Вінницький національний технічний університет
Факультет інформаційних технологій та комп’ютерної інженерії
Кафедра комп’ютерних наук

Пояснювальна записка

до магістерської кваліфікаційної роботи

на тему «Інформаційна технологія продуктивної організації часу»

Виконав: студент 2 курсу,
групи 1КН-18 м
спеціальності 122 «Комп'ютерні науки»
Волянська Є.В.
Керівник: к.т.н., ст.викл. Озеранський В.С.
Рецензент: канд. техн. наук, доцент
Черноволик Г.О.

Вінниця - 2019 року

ЗАТВЕРДЖУЮ
Завідувач кафедри____КН____
д.т.н., проф.. Яровий А.А.

_____
(підпис)
“____” _____2019 року

ЗАВДАННЯ

на магістерську кваліфікаційну роботу на здобуття кваліфікації магістра зі спеціальності: 122 – «Комп'ютерні науки»

08-22.МКР.004.19.000.ПЗ

Магістрантки групи 1КН-18м Волянської Єлизавети Валеріївни

Тема магістерської кваліфікаційної роботи: «Інформаційна технологія продуктивної організації часу»

Вхідні дані: СУБД SQLite, не менше 2 таблиць; Середовище розробки Visual Studio 2019; об’єктно – орієнтована мова програмування; операційна система сімейства Windows;.

Короткий зміст частин магістерської кваліфікаційної роботи:

1. Графічна: Граф-схема алгоритму роботи програмного забезпечення продуктивної організації часу, робочі вікна програмного засобу продуктивної організації часу, результати роботи програмного засобу продуктивної організації часу.
2. Текстова (пояснювальна записка): вступ, аналіз предметної області продуктивної організації часу, розробка інформаційної технології продуктивної організації часу, програмна реалізація інформаційної технології продуктивної організації часу, економічна частина, висновки, перелік використаних джерел, додатки.

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ МКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз сучасного рівня інформаційних технологій продуктивної організації часу. Постановка задач дослідження			Аналітичний огляд літературних джерел, задачі досліджень, розділ 1 ПЗ
2	Розробка алгоритму продуктивної організації часу			Розроблений алгоритм, розділ 2
3	Практичне застосування та оцінка ефективності розробленого алгоритму			розділ 3
4	Підготовка економічної частини			розділ 4
5	Апробація та/або впровадження результатів дослідження			тези доповідей/акт впровадження
6	Оформлення пояснювальної записки, графічного матеріалу та презентації			Пояснювальна записка, графічний матеріал, презентація

Консультанти з окремих розділів магістерської кваліфікаційної роботи

1. Науковий керівник _____ канд. техн. наук, ст.викл. кафедри КН
(підпис) наук. ступінь, вчене звання (посада)
“____” _____20___ р. _____В.С.Озеранський_____
ініціали та прізвище

2. Економічна частина _____ канд. екон. наук, доц., доц. кафедри ЕПВМ
(підпис) наук. ступінь, вчене звання (посада)
_____Бальзан М.В._____
ініціали та прізвище
“____” _____20___ р.

Дата попереднього захисту роботи “____” _____20___ р.

Рецензент _____ канд. техн. наук, доц., доц. кафедри ПЗ
(підпис) наук. ступінь, вчене звання (посада)
_____Черноволик Г.О. _____
ініціали та прізвище

Завдання видав
науковий керівник _____ канд. техн. наук, ст. викл. кафедри КН
(підпис) наук. ступінь, вчене звання (посада)
_____В.С. Озеранський_____
ініціали та прізвище
“____” _____20___ р.

Завдання отримав магістрант _____ _____Є.В. Волянська_____
(підпис) ініціали та прізвище
“____” _____20___ р.

АНОТАЦІЯ

У даній магістерській кваліфікаційній роботі описано процес створення інформаційної технології продуктивної організації часу. Проведено аналіз предметної області, здійснено постановку проблеми і задач необхідних для її вирішення. Описано практичну цінність, а також доцільність розробки програмного продукту. В роботі наведено алгоритм розробки інформаційної технології продуктивної організації часу, а також методи та засоби, що використовувались для розробки програмного продукту. Обґрунтовано економічну доцільність розробки даного програмного продукту.

ABSTRACT

This master's qualification work describes the process of creating information technology productive time organization. There was made analysis of the subject area, the statement of the problem and the tasks necessary for its solution. Were described the practical value and the expediency of software development. In the work are presented the algorithm of development of information technology of productive organization of time, and methods and tools that were used to develop the software product. There were substantiated the economic feasibility of this software.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. У сучасному світі розвиток відбувається такими темпами, що постає проблема браку часу, відсутність чіткого плану завдань, які потрібно виконати, що особливо позначається на продуктивності людини. Продуктивна організація часу, так званий тайм-менеджмент – техніка, яка спрямована правильно організувати діяльність і досягти максимальної продуктивності.

Актуальність даного дослідження полягає, в тому, що розстановка пріоритетів допоможе організувати не тільки час, але й життя в цілому. Найважче за все це зробити перший крок – зрозуміти свої прагнення і сформулювати життєві плани. Пріоритетність тієї чи іншої цілі визначає кількість часу, який їй приділяється. За допомогою оцінки важливості дій можна виділити поглиначів часу – заняття, які відволікають від поставленої цілі, не мають нічого спільного з життєвим планом і відмовитись від них. Людина стикається з великим потоком нової інформації, яка постійно оновлюється. Досягнути успіху в сучасному світі не навчившись керувати цими потоками неможливо. Питання ефективного управління пов’язано не лише з досягненням результату, але й з тим яким шляхом це досягнуто.

Тому постає питання як полегшити і зробити управління часом більш простим і менш трудомістким. Одним із шляхів вирішення цього питання є розробка інформаційної технології для продуктивної організації часу. Людина в професійному, особистому житті реалізує певні проекти, усвідомлює свої потреби і вміло переводить їх на мову цілей та задач. Людина - планувальник вміє себе організувати, тому що розуміє які кроки і в якій послідовності потрібно зробити для досягнення результату.

Так як наш світ все далі розвивається у сфері технологій і практично всі більшість часу нерозривно проводять зі смартфонами, для прикладу, то доцільним буде розробити програмний засіб, який можна було б використовувати кожен день, не докладаючи для цього зайвих зусиль.

Зв'язок роботи з науковими програмами, планами, темами. Магістерська робота виконана відповідно до напрямку наукових досліджень кафедри комп'ютерних наук Вінницького національного технічного університету 22 К1 «Моделі, методи, технології та пристрої інтелектуальних інформаційних систем управління, економіки, навчання та комунікацій» та плану наукової та навчально-методичної роботи кафедри.

Мета та завдання дослідження. Метою дослідження магістерської кваліфікаційної роботи є розширення функціональних можливостей програми організації роботи користувача з використанням інформаційної технології у порівнянні з програмами-аналогами.

Для досягнення поставленої мети необхідно розв'язати такі наступні завдання:

- провести аналіз проблеми розв'язання задачі організації часу;
- розглянути існуючі методи вирішення задачі продуктивної організації часу й обґрунтувати вибір методу, який задовольняє мету даної магістерської кваліфікаційної роботи;
- розробити алгоритм програмного засобу інформаційної технології продуктивної організації часу та удосконалити її згідно з метою роботи;
- виконати програмну реалізацію інформаційної технології продуктивної організації часу;
- провести тестування програмного продукту та виконати аналіз отриманих результатів.

Об'єкт дослідження – це процес організації часу з використанням інформаційних технологій.

Предмет дослідження – це методи та програмні засоби організації часу користувача.

Методи дослідження. У даній роботі використані наступні методи наукових досліджень: методи об'єктно-орієнтованого програмування для автоматизації організації часу, методи системного аналізу для аналізу

структури інформаційної системи, методи розробки баз даних для інформаційних систем.

