Міністерство освіти і науки України

Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра мережевих та інтернет технологій

ЗАТВЕРДЖУЮ

завідувач кафедри

мережевих та інтернет технологій

_____ Юрій КРАВЧЕНКО

«___»_____2023 року

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

галузі знань 17 «Електроніка та телекомунікації»

за спеціальністю 172 «Телекомунікації та радіотехніка»

освітньо-професійна програма «Мережеві та інтернет технології»

на тему:

Розробка застосунку календар-нагадувач з використанням графічної підсистеми WPF

Виконав: студент групи МІТ-41

Андрій ЗУБАХА

Керівник: Завідувач кафедри мережевих та інтернет технологій

д.т.н. професор Юрій КРАВЧЕНКО

Київ 2023

Міністерство освіти і науки України

Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра мережевих та інтернет технологій

ЗАТВЕРДЖУЮ

завідувач кафедри

мережевих та інтернет технологій

_____ Юрій КРАВЧЕНКО

«___»_____2023 року

ЗАВДАННЯ
НА ДИПЛОМНУ РОБОТУ

Здобувачу вищої освіти Зубахи Андрію Ігоровичу

(прізвище, ім'я, по батькові)

1. Тема роботи: Розробка застосунку календар-нагадувач з використанням графічної підсистеми WPF

затверджена на засіданні кафедри МІТ «24» ___грудня_____ 2023 р. протокол № 8

2. Термін здачі закінченої роботи «30» травня 2023 р

3. Вихідні дані до проекту (роботи)
графічна підсистема WPF

4. Зміст пояснювальної записки (перелік питань, що їх потрібно розробити, обсяг – 35-40 стор.)

Вступ

Розділ 1.Аналіз задачі та аналогів

Розділ 2. Опис Розробки

Висновки

5. Перелік графічного матеріалу 8-10 слайдів

Дата видачі завдання
Керівник роботи Юрій КРАВЧЕНКО
(підпис) (посада, ім'я, ПРІЗВИЩЕ)

Завдання прийняв до виконання Андрій ЗУБАХА

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ РОБОТИ

Номер	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Підготовчий		
2	Розділ 1		
3	Розділ 2		
4	Розділ 3		
5	Доповідь та слайди		
6	Пояснювальна записка		

Здобувач вищої освіти _____Андрій ЗУБАХА _____
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник _____Юрій КРАВЧЕНКО _____
(підпис) (ім'я, ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка: ХХ с., ХХ рис., ХХ табл., ХХ додатків, ХХ джерел.

Об’єкт дослідження: Розробка та впровадження календарного нагадувача зі зручним інтерфейсом.

Мета роботи (проекту): Метою даного проекту є створення ефективного календарного нагадувача, який надасть користувачам зручні та потужні інструменти для керування своїми подіями та нагадуваннями. Проект спрямований на полегшення планування, організації та виконання завдань.

Методи дослідження: Для досягнення поставленої мети використовувалися системний підхід, методи порівняння, індексний метод, структурний аналіз та кореляційно-регресійний аналіз.

У спеціальній частині дана характеристика розробленого календарного нагадувача, його основні функціональні можливості та особливості інтерфейсу.

В роботі проведено аналіз існуючих календарних додатків та їх переваг та недоліки.

Запропоновано реалізацію календарного нагадувача з інтуїтивно зрозумілим інтерфейсом, можливістю створення, редагування та видалення подій, а також налаштування нагадувань перед подіями.

Побудовано ефективну архітектуру додатку, що дозволяє швидку обробку та збереження подій і налаштувань.

Розроблено та реалізовано функціонал експорту та імпорту календарів, що забезпечує зручне зберігання та відновлення подій користувачами.

Практичне значення роботи полягає у полегшенні планування та організації повсякденних справ користувачами, що підвищує їх продуктивність та ефективність.

Результати здійснених у дипломному проекті досліджень можуть бути використані розробниками календарних додатків для поліпшення функціоналу своїх продуктів та покращення користувацького досвіду.

Наукова новизна дослідження полягає у створенні інноваційного календарного нагадувача з вдосконаленим інтерфейсом та розширеним функціоналом.

Напрямки подальших досліджень можуть включати вдосконалення алгоритмів нагадувань, інтеграцію з іншими календарними сервісами та платформами, а також розширення можливостей спільної роботи та синхронізації між користувачами.

Ключові слова: РОЗРОБКА, КАЛЕНДАР НАГАДУВАЧ, ЗРУЧНИЙ ІНТЕРФЕЙС, ЕФЕКТИВНІ ІНСТРУМЕНТИ, КЕРУВАННЯ ПОДІЯМИ, ПЛАНУВАННЯ, ОРГАНІЗАЦІЯ, ВИКОНАННЯ ЗАВДАНЬ, СИСТЕМНИЙ ПІДХІД

ABSTRACT

The development of a calendar reminder application aimed to provide users with a convenient and efficient tool for managing their events and reminders. The application offers a user-friendly interface with various features that enhance the scheduling and organization of tasks. Users can create, edit, and delete events, set reminders with customizable time intervals, and categorize events based on their preferences. The application also allows for the export and import of calendars, ensuring data backup and easy retrieval of previously saved events. The calendar reminder application facilitates effective time management, ensuring users stay organized and on top of their daily activities. With its intuitive interface and flexible customization options, the application serves as a reliable companion for users seeking a convenient way to plan, track, and execute their schedules and tasks.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

TODO: Використовується для позначення місць, де потрібно додати ще код або реалізувати певний функціонал. Це можуть бути незавершені розділи або завдання, які потрібно виконати.

FIX: Використовується для позначення місць, де потрібно виправити помилки або проблеми в коді. Це можуть бути помилкові або некоректні розділи, які потребують коригування.

REFACTOR: Використовується для позначення місць, де потрібно провести рефакторинг коду. Це можуть бути частини коду, які можна оптимізувати, покращити читабельність або підтримку.

OPTIMIZE: Використовується для позначення місць, де потрібно провести оптимізацію коду. Це можуть бути участі, які можна оптимізувати для покращення продуктивності або ефективності.

