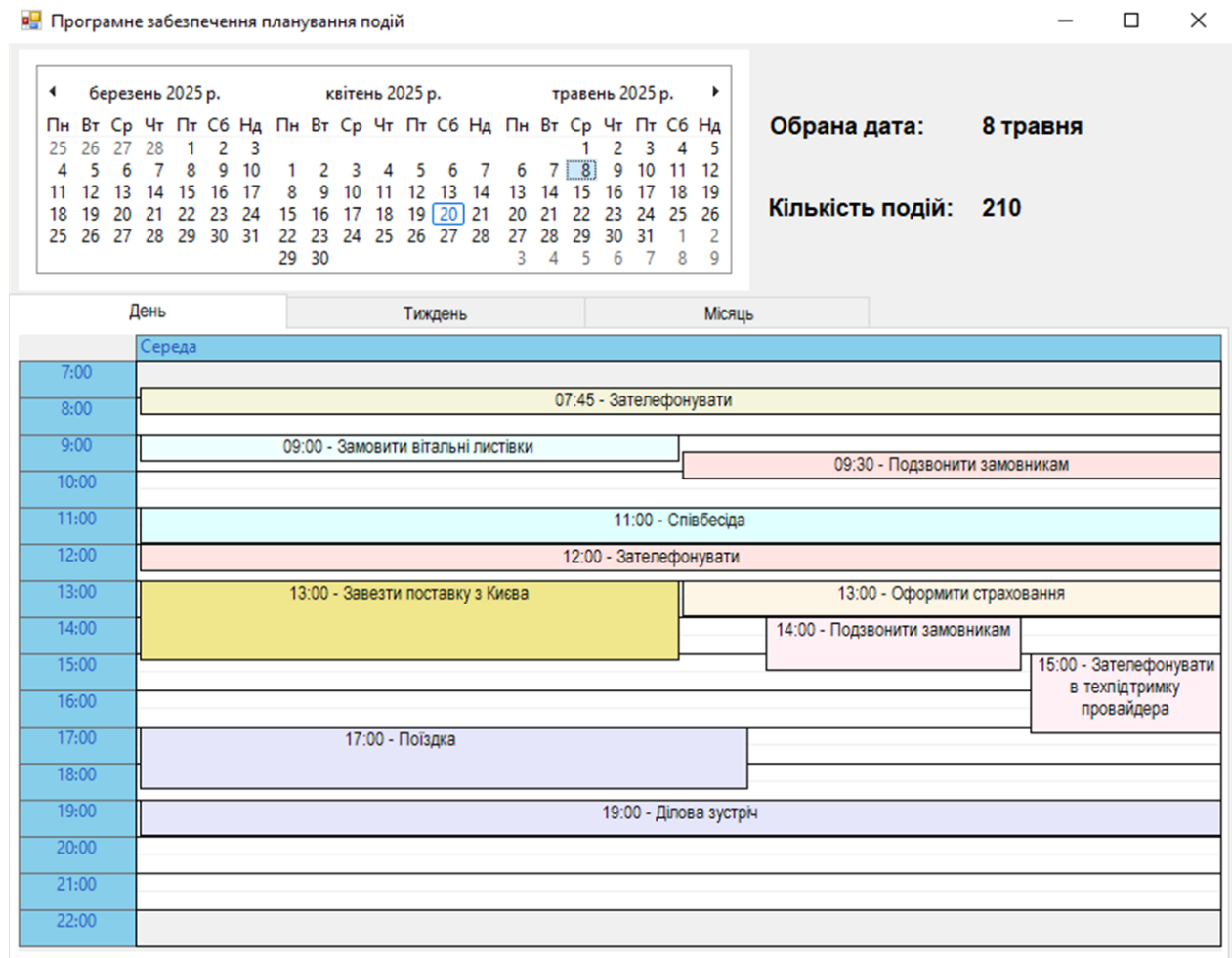


Рисунок 4.1 – Інтерфейс головної форми програмного забезпечення для календарного планування

При подвійному натисканні миші на подію відкривається форма редагування інформації про подію (рис. 4.2), де можна змінити інформацію про неї. Аналогічно для створення нової події (додавання інформації про неї) необхідно обрати у календарі вільний час та ввести параметри події у вікні, аналогічному до того, що зображено на рис. 4.3. Для видалення події необхідно натиснути на неї правою кнопкою миші та у меню обрати пункт Видалення події.

Рисунок 4.2 – Форма редагування інформації про подію у програмному забезпеченні для календарного планування

Рисунок 4.3 – Форма створення нової події

При необхідності отримання інформації про заплановані події на тиждень потрібно перейти до вкладки Тиждень (рис. 4.4), внаслідок чого буде відображено перелік подій тижня, що відповідає даті, обраній у календарі. У вкладці Тиждень у хронологічному порядку відображено події кожного дня обраного тижня.