Наукова новизна одержаних результатів полягає в наступному:

− удосконалено інформаційну технологію продуктивної організації часу, що відрізняється від відомих поєднанням методів тайм-менеджменту С.Кові та Д. Аллена, що дозволило збільшити функціональні можливості програми організації часу користувача.

Практичне значення одержаних результатів полягає у наступному:

1. Розроблено новий програмний засіб для продуктивної організації часу.
2. Розроблено математичну модель продуктивної організації часу
3. Розроблено алгоритм продуктивної організації часу на основі матриці С. Кові та системи GTD Д.Аллена

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується строгістю постановки задач, коректним застосуванням методів під час доведення наукових положень, порівнянням результатів з відомими.

Апробація результатів роботи. Результати роботи були апробовані на 1 міжнародній науково-практичній конференції: XII Міжнародній науково-практичній конференції «Інформаційні технології та автоматизація - 2019», (м. Одеса, Україна, 2019 р.) та опубліковані у збірниках даної конференції.

Публікації. За результатами магістерської кваліфікаційної роботи опубліковано тези доповіді конференції [1].

Результати, одержані в процесі виконання магістерської роботи, впроваджені на підприємстві ТОВ «НПК «Хоум-нет»

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ПРОДУКТИВНОЇ ОРГАНІЗАЦІЇ ЧАСУ

1.1 Постановка задачі продуктивної організації часу

Управління часом – це дія або процес тренування свідомого контролю над кількістю часу, витраченого на конкретні види діяльності, при якому спеціально збільшуються ефективність і продуктивність. Управління часом може допомогти поруч навичок, інструментів і методів, використовуваних при виконанні конкретних завдань, проектів і цілей. Цей набір включає в себе широкий спектр діяльності, а саме: планування, розподіл, постановку цілей, делегування, аналіз тимчасових витрат, моніторинг, організація, складання списків і розстановка пріоритетів. Спочатку управління приписувалося тільки бізнесу або трудової діяльності, але з часом термін розширився, включивши особисту діяльність з таким же підставою. Система управління часом складає поєднання процесів, інструментів, технік і методів. Зазвичай управління часом є основною необхідністю в розвитку будь-якого проекту, оскільки визначає час завершення проекту і масштаб.

Дефіцит робочого часу – брак часового ресурсу, викликана неправильною організацією діяльності, що призводить до поспіху, затягування виконання робіт, завдань, неякісній роботі і т. д.

Найбільш характерні причини дефіциту часу:

- безплановість роботи;
- неадекватне навантаження (потік завдань перевищує всі можливості їх вчасного виконання відведеними для цього силами і засобами);
- нечіткість формулювання завдання і кінцевих результатів;
- недостатність ресурсів для виконання завдання;

Методи планування

АВС планування

Причиною цього методу служить досвід, який наочно показує, що співвідношення важливих і неважливих справ у відсотках завжди приблизно однаково. Будь-які завдання, виходячи з їх важливості, стосовно досягнення поставлених результатів, повинні розподілятися з використанням буквених значень АВС. З цього випливає, що першим має виконуватись завдання, які мають найбільшу важливість і значимість (А), а потім вже все решта (В, С). Планувати свій час, застосовуючи дану методику, потрібно, беручи до уваги саме на важливості завдань, а не необхідні для їх виконання зусилля.

Методика АВС базується на 3 основних правилах:

- Категорія А – найважливіші справи. Вони складають приблизно 15% всіх справ, якими Ви займаєтесь, але приносять приблизно 65% результатів.
- Категорія В – важливі справи. Вони складають приблизно 20% всіх справ і приносять приблизно 20% результатів.
- Категорія С – справи найменшої значимості. Вони складають приблизно 65% всіх справ, але приносять 15% результатів.

Для використання АВС-аналізу необхідно дотримуватись наступних правил:

- скласти список всіх майбутніх завдань;
- систематизувати їх за важливістю і встановити черговість;
- пронумерувати ці завдання;
- оцінити завдання відповідно за категоріями А, В і С;
- завдання категорії А (15% загальної їх кількості) вирішує перший керівник;
- завдання категорії В (20%) підлягають передоручення;
- завдання категорії С в силу своєї мало значимості підлягають обов'язковому передорученню. [2]

Принцип Ейзенхауера

Цю методику запропонував свого часу американський генерал Дуайт Девід Ейзенхауер. Вона є відмінним додатковим вимірником для швидкого прийняття найбільш важливих рішень. Даний принцип має на увазі розстановку пріоритетів, відповідно до критеріїв важливості і терміновості.

Всі свої справи ви повинні розділити на чотири основні категорії і виконувати в порядку черговості:

- Категорія А - найбільш термінові і важливі справи.
- Категорія В - термінові, але не важливі справи. Важливо вміти відокремлювати їх за критерієм важливості від першої категорії, інакше можна витратити на їх виконання час, залишивши насправді важливі справи на потім.
- Категорія С - не термінові, але важливі справи. Тут потрібно враховувати фактор терміновості: через те, що ці справи не є терміновими, їх часто відкладають в «довгий ящик», після чого вони стають терміновими, що не дуже добре. Тому, їх виконанням ні в якому разі не можна нехтувати. Такі справи, до всього іншого, можна делегувати - доручити їх виконання комусь ще.
- Категорія D - не термінові і не важливі справи. Нерідко, такими справами людина стурбована найбільше і витрачає на їх виконання більшу частину часу. Навчіться точно визначати справи з цієї категорії. Робити їх потрібно в останню чергу, коли виконані попередні.

Принцип Ейзенхауера можна представити матрицею, зображеною на рисунку 1.1

Рисунок 1.1 – Матриця за принципом Ейзенхауера

Правило Парето

Іноді це правило називають принципом «80 на 20». Сформулював його італійський економіст Вільфредо Парето. Основний його передумовою є те, що найменша частина дій приносить найбільшу частину результатів, і навпаки.

Наочно правило виглядає так:

- 20% дій = 80% результатів
- 80% результатів = 20% результатів

Принцип Парето можна розглянути за допомогою графіка, зображеного на рисунку 1.2.

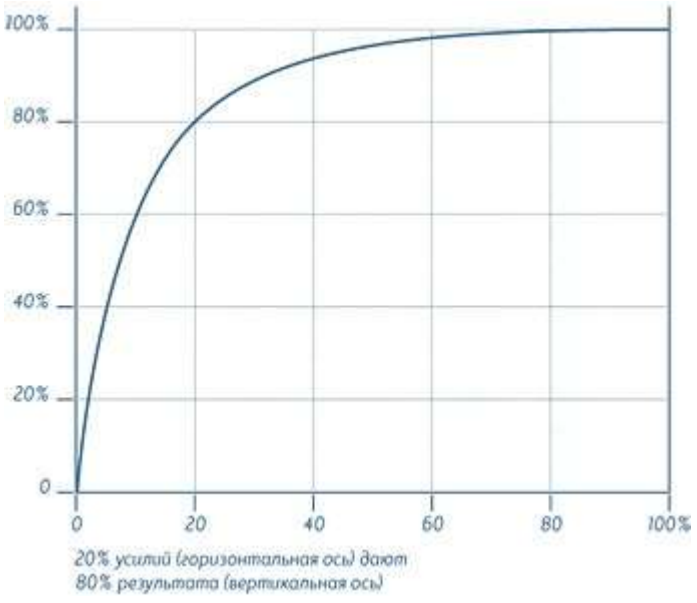

Рисунок 1.2 – Крива Парето

Техніка SMART

Техніка постановки цілей SMART визнана однією з кращих в світі. Саме слово «SMART» є абревіатурою, утвореної з перших літер п'яти слів, що визначають критерії цілей. Розглянемо їх трохи докладніше.

1. Specific - мета повинна бути конкретною, тобто при її постановці ви повинні обов'язково чітко уявляти собі результат, якого бажаєте досягти. Наприклад, «я хочу стати фахівцем в області антропології».

2. Measurable - мета повинна бути вимірною, тобто ви повинні представляти бажаний результат в кількісному еквіваленті. Наприклад, «до 2015 року я хочу заробляти 50 тисяч щомісяця».