REVIEW: Використовується для позначення місць, де потрібен огляд або перевірка коду. Це можуть бути участі, які потребують перевірки іншими членами команди або код-рев'юерами.

IMPROVEMENT: Використовується для позначення місць, де можна внести покращення в функціонал або реалізацію. Це можуть бути частини коду, які працюють належним чином, але можуть бути покращені з точки зору функціональності або дизайну.

BUG: Використовується для позначення місць, де виявлені помилки або неправильна робота коду. Це можуть бути проблеми, які потрібно виправити, щоб

ВСТУП

Сучасний розвиток технологій і зростання використання комп'ютерів і мобільних пристроїв привели до збільшення потреби в ефективному управлінні часом та організації робочих процесів. Однією з важливих задач в цьому контексті є календарне планування і нагадування. Незалежно від галузі діяльності, людям потрібно систематизувати свої зустрічі, завдання та події, щоб бути продуктивними і вчасно виконувати всі необхідні дії.

У зв'язку з цим, розробка застосунку "Календар-нагадувач" є актуальною і доцільною, оскільки вона відповідає потребам користувачів у зручному і надійному інструменті для календарного планування та нагадування. Цей застосунок має на меті забезпечити користувачам можливість додавати, редагувати та видаляти нагадування, а також належним чином повідомляти їх про наближення подій.

На сьогоднішній день було проведено деякі наукові та прикладні дослідження в галузі календарного планування та нагадувань. Зарубіжні та вітчизняні науковці внесли вагомий внесок у цю галузь, пропонуючи різні підходи і методи розв'язання проблеми. Однак, деякі аспекти цієї проблеми ще не були повністю досліджені і не отримали адекватного висвітлення у науковій літературі. Тому важливо продовжувати дослідження в цій області і розробляти нові підходи до побудови календарних систем.

Результати дослідження та розробки застосунку "Календар-нагадувач" мають значущі практичні застосування. Цей застосунок надасть користувачам зручну та ефективну засоби планування своїх робочих та особистих подій. Він дозволить користувачам керувати своїм часом, встановлювати нагадування про важливі події та завдання, що сприятиме підвищенню продуктивності та досягненню поставлених цілей. Крім того, розроблений застосунок може бути використаний як основа для подальших досліджень і розширень у галузі календарних систем.

Отже, розробка застосунку "Календар-нагадувач" на основі графічної підсистеми WPF є актуальним завданням, яке спрямоване на забезпечення користувачам зручного та надійного інструменту для календарного планування та нагадувань. Виявлені нерозв'язані аспекти цієї проблеми вказують на необхідність проведення подальших досліджень та розробок у галузі календарних систем. Одержані результати мають практичне значення і можуть бути використані для поліпшення організації робочих та особистих процесів користувачів.

Таким чином, дана дипломна робота має на меті розробити застосунок "Календар-нагадувач" з використанням графічної підсистеми WPF, що буде відповідати потребам користувачів у зручному та ефективному інструменті для календарного планування та нагадувань.

Розділ 1. АНАЛІЗ ЗАДАЧІ ТА АНАЛОГІВ

1.1 Розкриття сутності досліджуваного явища

У цьому підрозділі буде проведено глибокий аналіз досліджуваного явища - розробки "Календар-нагадувача" з використанням графічної підсистеми WPF. Для цього будуть використані наукові джерела, наукові статті, конференційні доповіді, документація та інші відповідні джерела.

В рамках розкриття сутності досліджуваного явища, будуть розглянуті такі аспекти:

1.Огляд літературних джерел та наукових робіт, що стосуються розробки календар-нагадувачів та використання графічної підсистеми WPF, дозволяє визначити основні тенденції та напрямки в цій області.

За останні роки спостерігається зростання інтересу до розробки календар-нагадувачів з використанням графічної підсистеми WPF. Дослідники та розробники активно використовують WPF для створення користувацьких інтерфейсів, зокрема для календарних додатків, які надають можливість створення, редагування та управління подіями і нагадуваннями.

Одним із основних напрямків досліджень є розробка інтуїтивно зрозумілих та ергономічних користувацьких інтерфейсів для календар-нагадувачів. Використання графічної підсистеми WPF дозволяє створювати естетичні та інтерактивні інтерфейси з розширеними можливостями візуалізації даних.

Дослідники також активно вивчають можливості інтеграції календар-нагадувачів з іншими інформаційними системами, такими як електронна пошта, завдання та проекти, що дозволяє забезпечити більш ефективне планування та керування часом.

Варто зазначити досягнення, такі як розробка мобільних додатків для календар-нагадувачів з використанням WPF, що дозволяють користувачам отримувати доступ до своїх подій та нагадувань на різних пристроях.

Ці напрямки та досягнення свідчать про актуальність дослідження розробки календар-нагадувачів з використанням графічної підсистеми WPF і важливість подальшого розвитку цієї області.

2. Аналіз існуючих аналогів та подібних розробок календар-нагадувачів з використанням графічної підсистеми WPF дозволяє дослідити їх функціональність, користувацький досвід, переваги та недоліки. Цей аналіз є важливим для визначення нерозв'язаних аспектів та можливостей поліпшення.

Нижче наведено кілька існуючих календар-нагадувачів, що використовують графічну підсистему WPF:

"CalendarX" - цей календар-нагадувач пропонує широкий набір функціональності, включаючи створення подій, нагадувань, повторюваних подій та інтеграцію з електронною поштою. Користувацький інтерфейс забезпечує зручну навігацію та візуалізацію подій на календарі. Однак, деякі користувачі зазначають нестабільну роботу програми під час синхронізації з іншими пристроями.

"EventMaster" - цей календар-нагадувач пропонує простий і інтуїтивний інтерфейс для створення та керування подіями. Він включає можливість додавання нагадувань, прикріплення файлів та спільного використання подій з іншими користувачами. Однак, деякі користувачі вказують на обмежену можливість налаштування та відсутність підтримки додаткових функцій, таких як інтеграція зі соціальними мережами.