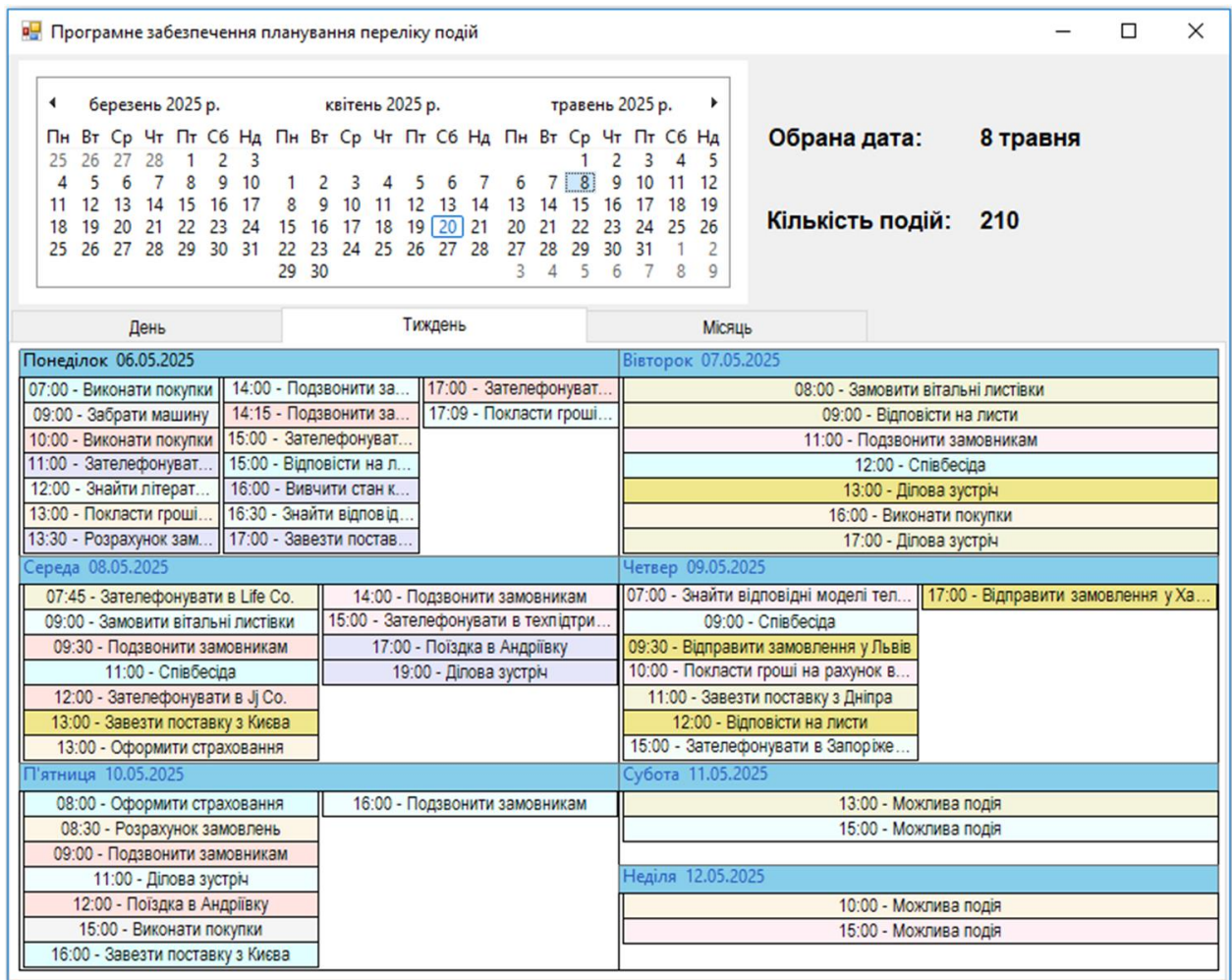


Рисунок 4.4 – Календарне планування подій на тиждень

При необхідності отримання інформації про заплановані події на місяць потрібно перейти до вкладки Місяць (рис. 4.5), внаслідок чого буде відображено перелік подій місяця, що відповідає даті, обраній у календарі. У вкладці Місяць у хронологічному порядку відображено кількість події кожного дня обраного тижня. Для того, щоб отримати повну інформацію про заплановані події необхідно обрати відповідні дату та час.

У процесі роботи програми у деяких випадках користувачу виводяться повідомлення, наприклад, повідомлення про некоректно введені дати, коли дата завершення події не відповідає даті її початку (рис. 4.6).

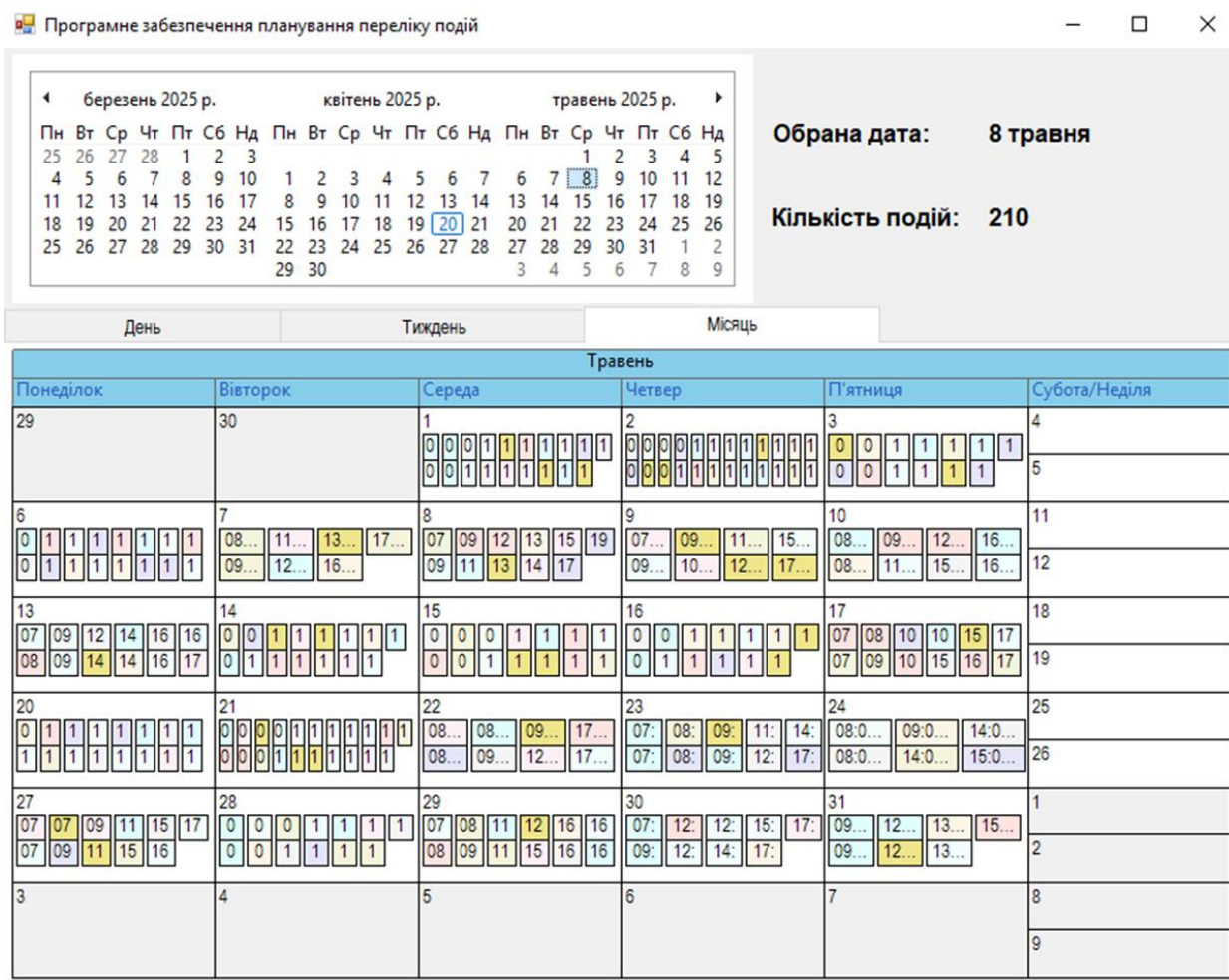


Рисунок 4.5 – Календарне планування подій на місяць

Четвер

Редагувати подію

Назва: Виконати покупки

Початок: 18/04/2025 08:00

Кінець: 18/04/2025 06:15

Кінець повинен бути пізніше початку!

Зберегти Скасувати

Рисунок 4.6 – Повідомлення про некоректно введені дати події

4.5 Тестування програмного забезпечення для календарного планування

Виконано тестування програмного забезпечення для календарного планування.

Виявлено, що програма для календарного планування функціонує правильно та злагоджено. Вона реалізує всі функціональні вимоги та успішно виконує свою основну задачу підтримки процесу управління планами та діяльністю користувача.

Розроблене програмне забезпечення дозволяє забезпечувати автоматизацію процесів, пов'язаних з підтримкою календарного планування.

4.6 Висновки за розділом 4

Описано програмне забезпечення для календарного планування. Виконано тестування розробленого програмного забезпечення для календарного планування. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою процесу календарного планування.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для календарного планування.

Визначено, що програмне забезпечення для календарного планування є інструментом, призначеним для впорядкування та координації заходів, зустрічей і завдань у межах встановленого розкладу. Його використання дозволяє автоматизувати процес управління часом, полегшуючи створення, редагування та відстеження подій. Завдяки такому програмному забезпеченню забезпечується зручність доступу до графіків, що дає змогу своєчасно реагувати на зміни та уникати накладок. Застосування цього типу програмних рішень сприяє підвищенню організованості, зменшенню кількості пропущених заходів і кращій координації діяльності як на індивідуальному, так і на командному рівнях. У результаті забезпечується більш ефективне використання робочого часу та ресурсів, що позитивно впливає на загальну продуктивність.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для календарного планування. Проте деякі програмні засоби для календарного планування можуть мати складний інтерфейс або надмірну кількість функцій, що може стати бар'єром для користувачів, які не мають достатнього досвіду роботи з такими програмними системами, або призводити до перевантаження інформацією. Іншою проблемою використання таких програмних засобів є питання конфіденційності та захисту даних, оскільки багато сервісів для календарного планування зберігають дані користувачів у хмарі, що підвищує ризик витоку інформації, особливо якщо календар містить особисті чи корпоративно важливі дані. Ще одним недоліком є обмежена сумісність між різними програмними платформами та сервісами. Не всі програмні засоби для календарного планування легко інтегруються з іншими програмами, через що

виникають проблеми з синхронізацією даних між різними пристроями чи застосунками. Тому актуальною є розробка програмного забезпечення для календарного планування.