3. Attainable - мета повинна бути досяжною, тобто ви повинні враховувати особливості своєї особистості: здатності, схильність, талант і т.д. Наприклад, якщо у вас туго з математикою і ви абсолютно не розумієте цю науку, краще не ставити собі за мету стати видатним математиком.

4. Relevant - мета повинна співвідноситися з іншими вашими завданнями. Наприклад, досягнення середньострокової мети повинно підспудно включати в себе досягнення декількох короткострокових.

5. Time-bounded - мета повинна бути визначена в часі і мати чітко встановлені тимчасові рамки. Наприклад, «я хочу скинути вагу з 95 до 80 кг через півроку до такого-то місяця» [3]

1.2 Огляд методів організації часу користувача

Тайм-менеджмент має на увазі використання щоденників, планерів, створення списків завдань, але не обмежується ними. Під системою ми розуміємо цілісну структуру взаємопов’язаних деталей, де все вищеперераховане є лиш елементом.

В цілому ж система управління часом – це спеціальна методика, найчастіше з власним інструментарієм, а також рекомендаціями і порадами по ефективній організації своєї діяльності. Її задача – не просто нагадати про якесь завдання чи зустріч, розпланувати день, але й показати як це ефективно зробити, щоб не тільки закінчити роботу с термін, але й досягти більшого. В класичному понятті систем тайм-менеджменту закладена суть: вона допомагає людині і планувати зайнятість, і управляти часом, що витрачається на різні потреби.

Сьогодні існує приблизно десятка відомих систем управління часом і багато варіацій, заснованих на особистому досвіді їх застосування. Кожна з них має свої переваги, тому розглянемо кожну з них.

Піраміда Франкліна

Структура системи представляється як піраміда, де кожен елемент залежить від попереднього. Фундаментом виступають життєві цінності планувальника. Це те, що важливо для людини, те, навколо чого будується його життєва концепція. Наступний блок – глобальні цілі – конкретне вираження результату, головна ціль. На основі глобальних цілей будується генеральний план по їх досягненню – своєрідна узагальнена інструкція до того, як може бути досягнута ціль. Далі встановлюється більш конкретний план, на 3-5 років, який включає в себе конкретні кроки. З урахуванням масштабу поставлених задач складаються середньостроковий і короткостроковий плани. Таким чином, виходить чітка і зрозуміла картина того, як досягти цілі всього життя. Ця система будується по принципу «верху вниз» - від визначення глобальних життєвих задач до більш конкретних кроків досягнення успіху.

Перевагою піраміди є її наочність і простота. Недоліком – те, що вона н гнучка до змін і позбавлена деталізації в плані управління справами, адже більш спрямована на їх планування. Схема піраміди Франкліна зображена на рисунку 1.3

Рисунок 1.3 – Схема піраміди Франкліна

Матриця управління часом Стівена Кові

Стівен Кові – один з найбільш відомих сучасних послідовників і популяризаторів піраміди Франкліна як техніки планування життя і досягнення поставлених цілей. Він також є автором власної методики, яка доповнює піраміду, перекриваючи її слабкі сторони(що стосується, в першу чергу, продуктивного управління щоденними справами).

Створена С. Кові матриця управління часом є чудовим інструментом для планування зайнятості і визначення «крадіїв часу» - дій, через які він втрачається даремно. По суті це готовий шаблон, куди кожен, в залежності з запропонованою класифікацією, відносить задачі і справи, з якими щоденно стикається. Матриця Кові зображена на рисунку 1.4

Матриця тайм-менеджменту

	Терміново	Нетерміново
Важливо	I ЗАНЯТТЯ Конфлікти. Нагальні проблеми. Проекти з дедлайнами	II ЗАНЯТТЯ Профілактика, діяльність, пов'язана з ПМ. Налагодження стосунків. Визначення нових можливостей. Планування, відпочинок
Неважливо	III ЗАНЯТТЯ Перерви, деякі дзвінки. Деяка пошта, деякі звіти. Деякі зустрічі. Нагальні в найближчому майбутньому проблеми. Популярна діяльність	IV ЗАНЯТТЯ Дрібниці, подоба праці. Деяка пошта. Деякі дзвінки. «Пожирачі часу». Приємні заняття

Рисунок 1.4 – Матриця Стівена Кові

Проаналізувавши на основі матриці свої справи і визначивши пріоритети, можна починати будувати піраміду. Але ні в якому разі в план не потрібно включати задачі лише з квадранта II – таким чином створити ефективну схему не вийде, оскільки ігнорувати інші справи не можна, а уникнути їх в силу різних факторів неможливо.

GTD Девіда Аллена

Якщо спробувати описати метод Алена одним реченням, то можна сказати, що основна рекомендація – максимально використовувати сучасні технічні інструменти для цілей планування і управління часом. Це не відобразить всю суть GTD, але дозволить скласти уявлення хоча б про частину техніки. Насправді, система тайм-менеджменту Д. Аллена відома, в першу чергу, своїми технічними «вивертами», ціллю яких є спрощення управління проектами.

Ще до створення проекту автор був впевнений, що сама організація роботи ніколи не повинна займати більше часу ніж сама робота. Тому в основу

системи лягло упередження, що максимальні зусилля необхідно прикладати до практичного виконання завдань, а не пам’ятати все, що потрібно зробити.

На відміну від С. Кові автор GTD вважає головним не виділяти пріоритети, а контролювати процес виконання задач і на основі результатів будувати бачення майбутнього. Очевидно, що це діаметрально протилежно піраміді Франкліна, адже Аллен пропонує закінчувати глобальними планами, а не починати з них. Для досягнення успіху в даному відношенні він пропонує застосовувати 3 самостійні моделі:

- Управління робочим процесом має на увазі контроль над усіма обов’язками і задачами і реалізується через 5 етапів: збір, обробка, організація, огляд, дії;
- 6-рівнева модель огляду роботи для бачення перспектив (поточні справи, поточні проекти, коло обов’язків, найближчі роки, п’ятирічна перспектива, життя);
- Природний метод планування (визначення цілі і пріоритетів, бачення бажаних результатів, мозковий штурм, організація, визначення наступної конкретної дії).

Алгоритм обробки вхідних повідомлень згідно з методикою GTD зображено на рисунку 1.5

Безперечним недоліком GTD як системи управління часом є те, що вона найбільш цілісна, пропонує готовий набір інструкцій, слідуючи яким можна привести справи в порядок. Д. Аллен також пропонує багато практичних інструментів, що сприяють цьому. З іншої сторони GTD потребує жорсткої самодисципліни і постійної роботи над підтримкою особистої системи в актуальному стані. До того ж, є думка, що вона не зовсім підходить для людей творчих професій. [4]

Рисунок 1.5 - Алгоритм обробки вхідних повідомлень згідно з методикою GTD

Інтуїтивний фокус Марка Форстера

Марк Форстер пропонує скласти абсолютно упорядкований список справ (так, щоб на один лист поміщати не більше 25 пунктів) і слідувати інструкції:

- Переглядайте сторінку повільно, поки ваша увага не зупиниться на одному з пунктів.
- Працюйте над цим пунктом, поки не втомитеся від даної діяльності.
- Потім викресліть його або допишіть в кінець списку, якщо не встигли доробити.
- Повторюйте до тих пір з вашим списком кожен день, поки не викреслите всі пункти.
- Переходьте до наступного листу.
- Повторюйте ці пункти з дня в день, поки список не закінчено.