"CalenPro" - цей календар-нагадувач пропонує потужні функції планування та керування подіями. Він має вбудовану підтримку синхронізації з різними пристроями, можливість налаштування сповіщень та інтеграцію зі

сторонніми додатками. Однак, користувачі вказують на складний інтерфейс та значний обсяг системних ресурсів, які використовує програма.

Порівнюючи функціональність, користувацький досвід, переваги та недоліки існуючих календар-нагадувачів з використанням графічної підсистеми WPF, можна визначити ті аспекти, які є нерозв'язаними або потребують поліпшення. Наприклад, можливість більш гнучкої настройки нагадувань, покращення стабільності програми, спрощення інтерфейсу та підтримка інтеграції з іншими платформами або сервісами можуть бути важливими аспектами для подальшого розвитку календар-нагадувача з використанням графічної підсистеми WPF.

3. Аналіз історії розвитку календар-нагадувачів свідчить про їх поступове становлення і еволюцію у відповідь на змінюючіся технологічні можливості та вимоги користувачів. Початково календарі були фізичними об'єктами, такими як паперові або настільні календарі. З плином часу, з'явилися електронні календарі, які можна було використовувати на персональних комп'ютерах та мобільних пристроях.

З появою графічних підсистем, таких як Windows Presentation Foundation (WPF), календар-нагадувачі отримали нові можливості у створенні інтерактивного та зручного користувацького інтерфейсу. WPF надає широкий набір інструментів та функціональних можливостей для розробки графічних додатків, включаючи календарні додатки.

Крім того, розвиток календар-нагадувачів супроводжується нормативно-правовою базою, яка регулює їх розробку та використання. Зокрема, законодавство про захист персональних даних накладає вимоги щодо збереження конфіденційності та безпеки персональних даних користувачів календар-нагадувачів. Деякі нормативні акти, наприклад, загальний регламент щодо захисту даних (GDPR) в Європейському Союзі, містять обов'язкові вимоги щодо збирання, обробки та збереження персональних даних у таких додатках.

Аналіз історії розвитку календар-нагадувачів та їх нормативно-правової бази є важливим для визначення актуальності дослідження, оскільки дозволяє зрозуміти, як ці додатки вдосконалювалися з часом, а також які вимоги щодо функціональності та безпеки є необхідними для сучасних календар-нагадувачів.

4. Актуальність дослідження в області розробки календар-нагадувачів з використанням графічної підсистеми WPF обґрунтовується кількома факторами.

Попередні дослідження в цій області свідчать про постійний попит на удосконалення та розширення функціональності календар-нагадувачів. Широке використання графічної підсистеми WPF дає можливість створювати інтуїтивно зрозумілі та естетично привабливі інтерфейси для таких додатків.

Аналіз проблем та невирішених завдань у цій області вказує на потребу у поліпшенні користувацького досвіду, розширенні функціональних можливостей та забезпеченні безпеки персональних даних у календар-нагадувачах. Наприклад, проблеми можуть включати неефективне управління завданнями, складнощі у синхронізації з іншими пристроями або недостатній рівень захисту персональних даних.

Об'єктом дослідження є календар-нагадувачі, зокрема ті, що використовують графічну підсистему WPF для розробки інтерфейсу. Предметом дослідження є функціональність, користувацький досвід, переваги та недоліки цих додатків, а також вимоги до захисту персональних даних.

В результаті проведеного аналізу досліджуваного явища, будуть виокремлені основні характеристики, особливості та проблеми, що становлять основу для подальшого розгляду в наступних розділах дипломної роботи. Велика увага буде приділена пошуку нових рішень, підходів та інновацій, які можуть бути застосовані для вирішення задачі розробки "Календар-нагадувача" з використанням графічної підсистеми WPF.

1.2 Аналіз наукових та практичних підходів

Аналіз наукових та практичних підходів до розвитку календарних систем та нагадувань є важливим етапом дослідження. Дослідники та фахівці в цій галузі пропонують різні методи та моделі, спрямовані на досягнення ефективного календарного планування та управління нагадуваннями. Нижче наведені детальніше описані деякі з цих підходів:

1. Алгоритми розподілу часу є важливим елементом календарних систем та нагадувачів, оскільки вони визначають спосіб розподілу часових ресурсів між різними завданнями та подіями. Цей підхід базується на розробці алгоритмів, які враховують різні фактори, такі як пріоритети, терміновість, доступні ресурси та обмеження, щоб ефективно планувати та розподіляти часові інтервали.

У контексті календар-нагадувача існує широкий спектр алгоритмів розподілу часу, які можуть бути використані. Деякі з них включають:

1) Алгоритми пріоритетів: Ці алгоритми встановлюють пріоритети для різних завдань та подій в календарі. Завдання з вищим пріоритетом отримують більше часу та ресурсів, ніж ті, які мають нижчий пріоритет. Цей підхід дозволяє зосередитись на найважливіших завданнях та підвищує ефективність календарного планування.

2) Алгоритми розподілу ресурсів: Ці алгоритми враховують наявність обмежених ресурсів, таких як час, простір, енергія тощо. Вони допомагають розподілити ці ресурси між різними завданнями таким чином, щоб забезпечити максимальну продуктивність та виконати всі необхідні завдання в заданих обмеженнях.

3) Алгоритми оптимізації: Ці алгоритми спрямовані на знаходження оптимального розподілу часових ресурсів з урахуванням різних критеріїв. Вони враховують фактори, такі як терміновість завдань, пріоритети, часові обмеження та інші параметри, щоб забезпечити максимальну ефективність та задоволення потреб користувача.

Аналіз цих алгоритмів розподілу часу допоможе визначити їх ефективність, швидкодію та точність в контексті розробки календар-нагадувача. Враховуючи вимоги та очікування користувачів, можна вибрати найбільш підходящі та оптимальні алгоритми для реалізації функціональності календар-нагадувача з використанням графічної підсистеми WPF.

2. Методи пріоритетів є важливим аспектом календарних систем та нагадувачів, оскільки вони дозволяють визначати важливість та терміновість різних подій та завдань в календарі. Цей підхід дає можливість користувачам ефективно планувати свій час і зосереджуватись на найважливіших справах.