Сформульовано функціональні вимоги до програмного забезпечення для календарного планування.

Для реалізації програмного забезпечення для календарного планування обрано мову програмування C#, яка добре підходить для створення інтерфейсів користувача та має великий набір готових бібліотек і компонентів для роботи з календарями, обробки часу, генерації повідомлень, а також інтеграції з іншими сервісами. Крім того, C# підтримується Microsoft, має велику спільноту розробників та якісну документацію, що спрощує процес навчання, розробки, пошуку рішень і підтримки проєкту. Також важливо, що ця мова забезпечує хороший баланс між продуктивністю й швидкістю розробки, що дозволяє реалізувати навіть складні функціональні вимоги у прийнятні терміни.

Для створення програмного забезпечення для календарного планування обрано середовище розробки Visual Studio завдяки його широким можливостям, зручності та інтеграції з мовою програмування C#. Це середовище має потужні вбудовані інструменти для розробки інтерфейсів користувача, що є важливим для створення зручних та функціональних календарних програмних застосунків. Крім того, Visual Studio пропонує засоби автоматизованого тестування, відлагодження, профілювання продуктивності та інші інструменти, що дозволяють контролювати якість програмного продукту на всіх етапах розробки.

Розроблено програмне забезпечення для календарного планування.

Запропоновано структуру програмного забезпечення для календарного планування. Визначено, що архітектура програмного продукту для календарного планування базується на принципі централізації управління через головний модуль, який служить відправною точкою для роботи всієї системи. Узгоджена робота програмних модулів створює цілісну систему, в

якій процеси створення, редагування, переміщення та перегляду подій інтегруються в єдиний функціональний простір. Така архітектура дозволяє забезпечити як простоту використання, так і розширюваність, що робить програмне забезпечення гнучким інструментом для щоденного планування та управління календарними завданнями.

Описано функціонування програмного забезпечення для календарного планування. Відзначено, що функціонування програмного забезпечення для календарного планування ґрунтується на інтерактивному підході, що передбачає постійну взаємодію користувача з візуальним інтерфейсом календаря.

Розроблено базу даних програмного забезпечення для календарного планування, яка є ключовим елементом, що забезпечує збереження, організацію та доступ до інформації про події, користувачів та інші пов'язані дані. Її структура розроблена таким чином, щоб ефективно підтримувати функціональність програми, забезпечуючи швидкий доступ до необхідної інформації та можливість її модифікації.

Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для календарного планування.

Виконано тестування розробленого програмного забезпечення для календарного планування. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою процесу календарного планування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Calendar App: The Ultimate Step-by-Step Guide to Develop a Calendar App [Electronic resource]. – Access mode: <https://www.addwebsolution.com/blog/how-to-create-a-calendar-app>.
2. Key Things You Should Know When Making a Calendar App [Electronic resource]. – Access mode: <https://weelorum.com/blog/key-things-you-should-know-when-making-a-calendar-app/>.
3. How to Create a Scheduling Software: Step-by-Step Guide [Electronic resource]. – Access mode: <https://vettedsoftware.com/how-to-create-a-scheduling-software/>.
4. How to make a calendar app [Electronic resource]. – Access mode: <https://www.jotform.com/blog/how-to-make-a-calendar-app/>.
5. Project Calendar Mastery: Tools Comparison & Expert Guide [Electronic resource]. – Access mode: <https://blog.virtosoftware.com/project-calendar-guide/>.
6. Top 20 Calendar Software to Keep Your Schedule Tidy at All Times [Electronic resource]. – Access mode: <https://startinfinity.com/blog/calendar-software>.
7. The 15 Best Digital Planners [Electronic resource]. – Access mode: <https://friday.app/p/digital-planners>.
8. Best Digital Planner Apps [Electronic resource]. – Access mode: <https://clickup.com/blog/digital-planner-apps/>.
9. The 5 best calendar apps [Electronic resource]. – Access mode: <https://zapier.com/blog/best-calendar-apps/>.
10. Best Project Management Calendar Tools For Visual Scheduling [Electronic resource]. – Access mode: <https://thedigitalprojectmanager.com/tools/project-management-calendar/>.
11. Best Online Planner Apps & Digital Planners [Electronic resource]. – Access mode: <https://upbase.io/blog/best-online-planner-apps-and-digital-planners/>.

12. NTask [Electronic resource]. – Access mode: <https://www.ntaskmanager.com/>.
13. Calendly [Electronic resource]. – Access mode: <https://calendly.com/>.
14. Notion [Electronic resource]. – Access mode: <https://www.notion.com/>.
15. What is C#? An Introduction to C# Programming Language [Electronic resource]. – Access mode: <https://learnsic.com/blog/what-is-c-an-introduction-to-c-programming-language>.
16. C# Attributes [Electronic resource]. – Access mode: https://www.tutorialspoint.com/csharp/csharp_attributes.htm.
17. Introduction to Visual Studio [Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/introduction-to-visual-studio/>.
18. Discover Visual Studio [Electronic resource]. – Access mode: <https://learn.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2022#discover-visual-studio>.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення може використовуватися для календарного планування.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмне забезпечення для календарного планування», затверджене наказом Національного університету «Запорізька політехніка» № 208 від 28 квітня 2025 р.

A.2 Призначення розробки

Програмний продукт призначений для календарного планування.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення для календарного планування забезпечує виконання таких функцій:

- підтримка можливості відображення інформації про події користувача у хронологічному порядку;
- підтримка можливості календарного планування шляхом додавання інформації про події користувача;
- можливість корегування інформації про події користувача, зокрема, інформації про початок та завершення події;
- можливість відображення переліку подій користувача на день, тиждень та місяць;

– підтримка можливості переміщення між списками подій за допомогою календаря.

А.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення для календарного планування повинен бути зручним для користувачів. Повинна бути забезпечена можливість роботи в візуальному режимі.

А.3.3 Вимоги до надійності

Програмне забезпечення для календарного планування повинно забезпечити надійне функціонування.

А.3.4 Умови експлуатації

Для експлуатації програмного забезпечення для календарного планування необхідна наявність персонального комп'ютера.

А.3.5 Вимоги до складу та параметрів технічний засобів

Для роботи програмного забезпечення для календарного планування потрібні такі технічні та програмні ресурси: комп'ютер з процесором із тактовою частотою не нижче 2.0 ГГц, не менше ніж двох ядерним. Оперативна пам'ять має становити не менше 4 ГБ для базових завдань і не менше 8 ГБ для одночасної роботи кількох користувачів або інтеграції з іншими системами. Вільне місце на жорсткому диску повинно бути не менше 500 МБ для встановлення програми та базових даних, а для зберігання великих обсягів

даних рекомендується мати 1–2 ГБ додаткового простору. З програмного забезпечення необхідна операційна система не нижче Windows 10.