На перший погляд здається, що так виконуються тільки найлегші і приємні справи, але насправді після них набагато простіше перейти до

складних завдань - вони перестають здаватися непосильними. До того ж, цей метод допомагає позбавлятися від зайвих справ. [5]

Ми оглянули основні методи організації часу користувача, а саме:

- Піраміда Франкліна
- Матриця Стівена Кові
- GTD Девіда Аллена
- Інтуїтивний фокус Марка Форстера

В результаті для розробки інформаційної технології продуктивної організації часу було обрано базові поняття системи Стівена Кові, а також GTD Девіда Аллена. Тобто наша система буде побудована на наступних принципах:

- Кожному завданню необхідно призначити рівень важливості, для цього будемо виставляти пріоритет: термінове, важливе, не термінове;
- Кожне завдання повинне буди відсортоване відповідно певної категорії: робота, сім’я, саморозвиток і т.д
- Виконані завдання необхідно відмічати, щоб бачити результат виконання
- Важливою складовою є ведення статистики, а саме відслідковування на які завдання витрачається найбільше часу, щоб зробити висновки про правильність використання часу і визначити завдання – поглиначі часу витрати часу на які треба зменшити.

1.3 Аналіз існуючих програмних засобів продуктивної організації часу

Для того, щоб зрозуміти, що саме потрібно користувачу для організації свого часу було проведено пошук існуючих програмних засобів, здійснено аналіз переваг та недоліків виявлених користувачами знайдених додатків. Під

час пошуків аналогів для вирішення поставленої задачі було виявлено наступні програмні засоби:

- Microsoft To Do
- Ike - To-Do List, Task List
- Мої Завдання: Планувальник завдань
- Завдання: список справ, додаток списку завдань

Головні можливості програми Microsoft To Do включають в себе:

- ефективний розпорядок дня з функцією "Мій день", персоналізованим щоденним планувальником завдань;
- перегляд списків будь-де та на будь-якому пристрої;
- надання спільного доступу до списків і завдань друзям, рідним, колегам і однокласникам;
- створення забарвлених списків;
- установлення одноразових або повторюваних термінів і нагадувань;
- розділення завдань на окремі етапи;
- додавання нотаток до завдань;
- додавання вкладень (до 25 МБ) до завдань;
- синхронізація завдань між Outlook і To Do; [6]

Недоліками даного додатку є наступні:

- відсутність роботи з календарем
- недостатньо продуманий інтерфейс

Вигляд програмного засобу зображено на рисунку 1.6

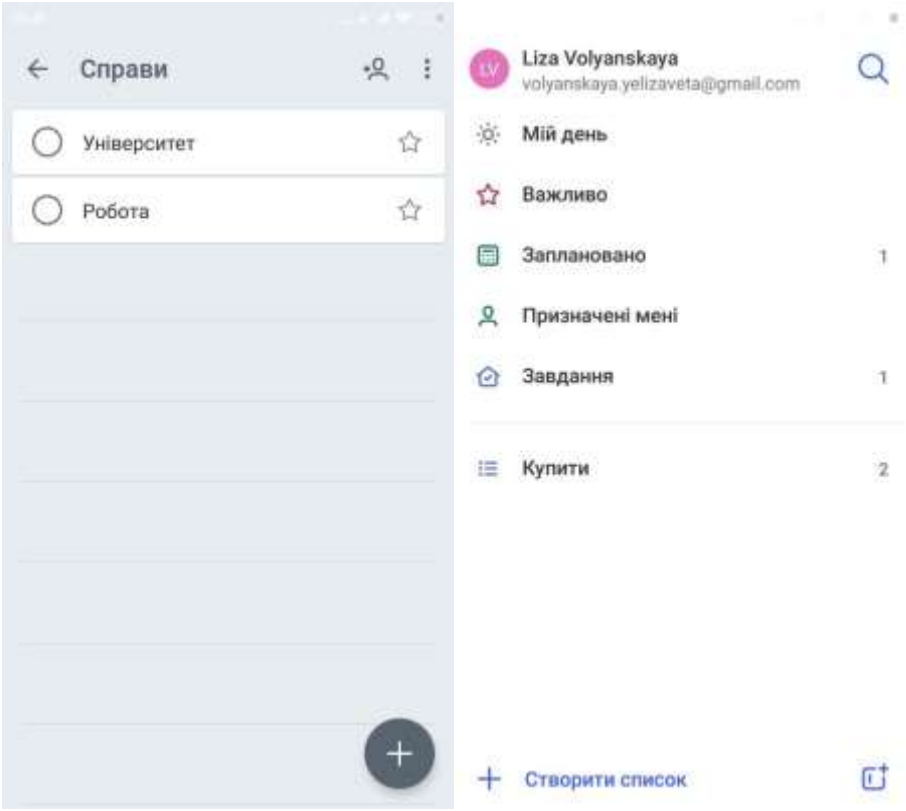

Рисунок 1.6 – Вигляд програмного засобу «Microsoft To Do»

Наступний програмний засіб із існуючих аналогів має назву Ike - To-Do List, Task List.

Даний продукт має наступні можливості:

- пріоритетність задач
- встановлення термінів виконання задачі
- нагадування про кожне завдання
- додавання аудіо до завдань
- налаштування оформлення під користувача[7]

Недоліками даного продукту є:

- відсутність синхронізації з аккаунтами користувача
- відсутність вибору іншої мови програмного середовища
- для доступу до всіх функцій потрібна оплата

Вигляд програмного засобу зображено на рисунку 1.7

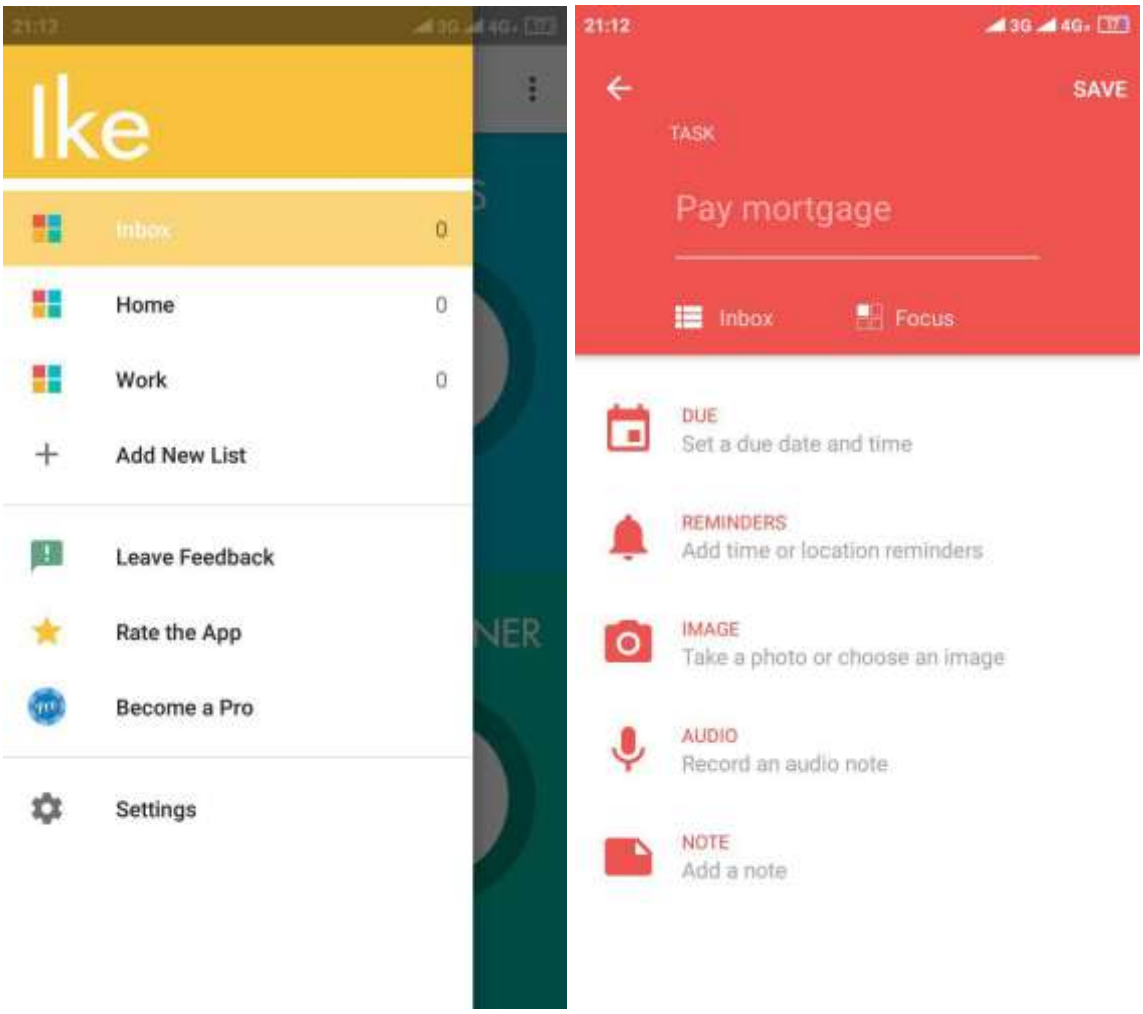

Рисунок 1.7 - Вигляд програмного засобу «Ike - To-Do List, Task List»