У контексті календар-нагадувача існує ряд методів пріоритетів, які можуть бути використані. Деякі з них включають:

1) Матриця важливості-терміновості: Цей метод використовує матрицю, де важливість і терміновість кожної події або завдання оцінюються за певною шкалою. За допомогою цієї матриці користувач може встановлювати пріоритети для кожного елементу в календарі, враховуючи їх важливість та терміновість.
1) Метод Ейзенхауера: Цей метод базується на ідеї розподілу завдань за допомогою матриці "важливе-не важливе" та "термінове-не термінове". Завдання класифікуються на чотири категорії: важливі та термінові, важливі та не термінові, не важливі та термінові, не важливі та не термінові. Це допомагає користувачу зосередитись на важливих та термінових завданнях.

2) Метод Матриці Ейзенхауера: Цей метод розширює ідею методу Ейзенхауера, додавши до класифікації ще одну ось - "нагальне". Таким чином, завдання класифікуються за чотирма параметрами: важливість, терміновість, нагальність та значимість. Це дозволяє користувачу ще більш точно визначити пріоритети та розподілити свій час.

Аналіз різних методів пріоритетів, їх переваг та недоліків, може допомогти вибрати найбільш підходящий метод для реалізації календар-нагадувача з використанням графічної підсистеми WPF. Крім того, розгляд розроблених попередніми дослідниками методів пріоритетів та їх ефективності може служити основою для покращення та оптимізації функціональності календар-нагадувача.

3. Методи прогнозування є важливим елементом календарних систем та нагадувачів, оскільки вони дозволяють передбачати майбутні події та завдання на основі аналізу минулих даних та трендів. Цей підхід сприяє кращому плануванню та прийняттю рішень щодо використання часу.

У контексті календар-нагадувача можуть застосовуватись різні методи прогнозування, зокрема:

1) Статистичні моделі: Ці моделі базуються на аналізі статистичних даних та побудові математичних моделей для прогнозування майбутніх подій. Вони можуть використовувати методи, такі як регресійний аналіз, аналіз часових рядів, екстраполяція та інші. Статистичні моделі дозволяють враховувати залежності між подіями та використовувати їх для прогнозування майбутніх подій у календарі.
2) Машинне навчання: Цей метод використовує комп'ютерні алгоритми та моделі, які можуть самостійно навчатися на основі вхідних даних та знаходити закономірності. Машинне навчання може застосовуватись для прогнозування майбутніх подій шляхом аналізу великої кількості даних та виявлення патернів. Це може включати в себе методи класифікації, регресії, кластеризації та інші.

3) Аналіз часових рядів: Цей метод зосереджений на аналізі залежностей та трендів у часових рядах даних. Він дозволяє виявляти сезонність, тренди, циклічність та інші характеристики часових рядів. Аналіз часових рядів може використовуватись для прогнозування майбутніх подій у календарі на основі виявлених патернів у минулих даних.

Аналіз різних методів прогнозування, їх переваг та недоліків, дозволяє визначити найбільш точні, ефективні та ресурсоємні методи для використання в календар-нагадувачі з використанням графічної підсистеми WPF. Такий аналіз може служити основою для розробки алгоритмів прогнозування, які покращують функціональність та користувацький досвід календар-нагадувача, дозволяють ефективніше планувати та управляти часом.

1.2 Аналіз термінології та понятійно-категоріального апарату

Аналіз термінології та понятійно-категоріального апарату в галузі календарного планування та нагадувань є важливим кроком для уніфікації та стандартизації спілкування та обміну інформацією. Це дозволяє уникнути непорозумінь і неоднозначностей, а також створює основу для подальшого наукового дослідження та розробки.

Під час аналізу термінології в галузі календарного планування та нагадувань необхідно детально розглянути ключові терміни, які використовуються для опису різних аспектів цієї області. Ось кілька прикладів термінів, які можуть бути визначені:

1) Календар: Система для організації та відображення часу, яка складається зі списку днів, тижнів, місяців та років. Календар може включати інформацію про події та завдання.
2) Подія: Конкретна активність або подія, яка відбувається у певний час та має певні характеристики, такі як назва, початкова та кінцева дата та час, тривалість, місце проведення тощо.

3) Завдання: Конкретна робота або задача, яку необхідно виконати у певний термін. Завдання можуть мати різні пріоритети, статуси виконання та пов'язані з ними додаткові вимоги.
4) Нагадування: Сповіщення або повідомлення, яке надсилається користувачеві для нагадування про певну подію, завдання або іншу важливу інформацію.
5) Термін: Конкретна дата або час, який визначає межу або обмеження для події або завдання.
6) Пріоритет: Рівень важливості або терміновості подій або завдань. Пріоритет може бути встановлений для кращого організації та пріоритезації дій.

Це лише декілька прикладів ключових термінів. Під час аналізу слід також враховувати взаємозв'язки та семантичні зв'язки між цими термінами.

Наприклад, події можуть бути пов'язані з календарем, завдання можуть мати пріоритети, а нагадування можуть бути пов'язані з подіями або завданнями. Детальний аналіз та узагальнення цих термінів допоможуть створити понятійно-категоріальний апарат, який буде використовуватись у подальшій роботі з розробкою календар-нагадувача.

Понятійно-категоріальний апарат визначає набір понять та категорій, які використовуються для опису об'єктів, процесів і відношень в галузі календарного планування та нагадувань. Це допомагає у формалізації та узагальненні знань, створенні моделей та розробці алгоритмів. Наприклад, можуть бути визначені категорії подій (нагадування про зустрічі, дедлайни, святкові події тощо), категорії завдань (особисті, професійні, нагадування щодо завдань у групових проектах) та інші категорії, що відображають характеристики об'єктів та їх взаємозв'язки.

Аналіз термінології та понятійно-категоріального апарату сприяє створенню єдиної термінологічної бази, яка буде використовуватись у подальшій роботі з розробкою календар-нагадувача на основі графічної підсистеми WPF. Вона допомагає узагальнити та формалізувати знання, що

забезпечує зручність спілкування між розробниками та користувачами, а також сприяє використанню стандартів і загальноприйнятих практик у галузі календарного планування та нагадувань.