А.3.6 Вимоги до маркування і пакування

Програма для календарного планування може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва програми – «Програмне забезпечення для календарного планування».

А.4 Вхідні дані до роботи

Вхідними даними до програмного забезпечення для календарного планування є інформація, що вводиться користувачем під час створення або редагування подій, це можуть бути назви заходів, дати та час початку й завершення, місця проведення, описи подій, а також параметри повторення подій.

Вихідними даними програмного забезпечення для календарного планування є відображені розклади подій у вигляді календаря, списки запланованих заходів, повідомлення про майбутні події, сформовані сповіщення та нагадування.

ДОДАТОК Б
Фрагмент тексту програми

```

public class DayPlnr : BasePlnr
{
    public DayPlnr()
    {
        MnPrthHrs = true;
        DayMd = true;
    }

    private readonly List<MnLnPrt> hrMnLns = new
List<MnLnPrt>();

    protected override DtTm? PrmtrzDtTm(int x, int y)
    {
        foreach (DHrPrt d in this.DPrts)
        {
            if (d.Pstns.Contains(x, y))
            {
                foreach (var oh in d.Hrs)
                {
                    if (oh.Pstns.Contains(x, y))
                    {
                        int mtm = 0;
                        if (oh.Pstns1.Contains(x, y))
                            mtm = 15;
                        if (oh.Pstns2.Contains(x, y))
                            mtm = 30;
                        if (oh.Pstns3.Contains(x, y))
                            mtm = 45;
                        return new DtTm(d.Date.Year,
d.Date.Month, d.Date.Day, oh.Hr, mtm, 0);
                    }
                }
            }
            return d.Date;
        }
    }

    return null;
}

protected override void DrgDrpEnd(Evnt ev, int x, int y)
{
    var d = PrmtrzDtTm(x, y);
    if (d != null)
    {
        StartEvntMode(ev, d.Value);
    }
}

```

```

    }
}

protected override void FndTmPPstns(Graphics g)
{
    if (Width == 0 || Height == 0)
        return;

    DPrts.Clear();
    hrMnLns.Clear();

    int DsNm = 1;

    if (!DayMd)
    {
        DsNm = 5;
        if (Date.WkDy != WkDy.Mndg)
        {
            if (Date.WkDy == WkDy.Sndg)
            {
                Date = Date.AdhdDays(1);
            }
            else if (Date.WkDy == WkDy.Strdg)
            {
                Date = Date.AdhdDays(2);
            }
            else
            {
                DtTm dtvl = Date;
                while (dtvl.WkDy != WkDy.Mndg)
                {
                    dtvl = dtvl.AdhdDays(-1);
                }
                Date = dtvl;
            }
        }
    }

    for (int i = 0; i < DsNm; i++)
    {
        DHrPrt dhgp = new DHrPrt();
        dhgp.Date = Date.AdhdDays(i);
        dhgp.IsInMonth = true;
        this.DPrts.Adhd(day);
    }

    int HrNm = 0;

```

```

for (int j = 0; j < DPrts.Count; j++)
{
    DHrPrt dhgp = DPrts[j] as DHrPrt;
    if (dhgp != null)
    {
        for (int i = 0; i < 24; i++)
        {
            if (MnPrthrs)
            {
                if (i >= (BegHr - 1) && i < (StopHr +
1))
                {
                    HrPart hp = new HrPart();
                    hp.IsSHr = (i >= BegHr && i <
StopHr);

                    hp.Hr = i;
                    dhgp.Hrs.Adhd(Hr);
                }
            }
            else
            {
                HrPart hp = new HrPart();
                hp.IsSHr = (i >= BegHr && i <
StopHr);

                hp.Hr = i;
                dhgp.Hrs.Adhd(Hr);
            }
        }
        if (j == 0)
        {
            HrNm = d.Hrs.Count;
        }
    }

    int tWdth = 0;
    if (this.Width > 120)
    {
        tWdth = 80;
    }

    int dMnLineHeight =
(int) (BasePlnr.MnLn.GetTextInfo(g, this.Font).Height * 1.2);

    int ohHght = ((Height - 1 - dMnLineHeight) / HrNm);
    int x = tWdth;
    int ohWdth = (Width - tWdth - 1) / DsNm;

```

```

int DelHght = (ohHght) / 4;

for (int j = 0; j < DPrts.Count; j++)
{
    DHrPrt dhgp = DPrts[j] as DHrPrt;

    if (dhgp != null)
    {
        int y = dMnLineHeight;
        dhgp.Pstns = new Rectangle(x, 0, ohWdth,
ohHght * dhgp.Hrs.Count);
        if (Date.WkDy == WkDy.Sndg)
            dhgp.Name = "Неділя";
        else if (Date.WkDy == WkDy.Strdg)
            dhgp.Name = "Субота";
        else if (Date.WkDy == WkDy.Frdh)
            dhgp.Name = "П'ятниця";
        else if (Date.WkDy == WkDy.Thrsdg)
            dhgp.Name = "Четвер";
        else if (Date.WkDy == WkDy.Wdnsdg)
            dhgp.Name = "Середа";
        else if (Date.WkDy == WkDy.Tsdg)
            dhgp.Name = "Вівторок";
        else dhgp.Name = "Понеділок";

        dhgp.NamePstns = new Rectangle(x, 0, ohWdth,
dMnLineHeight);
        dhgp.EvRectPstns = new
Rectangle(dhgp.Pstns.X, dhgp.Pstns.Y, dhgp.Pstns.Width,
dhgp.Pstns.Height);

        foreach (HrPart hp in day.Hrs)
        {
            hp.Pstns = new Rectangle(x, y, ohWdth,
ohHght);
            hp.Pstns0 = new Rectangle(x, y, ohWdth,
DelHght);
            hp.Pstns1 = new Rectangle(x, y +
hp.Pstns00.Height, ohWdth,
DelHght);
            hp.Pstns2 = new Rectangle(x, y +
hp.Pstns00.Height + hp.Pstns15.Height,
ohWdth,
DelHght);
            hp.Pstns3 = new Rectangle(x,

```

```

                                                                    y +
hp.Pstns0.Height + hp.Pstns1.Height +
hp.Pstns2.Height, ohWdth,
                                                                    DelHght);

        if (j == 0)
        {
            MnLnPrt mlp = new MnLnPrt();
            mlp.Pstns = new Rectangle(0, y,
tWdth, ohHght);
            mlp.Name = string.Format("{00}:00",
hp.Hr);
            mlp.Adhd(MnLn);
        }
        y += ohHght;
    }
    x += ohWdth;
}
PsTmeP = true;
}

protected override void FindEvntPstns(Graphics graphics)
{
    foreach (DHrPrt dhgp in this.DPrts)
    {
        dhgp.Evnts.Clear();
        foreach (HrPart hp in dhgp.Hrs)
        {
            hp.Evnts.Clear();
        }

        foreach (Evnt ev in Evnts)
        {
            if (ev.BeginDtTm.Year == dhgp.Date.Year &&
ev.BeginDtTm.YearDay == dhgp.Date.YearDay)
            {
                EvntHrPart evp = new EvntHrPart(ev);
                dhgp.Evnts.Adhd(evp);
                foreach (HrPart hp in dhgp.Hrs)
                {
                    if (evp.BeginDtTm.Hr == hp.Hr)
                    {
                        hp.Evnts.Adhd(evp);
                    }
                }
            }
        }
    }
}