Програмний засіб «Мої Завдання: Планувальник завдань» має наступні можливості:

- відображення завдань на цілий тиждень
- перелік завдань на день
- відмітка виконаних завдань
- додавання завдань без прив’язки до дати
- переміщення завдань в списку
- наявність календаря [8]

Недоліками програмного продукту є:

- відсутність поділу завдань на категорії
- недостатньо продуманий дизайн

- відсутність резервного копіювання

Вигляд програмного засобу «Мої Завдання: Планувальник завдань» зображено на рисунку 1.8

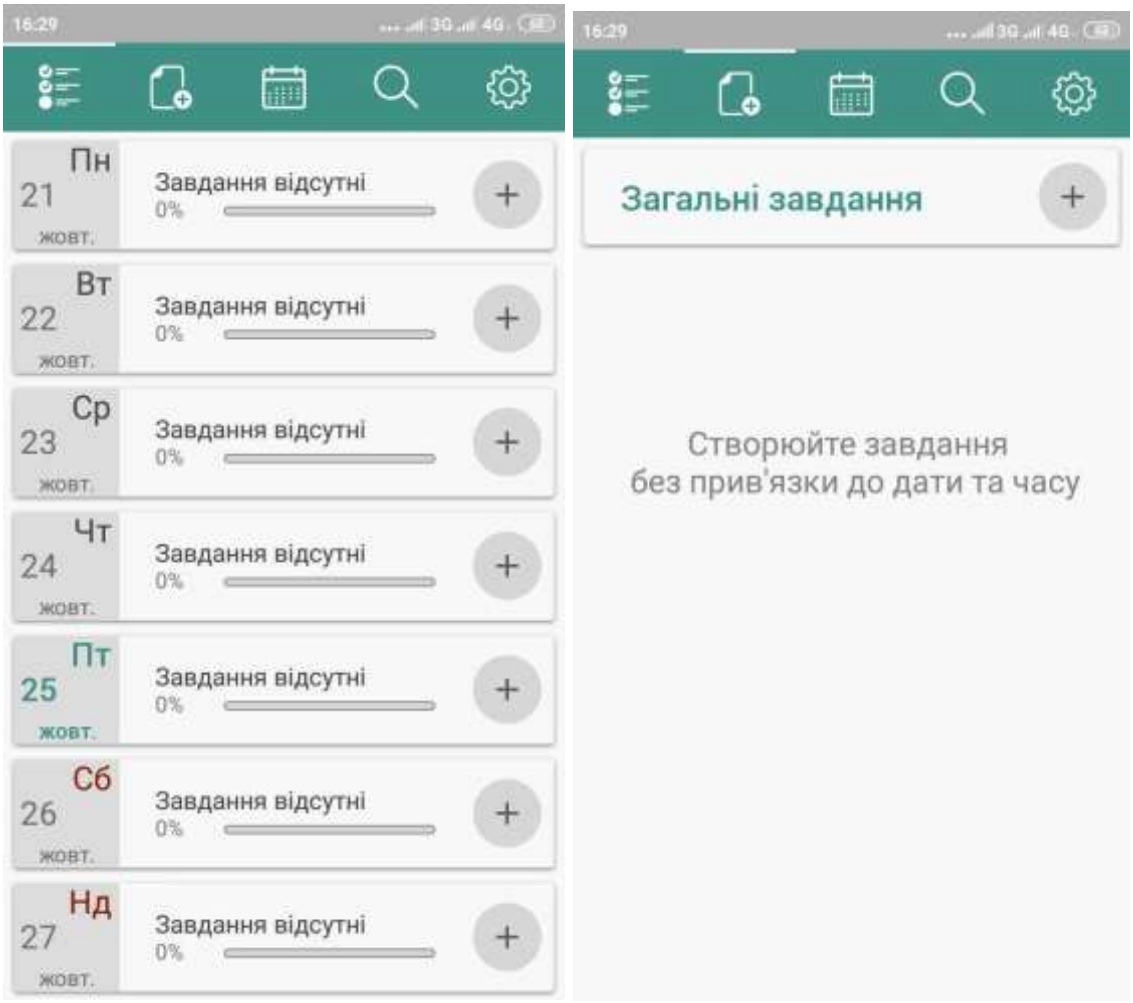

Рисунок 1.8 - Вигляд програмного засобу «Мої Завдання: Планувальник завдань»

Програмний засіб «Завдання: список справ, додаток списку завдань» має наступні можливості:

- організація завдань по категоріям
- віджет на головному екрані
- додавання завдань у 2 режимах
- управління інтуїтивними жестами [9]

Недоліками програмного продукту є:

- відсутність пріоритетності завдань
- відсутність фільтрації завдань по дням, лише по категоріям
- відсутність інтеграції з календарем

Вигляд програмного засобу «Завдання: список справ, додаток списку завдань» зображено на рисунку 1.9