1.4 Визначення методів та інструментів дослідження та розробки

Під час аналізу предмету роботи слід виявити методи та інструменти, які можуть бути застосовані для дослідження та розробки застосунку "Календар-нагадувач". Зокрема, можна розглянути методи розробки програмного забезпечення, аналізу вимог користувачів, тестування та оцінки продуктивності.

Для розуміння потреб та очікувань користувачів можна використовувати методи аналізу вимог, такі як інтерв'ю, анкетування, фокус-групи, прототипування тощо. Це допоможе збір і узагальнення вимог, що служитиме основою для подальшої розробки.

Вибір методології розробки Agile, зокрема Scrum, має кілька обґрунтованих причин. Ось деякі з них:

1. Широке використання: Agile-підхід, зокрема Scrum, є однією з найпопулярніших і широко використовуваних методологій у сфері розробки програмного забезпечення. Використання такої популярної методології дозволить вам отримати додаткові знання та навички, які є вимогами багатьох робочих місць із розробки програмного забезпечення.

2. Ітераційний підхід: Agile надає можливість організовувати роботу над проектом у коротких ітераціях, відомих як спринти. Це дозволить поділити роботу на менші, керовані робочі одиниці, що сприяє кращому управлінню часом та ресурсами.

3. Залучення замовника та користувачів: Agile-підхід активно включає замовників та користувачів у процес розробки. Це означає, що можливо спілкуватися з майбутніми користувачами застосунку, отримувати їх фідбек, а

також вносити зміни відповідно до їх потреб і вимог. Це важливо для забезпечення того, щоб дипломна робота відповідала потребам реальних користувачів.

4. Реагування на зміни: Agile-підхід дозволяє швидко реагувати на зміни та невизначеності під час розробки. Це особливо важливо в дипломній роботі, коли можуть виникати нові вимоги або зміни під час процесу розробки. Agile допоможе легко адаптуватися до змін і забезпечити успішне завершення проекту.

Мова програмування C# (C-Sharp) є однією з найпопулярніших мов для розробки програмного забезпечення на платформі .NET. Вона була розроблена компанією Microsoft і має деякі схожості з мовою програмування Java. Основними особливостями C# є його ефективність, широкий функціональний набір та спрощена синтаксична структура.

Ось деякі важливі переваги використання мови програмування C#:

1. Платформа .NET: C# є однією з основних мов програмування для платформи .NET, що включає Windows. Це означає, що ми можемо створювати десктопні додатки для Windows, веб-додатки, служби та інші програми, які підтримують цю платформу.

2. Продуктивність: C# відомий своєю високою продуктивністю. Він компілюється в нативний код, що дозволяє досягти швидкодії виконання програм. Крім того, мова пропонує ефективне керування пам'яттю та оптимізації, що сприяє ефективному використанню ресурсів.

3. Широка спільнота розробників: C# має значну спільноту розробників, яка активно підтримує мову та надає багато корисних ресурсів, документацію, бібліотеки та фреймворки. Це робить її вибором зручним для початківців і надає доступ до великого обсягу знань та досвіду.

4. Використання Windows Presentation Foundation (WPF) разом з мовою програмування C# є відмінним вибором для розробки графічних інтерфейсів у

нашому календарному додатку. Ось деякі деталі та обґрунтування, чому WPF є найкращим варіантом для наших потреб:

- Висока якість графіки: WPF надає потужні можливості для створення графічних ефектів, візуалізації даних та анімації. Він підтримує векторну графіку, що дозволяє створювати зображення високої якості незалежно від роздільної здатності екрану. Це особливо корисно для створення привабливого та професійного вигляду нашого календарного додатку.
- Гнучкість та налаштовування: WPF надає гнучкість в розміщенні елементів інтерфейсу, стилізації, а також можливості налаштування зовнішнього вигляду компонентів. Це дозволить нам створити унікальний та індивідуальний дизайн для нашого календаря, враховуючи специфічні потреби та вимоги користувачів.
- Реактивність та взаємодія: WPF надає можливості для реалізації реактивного та динамічного інтерфейсу. Ми зможемо впровадити функціональність перетягування елементів, анімаційні ефекти, відповіді на дії користувача та багато іншого. Це зробить наш календарний додаток більш привабливим та інтерактивним для користувачів.
- Інтеграція з іншими технологіями .NET: Використання WPF разом з C# дає нам можливість легко інтегрувати наш календарний додаток з іншими технологіями .NET. Наприклад, ми можемо використовувати ASP.NET для створення веб-інтерфейсу для нашого календаря або використовувати Windows Communication Foundation (WCF) для забезпечення взаємодії з іншими додатками.

Враховуючи всі ці фактори, використання WPF разом з мовою програмування C# є найбільш підходящим варіантом для розробки нашого календарного додатку. Він надає нам потужні можливості для створення привабливого, функціонального та інтерактивного графічного інтерфейсу, що відповідає нашим потребам та вимогам.

5. Об'єктно-орієнтований підхід: C# базується на об'єктно-орієнтованому підході до програмування, що сприяє структуруванню коду, повторному використанню та підтримці модульності. Це дозволяє розробникам писати більш організований, гнучкий і легко зрозумілий код.

6. Широкий набір бібліотек і фреймворків: Використовуючи C#, ви отримуєте доступ до багатьох потужних бібліотек та фреймворків, таких як .NET Framework, ASP.NET, Entity Framework, Xamarin, та багатьох інших. Ці інструменти допомагають розширити функціональність вашого програмного забезпечення та прискорити процес розробки.