```

```

    }
}
foreach (EvtHrPart egv1 in dhgp.Evnts)
{
    foreach (EvtHrPart egv2 in dhgp.Evnts)
    {
        if (egv1 != egv2
            &&
            egv1.Evnt.EndhdtTm !=
egv2.Evnt.BeginDtTm
            &&
            egv1.Evnt.BeginDtTm !=
egv2.Evnt.EndhdtTm
            &&
            (
                egv2.Evnt.BeginDtTm ==
egv1.Evnt.BeginDtTm
                ||
                egv2.Evnt.EndhdtTm ==
egv1.Evnt.EndhdtTm
                ||
                (egv2.Evnt.BeginDtTm <
egv1.Evnt.EndhdtTm
                && egv2.Evnt.BeginDtTm >
egv1.Evnt.BeginDtTm)
                ||
                (egv2.Evnt.EndhdtTm <
egv1.Evnt.EndhdtTm
                && egv2.Evnt.EndhdtTm >
egv1.Evnt.BeginDtTm)
                ||
                (egv1.Evnt.EndhdtTm <
egv2.Evnt.EndhdtTm
                && egv1.Evnt.EndhdtTm >
egv2.Evnt.BeginDtTm)
            )
        )
        {
            egv1.Crossed = true;
            egv1.Crossed = true;
            if (!egv1.Cross.Contains(egv2))
                egv1.Cross.Adhd(egv2);
            if (!egv2.Cross.Contains(egv1))
                egv2.Cross.Adhd(egv1);
        }
    }
}

```

```

for (int i = 0; i < dhgp.Hrs.Count; i++)
{
    HrPart hp = dhgp.Hrs[i];
    int Yev = hp.Pstns.Y;
    if (hp.Evnts.Count > 0)
    {
        int Xev = hp.Pstns.X;

        for (int j = 0; j < hp.Evnts.Count; j++)
        {
            EvntHrPart ev = hp.Evnts[j];

            int bgMn = ev.Evnt.BeginDtTm.Minute;
            if (bgMn >= 0 && bgMn < 15)
            {
                ev.BgDv = 0;
            }
            else if (bgMn >= 15 && bgMn < 30)
            {
                ev.BgDv = 1;
            }
            else if (bgMn >= 30 && bgMn < 45)
            {
                ev.BgDv = 2;
            }
            else if (bgMn >= 45 && bgMn < 60)
            {
                ev.BgDv = 3;
            }

            TmIntrv itv =
ev.Evnt.DateEnd.Minus(ev.Evnt.BeginDtTm);
            int meh = ((itv.Hrs * 60 +
itv.Minutes) / 15) * hp.Pstns0.Height;
            int ey = evY + (ev.BgDv *
hp.Pstns0.Height);

            int ew = dhgp.Pstns.Width;

            if (ev.Crossed)
            {
                ew = ew / dhgp.Evnts.Count;
            }
            ev.Pstns = new Rectangle(evtX, ey,
ew, meh);

            evhX += evWdth;
        }
    }
}

```

```

        }
    }

    for (int i = 0; i < 5; i++)
    {
        findCrosses(dhgp);
    }
}
PstnsValidEvnt = true;
}

private static void findCrosses(DHrPrt dhgp)
{
    if (dhgp.Evnts.Count > 0)
    {
        for (int j = 0; j < dhgp.Evnts.Count; j++)
        {
            EvntHrPart eghp = dhgp.Evnts[j] as
EvntHrPart;

            if (eghp.Crossed)
            {
                foreach (EvntHrPart cev in eghp.Cross)
                {
                    if (!cev.Pstns.IsEmpty &&
cev.Pstns.SavePos(new
Rectangle(eghp.Pstns.X,
eghp.Pstns.Y,
eghp.Pstns.Width,
eghp.Pstns.Height))

                    )
                    {
                        if (eghp.Pstns.Width >=
cev.Pstns.Width)

                        {
                            if (eghp.Pstns.X >=
cev.Pstns.X)

                            {
                                eghp.Pstns = new
Rectangle(
                                eghp.Pstns.X +
                                eghp.Pstns.Width / 4,
                                eghp.Pstns.Y,
                                eghp.Pstns.Width -
                                eghp.Pstns.Width / 4,
                                eghp.Pstns.Height);
                            }
                        }
                    }
                }
            }
        }
    }
}
else

```

```

        {
            eghp.Pstns = new
Rectangle(
            eghp.Pstns.X,
            eghp.Pstns.Y,
            eghp.Pstns.Width -
            eghp.Pstns.Height);
        }
    }

    if (eghp.Pstns.X + eghp.Pstns.Width >
dhgp.Pstns.X + dhgp.Pstns.Width)
    {
        Rectangle rect = eghp.Pstns;
        rect.Width = dhgp.Pstns.X +
dhgp.Pstns.Width - eghp.Pstns.X;
        eghp.Pstns = rect;
    }
}

}

}

private DtTm date;
public DtTm Date
{
    get
    {
        return date;
    }
    set
    {
        date = value;
        PstnsValidEvnt = false;
        PsTmeP = false;
    }
}

public int BegHr { get; set; }
public int StopHr { get; set; }
public bool MnPrtHrs { get; set; }
private bool DayMd;
public bool DayMd

```

```

    {
        get
        {
            return DayMd;
        }
        set
        {
            DayMd = value;
            PstnsValidEvnt = false;
            PsTmeP = false;
        }
    }

protected override void OnPaint(EvntArgs ev)
{
    base.OnPaint(ev);

    foreach (MnLnPrt mpl in hrMnLns)
    {
        BasePlnr.MnLn.DrawBox(ev.Graphics, Font,
mpl.Pstns, mpl.Name, true, BorderInfo.Default,
TextAlignment.Right);
    }

    for (int j = 0; j < DPrts.Count; j++)
    {
        DHrPrt dhgp = DPrts[j] as DHrPrt;
        if (dhgp!= null)
        {
            BasePlnr.MnLn.DrawBox(ev.Graphics, Font,
dhgp.NamePstns, dhgp.Name);

            foreach (HrPart hp in dhgp.Hrs)
            {
                if (hp.IsSHr)
                {
                    BasePlnr.Item.DrawBox(ev.Graphics,
Font, hp.Pstns);

BasePlnr.SimpleItem.DrawLine(ev.Graphics, hp.Pstns2.X + 1,
hp.Pstns2.Y,

hp.Pstns2.X + hp.Pstns2.Width - 2,

hp.Pstns2.Y);

                }
                else
                {

```



```

        }
    }
}

```

```

protected override void OnLoad(EvntArgs ev)
{
    base.OnLoad(ev);

    DblBffrd = true;

    DtTm bgndt = DtTm.Now;
    EvntList Evnts = CreateEvntList(bgndt);

    wkTbl.Date = bgndt;
    wkTbl.Evnts = Evnts;
    mnTbl.Date = bgndt;
    mnTbl.Evnts = Evnts;
    dTbl.Date = bgndt;
    dTbl.Evnts = Evnts;

    EvntNumLabel.Text = dTbl.Evnts.Count;
    ChosenDateLabel.Text = bgndt.ToLongDateString();

    wkTbl.EvntCreate += EvntAdhdProcessor;
    mnTbl.EvntCreate += EvntAdhdProcessor;
    dTbl.EvntCreate += EvntAdhdProcessor;

    wkTbl.EvntMove += EvntMoveProcessor;
    mnTbl.EvntMove += EvntMoveProcessor;
    dTbl.EvntMove += EvntMoveProcessor;

    wkTbl.EvntEdit += EvntEditProcessor;
    mnTbl.EvntEdit += EvntEditProcessor;
    dTbl.EvntEdit += EvntEditProcessor;
}