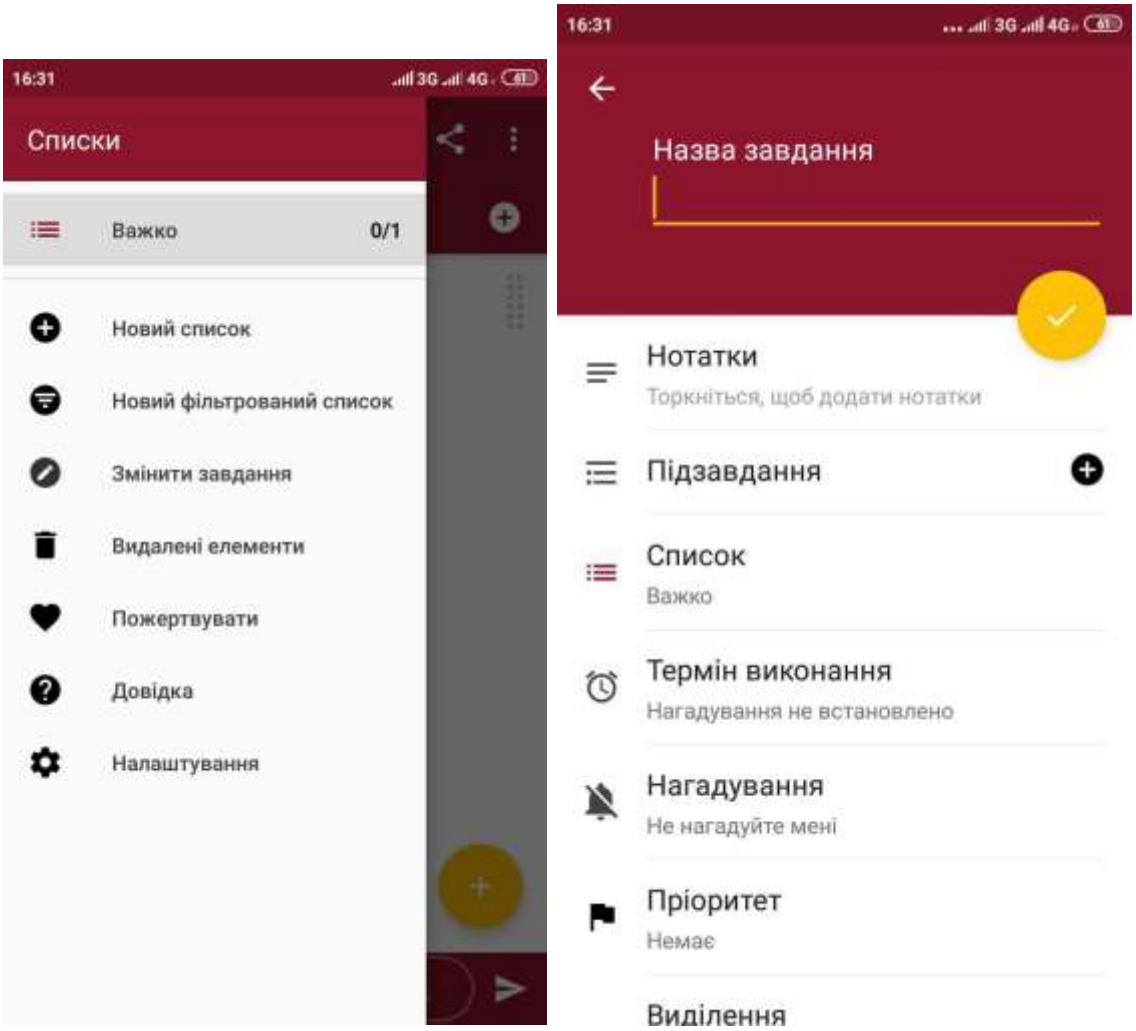

Рисунок 1.9 - Вигляд програмного засобу «Завдання: список справ, додаток списку завдань»

На основі вище перерахованих аналогів було сформовано основні функціональні можливості прогами, що розробляється, а саме:

- наявність зручного і простого у використанні інтерфейсу
- розбиття завдань на категорії

- встановлення пріоритетності завдань
- можливість планування завдань з прив’язкою до часу і без
- планування завдань на будь-який день місяця
- система сповіщень про завдання
- виведення завдань на робочий стіл користувача
- можливість зберігання та відновлення списків завдань
- відмітка виконаних завдань
- статистика по типу виконуваних задач

Наявність даних функцій повинно надавати користувачеві можливість швидше і ефективніше організовувати свій час і вирішувати завдання вчасно, а зручний і легкий для розуміння інтерфейс повинен допомогти користувачу у вирішенні даних завдань.

1.4 Висновок

У даному розділі було здійснено аналіз предметної області продуктивної організації часу, описано основні методи управління часом, а також пошук існуючих аналогів програмного засобу, а саме: Microsoft To Do, Ike - To-Do List, Task List, Завдання: список справ, додаток списку завдань, Мої Завдання: Планувальник завдань. Описано головні можливості аналогів та виявлено основні недоліки. В результаті чого було сформовано основний функціонал, яким повинен володіти програмний продукт, який розробляється: зручний інтерфейс, система сповіщень про завдання, зберігання та відновлення даних, виведення завдань на робочий стіл, планування завдань з прив’язкою до часу і без, відмітка виконаних завдань, ведення статистики для аналізу використання часу. На основі аналізу методів було обрано базові принципи матриці Стівена Кові (розбиття завдання на категорії з встановленням пріоритетності) та системи GTD Девіда Аллена (збір, аналіз, сортування

інформації з метою подальшої обробки) для подальшої розробки програмного продукту.

2 РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ПРОДУКТИВНОЇ ОРГАНІЗАЦІЇ ЧАСУ

2.1 Розробка математичної моделі продуктивної організації часу

Основні життєві позиції людини можна сформулювати у вигляді її стратегічних цілей, які в свою чергу можна поділити на більш конкретні (тактичні) цілі, для досягнення яких треба виконати ряд задач, що й будуть складати розклад людини.

Отже, основними сутностями для моделювання є цілі (G) та задачі (Z). Сукупність цілей людини має ієрархічну структуру, отже модель цілей може бути представлена формулою 2.1

$$MG = < G, EG >, \tag{2.1}$$

де G – множина цілей, EG – множина зв’язків між ними.

Множина цілей

$$G = \{G_{k,j}^{i}\}, i = \overline{l,n}; j = \overline{l,m}, \tag{2.2}$$

де n – кількість рівнів у ієрархії цілей, m – кількість підцілей цілі G_k^{i-1} (для i = 1,2, k = {Ø}).

Множина зв’язків між цілями може бути представлена як множина дуг(2.3):

$$EG = \{EG(G_{k,j}^{i}, G_{kj,l}^{i+1})\}. \tag{2.3}$$

Тактичні цілі, що не мають підцілей, тобто такі $G_{k,j}^{i}$,для таких не існує зв'язків $(G_{k,j}^{i}, G_{kj,l}^{i+1})$ є «листами» дерева, і для них повинна бути визначена множина задач, виконання яких призведе до досягнення цієї цілі. Цілі, які не є листами дерева, є стратегічними або ключовими.

Згідно практикам тайм-менеджменту, кожна ціль $G_{k,j}^{i}$ може мати наступні характеристики:

- Важливість цілі $W(G_{k,j}^{i})$ для людини;
- Термін досягнення цілі $T\left(G_{k,j}^{i}\right)$, який встановлюється людиною;
- Задачі для досягнення цілі-листа $G_{k,j}^{i}$ – множина

$$Z_{k,j}^{i} = \{Z_l\}, l = \overline{l,p}. \tag{2.4}$$

Термін досягнення цілі не може бути більшим за термін її надцілі, тобто якщо $\exists(G_{k,j}^{i}, G_{kj,l}^{i+1})$, то $T\left(G_{k,j}^{i}\right) \geq T(G_{kj,l}^{i+1})$.

Між підцілями $\{G_{k,j}^{i+1}\}$ цілі G_{j}^{i}, з урахуванням своїх переваг, людина може встановити відношення порядку $P(G_{k,j1}^{i+1}, G_{k,j2}^{i+1})$, яке означає , що ціль $G_{k,j1}^{i+1}$ є більш переважною для неї ніж ціль $G_{k,j2}^{i+1}$, тобто $G_{k,j1}^{i+1} > G_{k,j2}^{i+1}$. Операція встановлення порядку є достатньо легкою для людини. Для отримання числового еквіваленту такої переваги введемо поняття рангу $R_k, k = \overline{1,m}$, m – кількість підцілей цілі G_{j}^{i}. $R_k \in [1, m]$.

Ранг може бути присвоєний цілі відповідно до відношення порядку P цілі $G_{k,j}^{i+1}$. Встановлена ієрархія цілей дозволяє ввести наступні залежності:

- Важливість глобальної цілі $W(G_1^1) = 1$;
- Якщо $\exists P(G_{k,j1}^{i+1}, G_{k,j2}^{i+1})$, то $W\left(G_{k,j1}^{i+1}\right) > W\left(G_{k,j2}^{i+1}\right)$;
- Сумарна важливість усіх підцілей $\{G_{j,k}^{i+1}\}$ будь-якої цілі G_{j}^{i} дорівнюватиме її важливості:

$$\sum_{k=1,m} W\left(G_{j,k}^{i+1}\right) = W(G_j^i).$$

(2.5)

Виходячи з цього, важливість цілі може бути розрахована за формулою:

$$W\left(G_{j,k}^{i+1}\right) = R_k \,/\, \sum_{q=1,m} R_q.$$

(2.6)

Сукупність задач, які людина повинна виконати для досягнення певної цілі, може бути представлена у вигляді орієнтованого графа, що задає послідовність виконання пов'язаних задач. Отже, сукупність задач може бути представлена наступним чином: $MZ =< Z, EZ >$, де Z – множина задач, EZ – множина зв'язків між ними. При цьому множина задач $Z = \left\{\{Z_1\}_{k,j}^i\right\}, 1 = \overline{1,p}$, де p – кількість задач цілі-листа $G_{k,j}^i$, а множина зв'язків між задачами $EZ = \{EZ\left(Z_{k,j,l}^i, Z_{x,z,v}^y\right)\}$.