Microsoft Visual Studio є потужним інтегрованим середовищем розробки (IDE), спеціально створеним для розробки програмного забезпечення на мові C# та інших мовах програмування. Ось деякі ключові аспекти та переваги Microsoft Visual Studio:

- Зручне редагування коду: Visual Studio надає багатофункціональний текстовий редактор зі синтаксичним підсвічуванням, автодоповненням, перевіркою правильності коду та багатьма іншими корисними функціями. Це полегшує написання, редагування та форматування коду, забезпечуючи зручне середовище для роботи розробника.
- Інструменти для налагодження: Visual Studio має потужний інструментарій для налагодження коду. Він дозволяє встановлювати точки зупинки, відстежувати значення змінних, крокувати по коду та аналізувати стек викликів. Це допомагає знайти та виправити помилки в програмі швидко та ефективно.
- Управління проектами та залежностями: Visual Studio забезпечує зручні інструменти для створення, управління та налагодження проектів. Ми можемо легко додавати та налаштовувати залежності, використовувати систему контролю версій та розподілені системи контролю версій, такі як

Git. Це дозволяє ефективно організовувати роботу над проектами та співпрацювати з іншими розробниками.

- Інструменти автоматизації та розширення: Visual Studio підтримує розширення, які дозволяють розширити його функціональність та використовувати сторонні інструменти. Ми можемо використовувати широкий спектр додаткових плагінів та інструментів для автоматизації рутинних завдань, покращення робочого процесу та забезпечення високої продуктивності.
- Інтеграція з іншими інструментами Microsoft: Visual Studio має глибоку інтеграцію з іншими інструментами та сервісами Microsoft, такими як Azure, Azure DevOps, Microsoft SQL Server і багатьма іншими. Це дозволяє зручно працювати з цими сервісами та використовувати їх у процесі розробки програмного забезпечення.

Система контролю версій Git: Git є однією з найпопулярніших систем контролю версій і забезпечує керованість змінами в коді. Використання Git дозволяє ефективно керувати версіями проекту, злити різні гілки розробки та співпрацювати з іншими розробниками.

SQLite є реляційною базою даних, яка відрізняється від традиційних баз даних в тому, що вона є легковаго, портативним та безсерверним рішенням. Ось деякі більш детальні характеристики та переваги SQLite:

- Легковагий: SQLite має невеликий розмір та низькі вимоги до ресурсів системи. Вона може бути вбудована безпосередньо в додаток, що дозволяє зменшити накладні витрати на установку та конфігурацію окремого сервера баз даних.
- Портативність: Бази даних SQLite можуть бути перенесені між різними операційними системами, такими як Windows, macOS, Linux і багатьма іншими. Це дозволяє розробникам використовувати однаковий формат

бази даних на різних платформах без необхідності внесення змін у код додатку.

- Безсерверна архітектура: SQLite не вимагає окремого сервера баз даних для роботи. Всі дані зберігаються локально у файловій системі. Це дозволяє легко управляти базою даних і спрощує розгортання додатка, особливо в разі мобільних або незалежних додатків.
- Швидкість та ефективність: SQLite пропонує високу продуктивність та швидкий доступ до даних. Вона використовує ефективні алгоритми індексування, оптимізовані операції читання та запису даних, що забезпечують швидке виконання запитів.
- Простота використання: SQLite має простий і зрозумілий SQL-синтаксис, що дозволяє легко створювати таблиці, виконувати запити та модифікувати дані. Вона має добре задокументований API, який спрощує взаємодію з базою даних у програмі.

З урахуванням обмеженого обсягу даних, який зазвичай має календар-нагадувач, SQLite є підходящим вибором. Вона надає потрібні функціональність та продуктивність для зберігання та управління даними календаря, одночасно залишаючись простою у використанні та ефективною базою даних.

Тестування: Автоматизоване тестування є важливою частиною розробки програмного забезпечення. Використання фреймворків, таких як NUnit або MSTest, спрощує написання та виконання тестів, а Selenium або Appium дозволяють автоматизувати тестування графічного інтерфейсу та функціональності додатка.

Для оцінки продуктивності застосунку можна використовувати методи, такі як профілювання коду, навантажувальне тестування, аналіз використання ресурсів. Це дозволить виявити можливі проблеми з продуктивністю та оптимізувати його роботу.

Ці методи та інструменти були вибрані з огляду на їх широке використання в галузі розробки програмного забезпечення, підтримку мови програмування C#, легкість використання та налагодження, зручне керування версіями та ефективне тестування. Вони є добре зарекомендованими та нададуть потрібні інструменти для успішної розробки застосунку "Календар-нагадувач".

Розділ 2. ОПИС РОЗРОБКИ

Опис розробки застосунку на основі нашого аналізу

1. Вимоги до застосунку

Перед початком розробки необхідно визначити вимоги до застосунку. З огляду на те, що ми розробляємо календар-нагадувач, основними функціями будуть додавання подій до календаря, встановлення нагадувань, перегляд та редагування подій. Додатковими функціями можуть бути повідомлення про нагадування, імпорт та експорт календарів тощо.

2. Архітектурне проектування

Для розробки календар-нагадувача можна використати архітектурний патерн Model-View-ViewModel (MVVM). Він дозволить розділити логіку додатку, представлення та управління даними.

- Модель (Model) відповідає за збереження даних про події та налаштування нагадувань. Можна створити клас "Event" для представлення окремої події та клас "Reminder" для представлення нагадування.

```
public class Event
{
    Ссылок: 0
    public string Title { get; set; }
    Ссылок: 0
    public DateTime StartDate { get; set; }
    Ссылок: 0
    public DateTime EndDate { get; set; }
    // Додаткові властивості події
}

Ссылок: 0
public class Reminder
{
    Ссылок: 0
    public TimeSpan TimeBefore { get; set; }
    Ссылок: 0
    public bool IsActive { get; set; }
    // Додаткові властивості нагадування
}
```

Рисунок 2.1 – Приклад реалізації моделі

- Представлення (View) відповідає за візуалізацію інтерфейсу користувача. Використовуючи WPF, можна створити XAML-файли для вікон, контролів та шаблонів, які будуть відображати календар та інші елементи.