```

```

private void EvntEditProcessor(object sender, EvntArgs
ev)
{
    using (EditEvnt egd = new EditEvnt())
    {
        egd.EvntBgndt = ev.Evnt.BeginDtTm;
    }
}

```

```

        egd.EvntEndhdate = ev.Evnt.EndhdtTm;
        egd.EvntTitle = ev.Evnt.Name;
        DialogResult dr = egd.ShowDialog();
        if (dr == DialogResult.OK)
        {
            ev.Evnt.BeginDtTm = dr.EvBeginDtTm;
            ev.Evnt.EndhdtTm = dr.EvEndhdtTm;
            ev.Evnt.Name = dr.EvName;

            wkTbl.RefreshEvnts();
            mnTbl.RefreshEvnts();
            dTbl.RefreshEvnts();

            wkTbl.Invalidate();
            mnTbl.Invalidate();
            dTbl.Invalidate();
        }
    }
}

private void EvntMoveProcessor(object sender, EvntArgs
ev)
{
    using (MoveEvnt egd = new MoveEvnt())
    {
        egd.EvntPrevBgndt = ev.Evnt.BeginDtTm;
        egd.EvntPrevEndhdate = ev.Evnt.EndhdtTm;
        egd.EvntTitle = ev.Evnt.Name;
        if (ev.NewDate != null)
        {
            egd.EvntBgndt = ev.NewDate;
            egd.EvntEndhdate = new DtTm(ev.NewDate.Ticks
+ (egd.EvntPrevEndhdate.Ticks - dialog.EvntPrevBgndt.Ticks));
        }
        DialogResult dhd = egd.ShowDialog();
        if (dhd == DialogResult.OK)
        {
            ev.Evnt.BeginDtTm = egd.EvntBgndt;
            ev.Evnt.EndhdtTm = egd.EvntEndhdate;

            wkTbl.RefreshEvnts();
            mnTbl.RefreshEvnts();
            dTbl.RefreshEvnts();

            wkTbl.Invalidate();
            mnTbl.Invalidate();
            dTbl.Invalidate();
        }
    }
}

```

```

        }
    }
}

private void EvntAdhdProcessor(object sender, EventArgs
ev)
{
    using (NewEvnt egd = new NewEvnt())
    {
        if (ev.Date != null)
        {
            egd.EvntBgndt = ev.Date.Value;
            egd.EvntEndhdate =
ev.Date.Value.AdhdMinutes(15);
        }
        DialogResult dhd = egd.ShowDialog();
        if (dhd == DialogResult.OK)
        {
            string name = egd.EvntTitle;
            DtTm db = egd.EvntBeginDtTm;
            DtTm de = egd.EvntDateEnd;
            e.Control.Evnts.Adhd(new ExtEvnt() { Name =
name, BeginDtTm = db, EndhdtTm = de });

            wkTbl.RefreshEvnts();
            mnTbl.RefreshEvnts();
            dTbl.RefreshEvnts();

            wkTbl.Invalidate();
            mnTbl.Invalidate();
            dTbl.Invalidate();
        }
    }
}

void MonCalDateSelected(object sender, EventArgs ev)
{
    wkTbl.Date = ev.BeginDtTm;
    mnTbl.Date = ev.BeginDtTm;
    dTbl.Date = ev.BeginDtTm;

    if (dTbl.Evnts != null)
    {
        EvntNumLabel.Text = dTbl.Evnts.Count;
    }
    ChosenDateLabel.Text =
ev.BeginDtTm.ToLongDateString();
}

```

```

        wkTbl.Invalidate();
        mnTbl.Invalidate();
        dTbl.Invalidate();
    }
}

private DtTm bgndt;
public DtTm Date
{
    get { return bgndt; }
    set
    {
        if (value.WkDy != WkDy.Mndg)
        {
            DtTm dt=value;
            while (dt.WkDy != WkDy.Mndg)
            {
                dt=dt.AdhdDays(-1);
            }
            bgndt = dt;
        }
        else
        {
            bgndt = value;
        }
        PsTmeP = false;
        PstnsValidEvnt = false;
    }
}

protected override void FindEvntPstns(Graphics g)
{
    if (Width == 0 || Mndg == null)
        return;

    foreach (DPrt d in Days)
    {
        d.Evnts.Clear();

        if (Evnts != null)
        {
            foreach (Evnt ev in Evnts)
            {

```

```

        if (ev.BeginDtTm.Year == d.Date.Year &&
ev.BeginDtTm.YearDay == d.Date.YearDay)
        {
            d.Evnts.Adhd(new EvntPart(ev));
        }
    }
}
PstnsValidEvnt = true;
}

protected override void FndTmPPstns(Graphics g)
{
    if (Width == 0 || Mndg==null)
        return;

    Mndg.Date = bgndt;
    Tsdg.Date = bgndt.AdhdDays(1);
    Wdnsdg.Date = bgndt.AdhdDays(2);
    Thrsg.Date = bgndt.AdhdDays(3);
    Frdh.Date = bgndt.AdhdDays(4);
    Strdg.Date = bgndt.AdhdDays(5);
    Sndg.Date = bgndt.AdhdDays(6);

    int stdhdWidth = ((this.Width-1) / 2);

    int stdhdHeight= (this.Height / 3);
    int shrtDHght = ((stdhdHeight- 1) / 2);
    int Yr1 = 0;
    int Yr2 = stdhdHeight;
    int Yr3 = Yr2+stdhdHeight;

    int XfrMon = 0;
    Mndg.Pstns = new Rectangle(XfrMon, Yr1, stdhdWidth,
stdhdHeight);

    int XfrTue = XfrMon + stdhdWidth;
    Tsdg.Pstns = new Rectangle(XfrTue, Yr1, stdhdWidth,
stdhdHeight);

    int XfrWen = 0;
    Wdnsdg.Pstns = new Rectangle(XfrWen, Yr2, stdhdWidth,
stdhdHeight);

    int XfrThr = XfrTue;
    Thrsg.Pstns = new Rectangle(XfrThr, Yr2, stdhdWidth,
stdhdHeight);

```

```

        int XfrFri = 0;
        Frdh.Pstns = new Rectangle(XfrFri, Yr3, stdhdWidth,
stdhdHeight);

        int XfrSat = XfrTue;
        int satDHeight = shortDHeight;
        Strdg.Pstns = new Rectangle(XfrSat, Yr3, stdhdWidth,
satDHeight);

        int XfrSun = XfrTue;
        int YfrSun = Yr3 + satDHeight;
        int snDHght = stdhdHeight- satDHeight;
        Sndg.Pstns = new Rectangle(XfrSun, YfrSun,
stdhdWidth, snDHght);

        LttHght = (int) (BasePlnr.MnLn.GetTextInfo(g,
Font).Height * 1.3);

        foreach (DPrt d in this.Days)
        {
            if (LttHght < Pstns.Height)
            {
                d.NamePstns = new Rectangle(d.Pstns.X,
d.Pstns.Y, d.Pstns.Width, LttHght);

                d.EvRectPstns = new Rectangle(d.Pstns.X,
d.Pstns.Y + d.NamePstns.Height, d.Pstns.Width, d.Pstns.Height -
d.NamePstns.Height);
            }
        }

        PsTmeP = true;
    }
    private int LttHght;

    protected override void OnPaint(EvntArgs ev)
    {
        base.OnPaint(ev);

        int evHght = BasePlnr.Evnt.GetTextInfo(e.Graphics,
Font).Height + 2;

        foreach (DPrt d in this.Days)
        {
            if (LttHght < Pstns.Height)
            {
                if (d == ChsnPs)