Існування зв'язку $(Z_{k,j,1}^i, Z_{x,z,v}^y)$ означає, що задача $Z_{x,z,v}^y$ може бути розпочата лише після завершення задачі $Z_{k,j,1}^i$.

При цьому виконується правило: якщо $\exists(Z_{k,j,1_1}^i, Z_{x,z,v}^y)$, то не $\exists(Z_{x,z,v}^y, Z_{k,j,1_1}^i)$.

Крім того, граф задач MZ не повинен містити циклів.

Тобто не може існувати такого шляху, початкова і кінцева задача в якому співпадають, тобто не $\exists(Z_{k,k,l_1}^i, Z_{k,k,l_2}^i, \ldots, Z_{k,k,l_n}^i, Z_{k,k,l_1}^i)$.

Згідно практик тайм-менеджменту задача $Z_{k,k,1}^i$ характеризується наступними параметрами:

- Ціль $G_{k,j}^i$, для вирішення якої необхідно виконати задачу;
- Категорія задачі $K\left(Z_{k,j,l}^i\right) \in \{K_m\}$, наприклад, дзвінок, зустріч, заняття та інші;
- Відносна важливість задачі $V(Z_{k,j,l}^i)$;
- Дедлайн задачі $d(K_{k,j,l}^i)$, де $d\left(Z_{k,j,l}^i\right) \leq T(G_{k,j}^i)$;
- Точний час її завершення $t\left(Z_{k,j,l}^i\right)$ (для «жорстких» задач);

- Тривалість $1(Z^i_{k,j,l})$.

Відносна важливість задачі $t(Z^i_{k,j,l})$ традиційно може бути отримана на базі відношення порядку, яке людина може встановити на множені задач. Між тим, така важливість не враховує ціль, до якої прив'язана задача, бо людина практично не здатна врахувати ієрархічні відношення цілей під час порівняння задач. Запропоновано ввести поняття абсолютної важливості $W(Z^i_{k,j,l})$ задачі, що врахує важливість цілі, до якої належить задача, отже:

1) Сума абсолютних важливостей усіх задач множини $Z^i_{k,j} = \{Z_1\}$, що належать до цілі $G^i_{k,j}$, дорівнюватиме важливості самої цілі: $\sum_{1=1.p} W\left(Z^i_{k,j,l}\right) = W(G^i_{k,j})$, де p- кількість задач цілі-листа $G^i_{k,j}$;

2) Абсолютна важливість задачі $Z^i_{k,j}$ розраховується як $W\left(Z^i_{j,k,1}\right) = V\left(Z^i_{j,k,l}\right) * \frac{W(G^i_{j,k})}{\sum_{l=1,p} V(Z^i_{j,k,l})}$

Введення поняття абсолютної важливості задачі може бути використано під час побудови оптимізаційної моделі розкладу. У якості критерію оптимізації розкладу може бути обрано сумарну абсолютну важливість його задач $\sum_{i=l,p} W(Z_i) \to max$

В роботі було запропоновано математичну модель інформаційної технології продуктивної організації часу, що повинна бути виконана для досягнення цілей. Дана модель дозволяє описати стратегічні та тактичні цілі людини та врахувати їх під час визначення важливості задач, що входять до складу особистого розкладу людини, щоб досягти максимально ефективного їх виконання. Запропонована модель дозволяє формалізувати методики тайм-менеджменту та розробити алгоритми для складання розкладу, найбільш оптимального з точки зору абсолютної важливості задач, швидшого досягнення цілей або інших критеріїв, що будуть враховувати різні життєві позиції людини.

2.2 Розробка алгоритму організації часу користувача

Важливим етапом в розробці програмного продукту є алгоритм роботи - набір інструкцій, які описують порядок дій виконавця. На основі продуманого алгоритму реалізація програмного забезпечення стає чіткою і більш простою. В минулому розділі ми розглянули і обрали методи продуктивної організації часу. Для організації часу користувача було обрано основи систем тайм-менеджменту С. Кові та Д. Аллена. В основі яких лежать: розстановка пріоритетів задач, шляхом поділення завдань на різні категорії:

- Термінові
- Важливі
- Не термінові

А також сортування завдань шляхом проходження етапів за Д. Алленом:

- Збір даних
- Прояснення
- Організація
- Перегляд
- Виконання

На основі даних методів було розроблено алгоритм організації часу, зображений на рисунку 2.1.

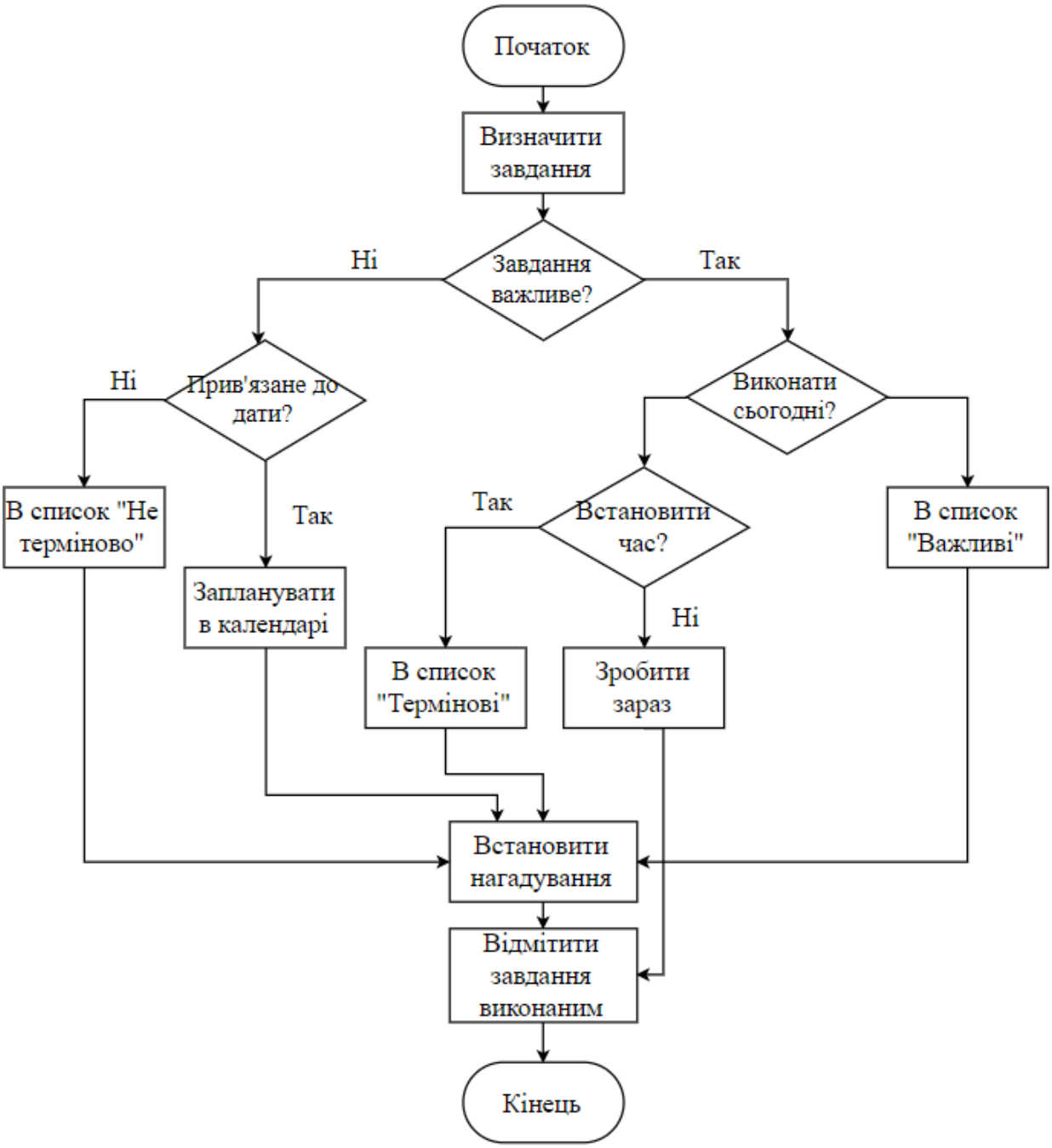

Рисунок 2.1 - Схема алгоритму організації часу користувача

Алгоритм можна описати наступними кроками:

1. Визначити завдання, які потрібно виконати
2. Якщо завдання важливе, то вирішуємо чи потрібно його виконати сьогодні, якщо ні то визначаємо чи прив'язано воно до конкретної дати

3. Якщо необхідно виконати сьогодні, то визначаємо чи воно прив'язано до конкретного часу.