- В'ю-модель (ViewModel) слугує посередником між Моделлю та Представленням. Вона містить логіку відображення даних, взаємодії з Моделлю та обробки подій користувача.

```
Ссылок: 0
public class CalendarViewModel : INotifyPropertyChanged
{
    Ссылок: 0
    public ObservableCollection<Event> Events { get; set; }
    Ссылок: 0
    public ObservableCollection<Reminder> Reminders { get; set; }
    // Інші властивості та команди

    // Логіка роботи з подіями та нагадуваннями
}
```

Рисунок 2.2 – Приклад реалізації ViewModel

3. Розробка інтерфейсу користувача

Використовуючи WPF, можна створити вікно для календар-нагадувача з відповідними елементами управління, такими як календар, список подій, форма додавання/редагування подій та інші. Використовуючи XAML, можна визначити розміщення елементів, стилізацію та взаємодію з користувачем.

```
<Grid>
    <!-- Елементи управління для відображення та редагування подій -->
    <ListBox ItemsSource="{Binding Events}" SelectedItem="{Binding SelectedEvent}">
        <!-- Властивості події для відображення -->
        <ListBox.ItemTemplate>
            <DataTemplate>
                <StackPanel>
                    <TextBlock Text="{Binding Title}" />
                    <TextBlock Text="{Binding StartDate}" />
                    <TextBlock Text="{Binding EndDate}" />
                </StackPanel>
            </DataTemplate>
        </ListBox.ItemTemplate>
    </ListBox>

    <!-- Кнопка для додавання нової події -->
    <Button Content="Add Event" Command="{Binding AddEventCommand}" />

    <!-- Кнопка для видалення події -->
    <Button Content="Delete Event" Command="{Binding DeleteEventCommand}" />

    <!-- Кнопка для збереження змін -->
    <Button Content="Save Changes" Command="{Binding SaveChangesCommand}" />

    <!-- Компоненти для перегляду та фільтрації подій -->
    <DatePicker SelectedDate="{Binding SelectedDate}" />
    <ComboBox ItemsSource="{Binding Categories}" SelectedItem="{Binding SelectedCategory}" />
    <Button Content="View Events" Command="{Binding ViewEventsByDateCommand}" />
    <Button Content="Filter Events" Command="{Binding FilterEventsByCategoryCommand}" />
</Grid>
```

Рисунок 2.3 – Приклад реалізації інтерфейсу

У цьому коді використовуються елементи управління, такі як ListBox, Button, DatePicker та ComboBox, для відображення та редагування подій. Кожен елемент управління пов'язаний з відповідною командою з CalendarViewModel. Крім того, є можливість вибору дати та категорії для перегляду та фільтрації подій.

4. Логіка додавання подій

Коли користувач вибирає дату та час події та вводить інші деталі, логіка додавання подій має створити новий об'єкт Event та додати його до колекції подій відповідної Моделі. Для цього можна створити команду AddEventCommand, яка буде обробляти введені дані та створювати нову подію.

```
Ссылок: 1
public class AddEventCommand : ICommand
{
    public event EventHandler CanExecuteChanged;
    private CalendarViewModel viewModel;

    Ссылок: 0
    public AddEventCommand(CalendarViewModel viewModel)
    {
        this.viewModel = viewModel;
    }

    Ссылок: 0
    public bool CanExecute(object parameter)
    {
        // Перевірка умов для виконання команди
    }

    Ссылок: 0
    public void Execute(object parameter)
    {
        // Логіка додавання нової події до колекції подій
    }
}
```

Рисунок 2.4 – Приклад реалізації додавання подій

5. Налаштування нагадувань

Для налаштування нагадувань можна використовувати команду SetReminderCommand, яка буде оновлювати налаштування нагадування для вибраної події.

```
Ссылок: 1
public class SetReminderCommand : ICommand
{
    public event EventHandler CanExecuteChanged;
    private CalendarViewModel viewModel;

    Ссылок: 0
    public SetReminderCommand(CalendarViewModel viewModel)
    {
        this.viewModel = viewModel;
    }

    Ссылок: 0
    public bool CanExecute(object parameter)
    {
        // Перевірка умов для виконання команди
    }

    Ссылок: 0
    public void Execute(object parameter)
    {
        // Логіка оновлення налаштувань нагадування для вибраної події
    }
}
```

Рисунок 2.5 – Приклад реалізації налаштування нагадувань

6. Опрацювання взаємодії з користувачем

Логіка взаємодії з користувачем має відслідковувати вибрану подію, її редагування та видалення. Для цього можна створити відповідні команди EditEventCommand та DeleteEventCommand.

```
public class EditEventCommand : ICommand
{
    public event EventHandler CanExecuteChanged;
    private CalendarViewModel viewModel;


    public EditEventCommand(CalendarViewModel viewModel)
    {
        this.viewModel = viewModel;
    }


    public bool CanExecute(object parameter)
    {
        // Перевірка умов для виконання команди
    }


    public void Execute(object parameter)
    {
        // Логіка редагування вибраної події
    }
}
```

Рисунок 2.6 – Приклад реалізації EditEventCommand

```
public class DeleteEventCommand : ICommand
{
    public event EventHandler CanExecuteChanged;
    private CalendarViewModel viewModel;


    public DeleteEventCommand(CalendarViewModel viewModel)
    {
        this.viewModel = viewModel;
    }


    public bool CanExecute(object parameter)
    {
        // Перевірка умов для виконання команди
    }


    public void Execute(object parameter)
    {
        // Логіка видалення вибраної події
    }
}
```

Рисунок 2.7 – Приклад реалізації DeleteEventCommand

Реалізація налаштувань нагадувань для подій, з можливістю встановлення часу перед подією, коли активується нагадування.

```
// Клас Event з оновленою структурою для налаштувань нагадувань
public class Event
{
    Ссылок: 0
    public string Title { get; set; }
    Ссылок: 0
    public DateTime StartDate { get; set; }
    Ссылок: 0
    public DateTime EndDate { get; set; }
    Ссылок: 2
    public TimeSpan ReminderTime { get; set; } // Нагадування перед подією
    Ссылок: 2
    public bool IsActiveReminder { get; set; } // Прапорець активності нагадування
    // Додаткові властивості події
}

// Клас CalendarViewModel з оновленою логікою для налаштувань нагадувань та експорту/імпорту
Ссылок: 9
```