```

```

        {
BasePlnr.MnLnSelected.DrawBox(ev.Graphics, Font, d.NamePstns,
d.OutName);
        }
        else
        {
            BasePlnr.MnLn.DrawBox(ev.Graphics, Font,
d.NamePstns, d.OutName);
        }
        BasePlnr.Item.DrawBox(e.Graphics, Font,
d.EvRectPstns);

        int maxEvnts = 0;
        if (day.EvRectPstns.Height > evHght)
        {
            maxEvnts = day.EvRectPstns.Height/evHght;
        }

        int clNm = 1;
        if (maxEvnts > 0)
        {
            clNm = day.Evnts.Count / maxEvnts;
            if (day.Evnts.Count % maxEvnts >0)
            {
                clNm++;
            }
        }
        int currClNm=0;
        int clWdth = day.NamePstns.Width;
        if (clNm>0)
        clWdth = day.NamePstns.Width /clNm;

        int Yps= day.EvRectPstns.Y;
        int Xps = day.EvRectPstns.X;
        EvntPart slEvPrt=null;
        foreach (EvntPart ep in day.Evnts)
        {
            if (Yps+ evHght > d.EvRectPstns.Y +
d.EvRectPstns.Height)
            {
                currClNm++;
                Xps = Xps+clWdth;
                Yps= day.EvRectPstns.Y;
            }
            ep.Pstns = new Rectangle(Xps, Yps,
clWdth, evHght);

```


ДОДАТОК В
Слайди презентації

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Програмне забезпечення для календарного планування

Виконав: Микита КУЩЕНКО

ст. групи КНТ-131

Керівник: Тетяна ЗАЙКО

к.т.н., доцент

Рисунок В.1 – Слайд 1

ОБ'ЄКТ, ПРЕДМЕТ ТА МЕТА РОБОТИ

- Об'єкт дослідження – процес розробки програмного забезпечення для календарного планування.
- Предмет дослідження – програмні засоби для календарного планування.
- Метою роботи є розробка програмного забезпечення для календарного планування.

2

Рисунок В.2 – Слайд 2

ЗАВДАННЯ РОБОТИ

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для календарного планування;
- здійснити проєктування програмного забезпечення для календарного планування;
- створити програмне забезпечення для календарного планування;
- виконати тестування розробленого програмного забезпечення для календарного планування.

3

Рисунок В.3 – Слайд 3

ПОРІВНЯННЯ ІСНУЮЧИХ АНАЛОГІВ

Критерій порівняння	nTask	Calendly	Notion
Функції календаря та планування завдань	+	+	+-
Співпраця в команді	+	-	+
Організація зустрічей	+-	+	-
Інтеграція з іншими сервісами	+-	+	+
Простота інтерфейсу	+-	+	+-
Гнучкість налаштування	+-	-	+
Побудова робочих процесів	+	-	+-

4

Рисунок В.4 – Слайд 4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

При розробці програмного забезпечення для календарного планування необхідно забезпечити такі функціональні вимоги:

- підтримка можливості відображення інформації про події користувача у хронологічному порядку;
- підтримка можливості календарного планування шляхом додавання інформації про події користувача;
- можливість корегування інформації про події користувача, зокрема, інформації про початок та завершення події;
- можливість відображення переліку подій користувача на день, тиждень та місяць;
- підтримка можливості переміщення між списками подій за допомогою календаря.

5

Рисунок В.5 – Слайд 5

ПОРІВНЯННЯ МОВ ПРОГРАМУВАННЯ

Критерій порівняння мов програмування	Мова програмування		
	C#	Python	Ruby
Продуктивність виконання	+	+—	+
Розробка графічного інтерфейсу	+	+—	+—
Інтеграція з Windows-сервісами	+	+—	—
Зручність для створення ПЗ для календарного планування	+	—	—
Швидкість розробки	+	+	+—

6

Рисунок В.6 – Слайд 6

ПОРІВНЯННЯ СЕРЕДОВИЩ РОЗРОБКИ

Критерій порівняння середовищ розробки	Середовища розробки		
	Visual Studio	Eclipse	PyCharm
Зручність засобів налагодження	+	+	–
Інструменти для роботи з базами даних	+	+–	+–
Інтелектуальні підказки	+	+–	+–
Швидкість налаштування середовища	+	+–	+
Розширення функціоналу через плагіни	+	+–	+

Рисунок В.7 – Слайд 7

ЗАГАЛЬНА СХЕМА ПРОГРАМИ

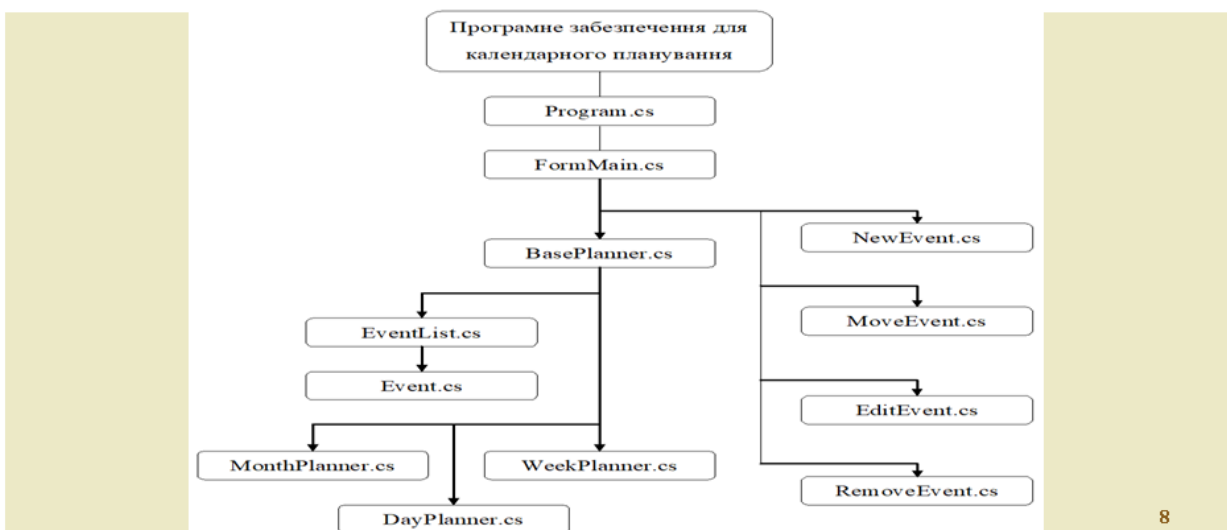
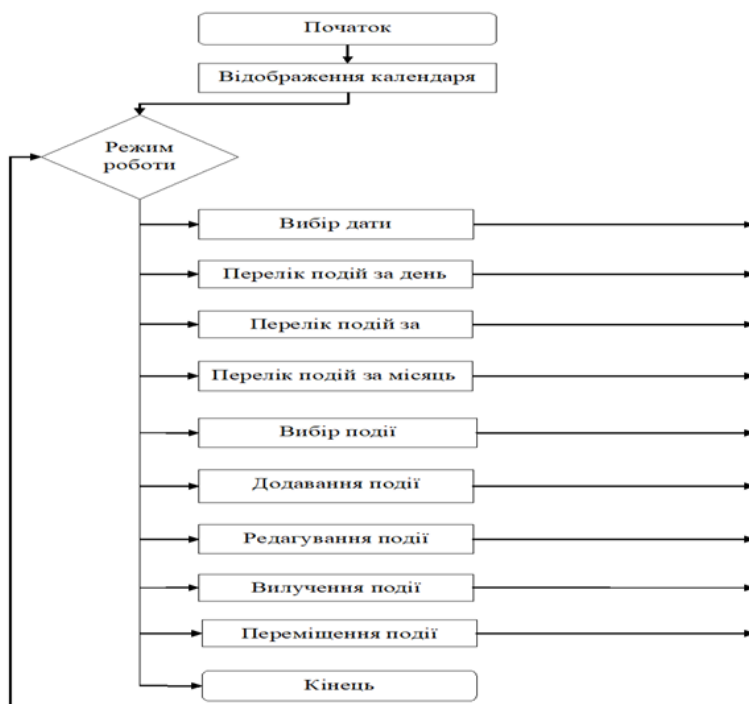