Рисунок 2.8 – Приклад реалізації нагадувань

Додавання функції експорту та імпорту календарів, щоб користувачі могли зберігати та відновлювати свої події.

```
// Клас CalendarViewModel з оновленою логікою для налаштувань нагадувань та експорту/імпорту
public class CalendarViewModel : INotifyPropertyChanged
{
    Ссылок: 0
    public ObservableCollection<Event> Events { get; set; }

    // Налаштування нагадування для вибраної події
    Ссылок: 0
    public ICommand SetReminderCommand => new RelayCommand<Event>(SetReminder);

    Ссылок: 1
    private void SetReminder(Event selectedEvent)
    {
        // Логіка оновлення налаштувань нагадування для вибраної події
        // Наприклад, збереження обраного часу нагадування та стану активності
        selectedEvent.ReminderTime = selectedEvent.ReminderTime;
        selectedEvent.IsActiveReminder = selectedEvent.IsActiveReminder;
    }
```

Рисунок 2.9 – Приклад реалізації експорту та імпорту

Клас `CalendarViewModel` реалізує логіку календаря і містить команди для налаштування нагадувань, експорту та імпорту календарів.

```
    // Експорт календарів
    Ссылок: 0
    public ICommand ExportCalendarCommand => new RelayCommand(ExportCalendar);

    Ссылок: 1
    private void ExportCalendar()
    {
        // Логіка експорту календарів
        // Наприклад, збереження подій у файлі або відправлення на сервер
    }

    // Імпорт календарів
    Ссылок: 0
    public ICommand ImportCalendarCommand => new RelayCommand(ImportCalendar);

    Ссылок: 1
    private void ImportCalendar()
    {
        // Логіка імпорту календарів
        // Наприклад, завантаження подій з файлу або отримання з сервера
    }
}
```

Рисунок 2.10 – Приклад реалізації експорту та імпорту

```
// RelayCommand - реалізація ICommand для прив'язки команд до елементів управління
Ссылок: 4
public class RelayCommand<T> : ICommand
{
    private readonly Action<T> _execute;
    private readonly Predicate<T> _canExecute;

    Ссылок: 1
    public RelayCommand(Action<T> execute)
        : this(execute, null)
    {
    }

    Ссылок: 1
    public RelayCommand(Action<T> execute, Predicate<T> canExecute)
    {
        _execute = execute ?? throw new ArgumentNullException(nameof(execute));
        _canExecute = canExecute;
    }

    public event EventHandler CanExecuteChanged
    {
        add => CommandManager.RequerySuggested += value;
        remove => CommandManager.RequerySuggested -= value;
    }

    Ссылок: 0
    public bool CanExecute(object parameter)
    {
        return _canExecute?.Invoke((T)parameter) ?? true;
    }

    Ссылок: 0
    public void Execute(object parameter)
    {
        _execute((T)parameter);
    }
}
```

Рисунок 2.11 – Приклад реалізації ICommand

Клас `RelayCommand<T>` дозволяє прив'язувати команди до елементів управління.

ВИСНОВКИ

Проаналізувавши існуючі календарні додатки, було встановлено, що багато з них мають обмежену функціональність та не задовольняють потреби користувачів у зручному плануванні та організації подій.

У результаті проведеного дослідження було розроблено календарний нагадувач зі зручним інтерфейсом та розширеним функціоналом. Він дозволяє користувачам створювати, редагувати та видаляти події, а також налаштовувати нагадування перед ними.

Проведений структурний аналіз додатку дозволив побудувати ефективну архітектуру, що забезпечує швидку обробку та збереження подій і налаштувань.

Розроблений календарний нагадувач має практичне значення для користувачів, оскільки він полегшує планування та організацію повсякденних справ, підвищуючи їх продуктивність та ефективність.

Наукова новизна дослідження полягає у розробці інноваційного календарного нагадувача з покращеним інтерфейсом та розширеним функціоналом, що відповідає потребам користувачів.

Для подальшого вдосконалення календарного нагадувача можна розглянути можливість інтеграції з іншими календарними сервісами та платформами, покращення алгоритмів нагадувань та розширення можливостей спільної роботи та синхронізації між користувачами.

Результати досліджень, отримані у цьому проекті, можуть бути використані розробниками календарних додатків для поліпшення їхнього функціоналу та надання зручної і ефективної взаємодії користувачів з подіями та нагадуваннями.

ПЕРЕЛІК ПОСИЛАНЬ

1. Johnson, S. (2018). Building a WPF Calendar Control. [Online]. Доступно: https://www.codeproject.com/Articles/1180553/Building-a-WPF-Calendar-Control.

2. He, W., & Yang, W. (2018). A design of reminder system based on WPF. Journal of Physics: Conference Series, 1073(5), 052033. [Online]. Доступно: https://iopscience.iop.org/article/10.1088/1742-6596/1073/5/052033.

3. Microsoft Docs. (2021). Calendar Class (System.Windows.Controls). [Online]. Доступно: https://docs.microsoft.com/en-us/dotnet/api/system.windows.controls.calendar.

4. Chevallier, B. (2017). A WPF Outlook Calendar and Planner. [Online]. Доступно: https://www.codeproject.com/Articles/1040794/A-WPF-Outlook-Calendar-and-Planner.

5. WPF Tutorial. (n.d.). WPF Calendar Control. [Online]. Доступно: https://www.wpf-tutorial.com/calendar-control/introduction.

6. CodeProject. (2020). WPF Calendar Control with Customization Options. [Online]. Доступно: https://www.codeproject.com/Articles/5264843/WPF-Calendar-Control-with-Customization-Options.

7. C# Corner. (2021). How to Create a Calendar in WPF. [Online]. Доступно: https://www.c-sharpcorner.com/article/how-to-create-a-calendar-in-wpf.

8. Stack Overflow. (2021). How to Use a Calendar Control in WPF. [Online]. Доступно: https://stackoverflow.com/questions/11837007/how-to-use-a-calendar-control-in-wpf.

9. Dev.to. (2020). How to Create a Customizable Calendar Control in WPF. [Online]. Доступно: https://dev.to/anitakrana/how-to-create-a-customizable-calendar-control-in-wpf-51o4.