Рисунок В.8 – Слайд 8

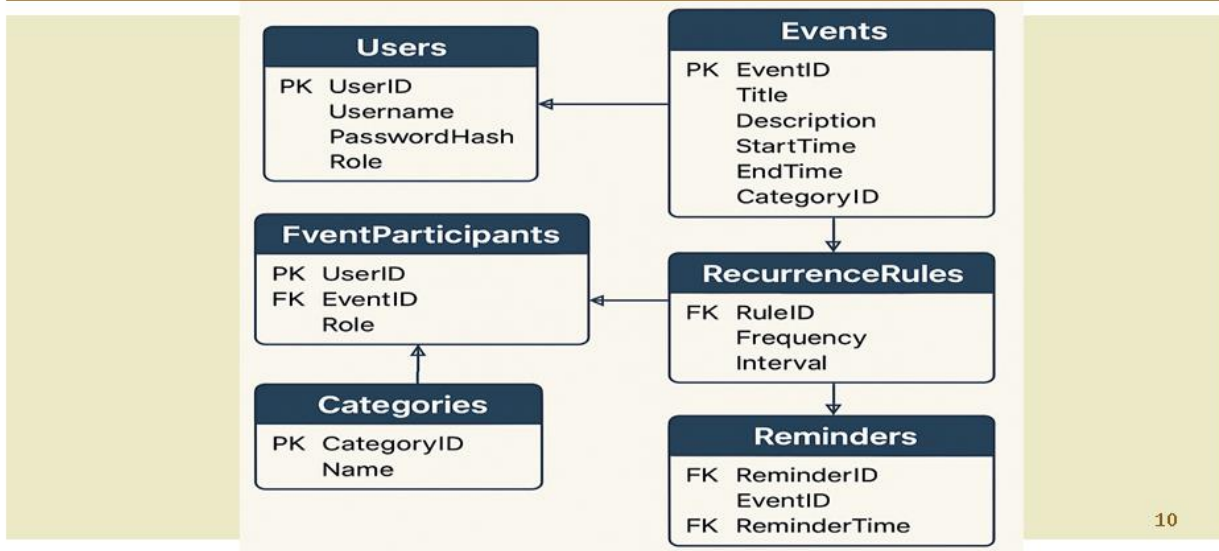


ФУНКЦІОНУВАННЯ ПРОГРАМИ

9

Рисунок В.9 – Слайд 9

БАЗА ДАНИХ



10

Рисунок В.10 – Слайд 10

ІНТЕРФЕЙС ЗАСТОСУНКУ. ГОЛОВНА ФОРМА

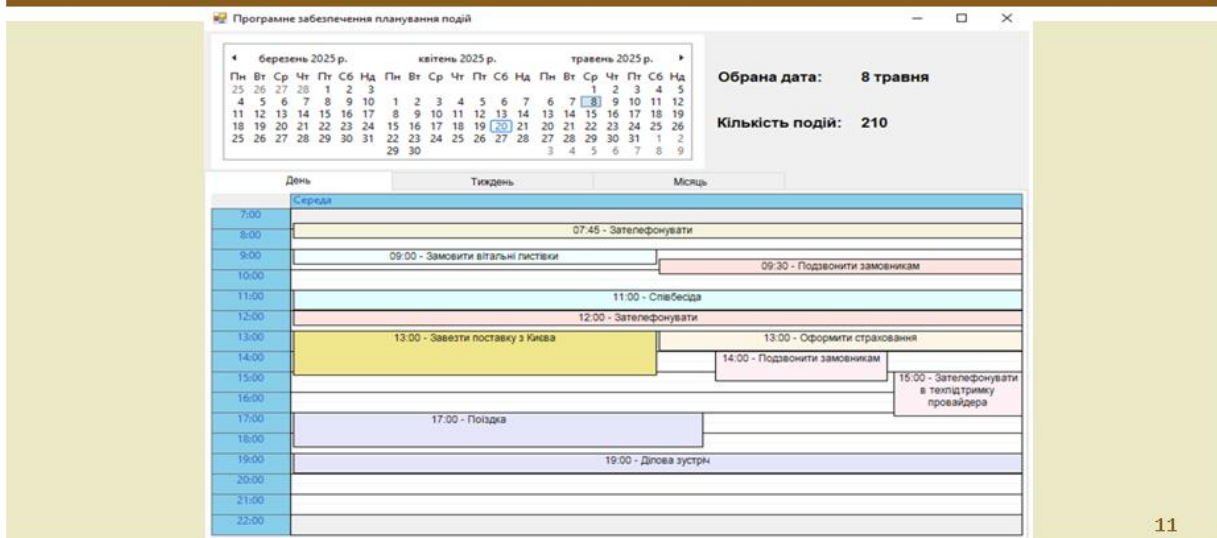


Рисунок В.11 – Слайд 11

ІНТЕРФЕЙС ЗАСТОСУНКУ. ПЛАНУВАННЯ ПОДІЙ

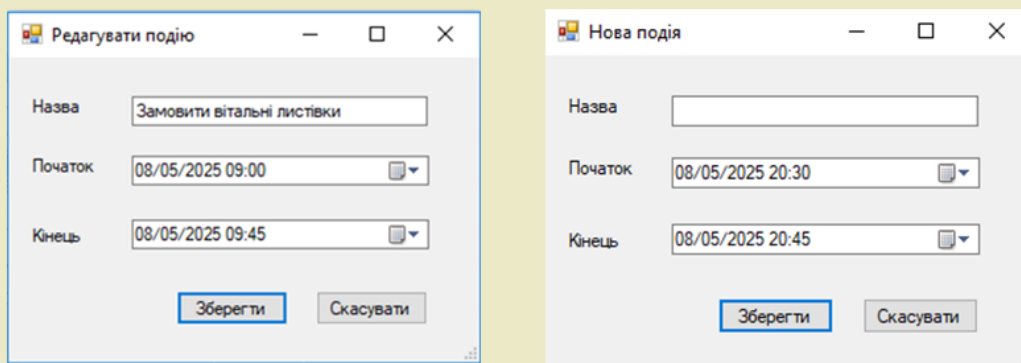


Рисунок В.12 – Слайд 12

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було реалізовано головну мету – розробка програмного забезпечення для календарного планування. Для цього було виконано ряд завдань:

- виконати аналіз предметної області та програмних засобів для календарного планування;
- здійснити проєктування програмного забезпечення для календарного планування;
- створити програмне забезпечення для календарного планування;
- виконати тестування розробленого програмного забезпечення для календарного планування.

Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою процесу календарного планування.

13

Рисунок В.13 – Слайд 13