4. Якщо залежить від часу, то додаємо в список Термінові і встановлюємо нагадування. Якщо не залежить, то виконуємо завдання і переносимо його у Завершені.

5. Якщо завдання важливе, але не потребує виконання сьогодні, то додаємо його у список Важливі.

6. Якщо завдання не важливе, то визначаємо чи прив'язане воно до дати.

7. Якщо так то плануємо його у календарі на відповідний день, якщо ні, то додаємо у список Не термінові. Встановлюємо нагадування.

8. Після виконання завдань ставимо відмітку про завершення завдання.

На основі даного алгоритму буде здійснюватися подальша розробка програмного забезпечення.

2.3 Розробка UML діаграм

Діаграма послідовності є однією з різновиду діаграм взаємодії та призначена для моделювання взаємодії об'єктів системи в часі, а також обміну повідомленнями між ними.

Одним з основних принципів ООП є спосіб інформаційного обміну між елементами системи, що виражається у відправці і отриманні повідомлень один від одного. Таким чином, основні поняття діаграми послідовності пов'язані з поняттям об'єкт і повідомлення.

На діаграмі послідовності об'єкти в основному являють екземпляри класу або суті, володіють поведінкою. Як об'єкти можуть виступати користувачі, які ініціюють взаємодію, класи, що володіють поведінкою в системі або програмні компоненти, а іноді і системи в цілому. Об'єкти

розташовуються зліва на права таким чином, щоб крайнім зліва був той об'єкт, який ініціює взаємодію.

Невід'ємною частиною об'єкта на діаграмі послідовності є лінія життя об'єкта. Лінія життя показує час, протягом якого об'єкт існує в Системі. Періоди активності об'єкта в момент взаємодії показуються за допомогою фокуса управління. Тимчасова шкала на діаграмі спрямована зверху вниз.

На діаграмах послідовності допустимо використання стандартних стереотипів класу:

- Actor - екземпляр учасника процесу (роль на діаграмі прецедентів)

- Boundary - Об'єкт boundary показує саме екранну форму, яка приймає і передає дані обробнику.

- Control - активний елемент, який використовуються для виконання деяких операцій над об'єктами (програмний компонент, модуль, обробник)

- Entity - Клас-сутність - зазвичай застосовується для позначення класів, які зберігають певну інформацію про бізнес-об'єктах (відповідає таблиці або елементу БД)

Також одним з основних понять, пов'язаних з діаграмою послідовності, є Повідомлення. На діаграмі діяльності виділяються повідомлення, які ініціюють ту чи іншу діяльність або є її наслідком. На діаграмі станів частково показаний обмін повідомленнями в рамках повідомлень ініціюють зміну стану об'єкта.

Діаграма послідовності об'єднує діаграму діяльності, діаграму станів і діаграму класів. Таким чином, на діаграмі послідовності ми можемо побачити такі аспекти:

- Повідомлення, які спонукають об'єкт до дії
- Дії, які викликаються повідомленнями (методи) - найчастіше це передача повідомлення наступного об'єкти або повернення певних даних об'єкта
- Послідовність обміну повідомленнями між об'єктами [10]

Діаграма послідовності зображена на рисунку 2.2

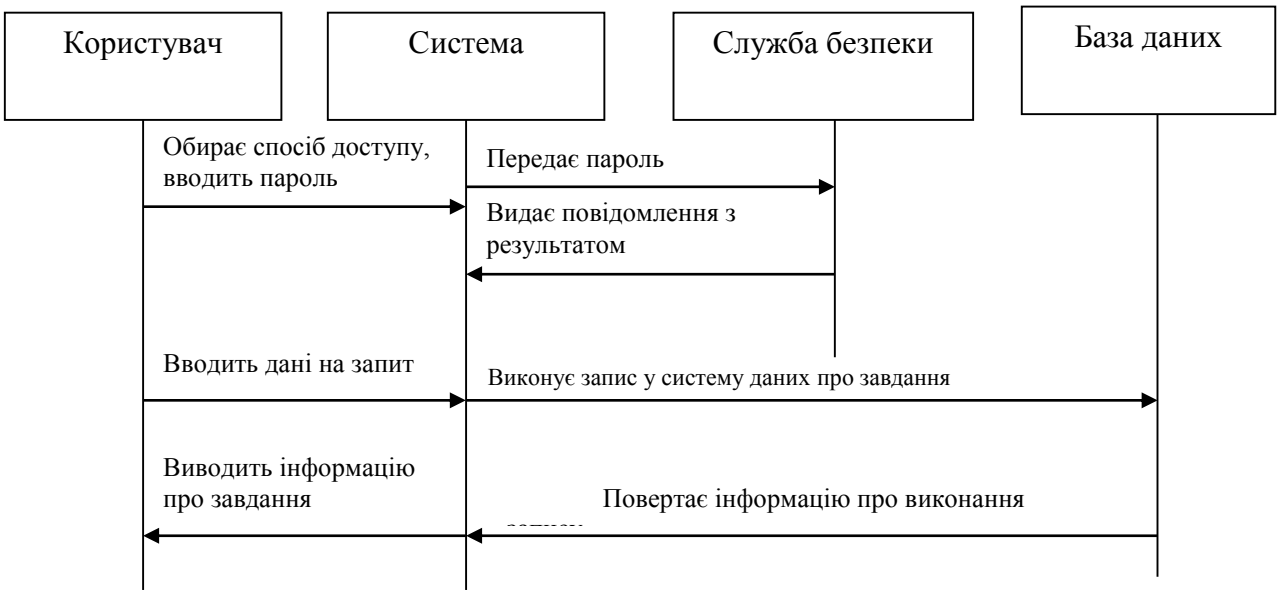

Рисунок 2.2 - Діаграма послідовності

В якості графічного уявлення для виділення основних функцій Системи ми застосовуємо діаграму варіантів використання (use case).

Діаграма варіантів використання дає нам уявлення ЩО повинна робити система. На питання ЯК ми можемо відповісти, використовуючи діаграму активності.

Тобто якщо варіанти використання ставлять перед Системою мету, то діаграма діяльності показує послідовність дій, необхідних для її досягнення. Дії (action) це елементарні кроки, які не передбачають подальшу декомпозицію.

Діяльність може містити вхідні або вихідні дуги діяльності, що показують потоки управління і потоки даних. Якщо потік з'єднує дві діяльності, він є потоком управління. Якщо потік закінчується об'єктом, він є потоком даних.

Діяльність виконується, тільки тоді, коли готові всі його «входи», після виконання, діяльність передає управління і (або) дані на свої «виходи». Саму

діаграму діяльності прийнято розташовувати таким чином, щоб дії слідували зліва направо або зверху вниз.

Щоб вказати, де саме знаходиться процес, використовується абстрактна точка «маркер» (або «токен»). Візуально на діаграмі маркер не відображається, дане поняття вводиться тільки для зручності опису динамічного процесу.

Перехід маркера здійснюється між вузлами. Маркер може не містити ніякої додаткової інформації (порожній маркер) і тоді він називається маркером управління (control flow token) або ж може містити посилання на об'єкт або структуру даних, і тоді маркер називається маркером даних (data flow token).

Діаграма діяльності зображена на рисунку 2.3.

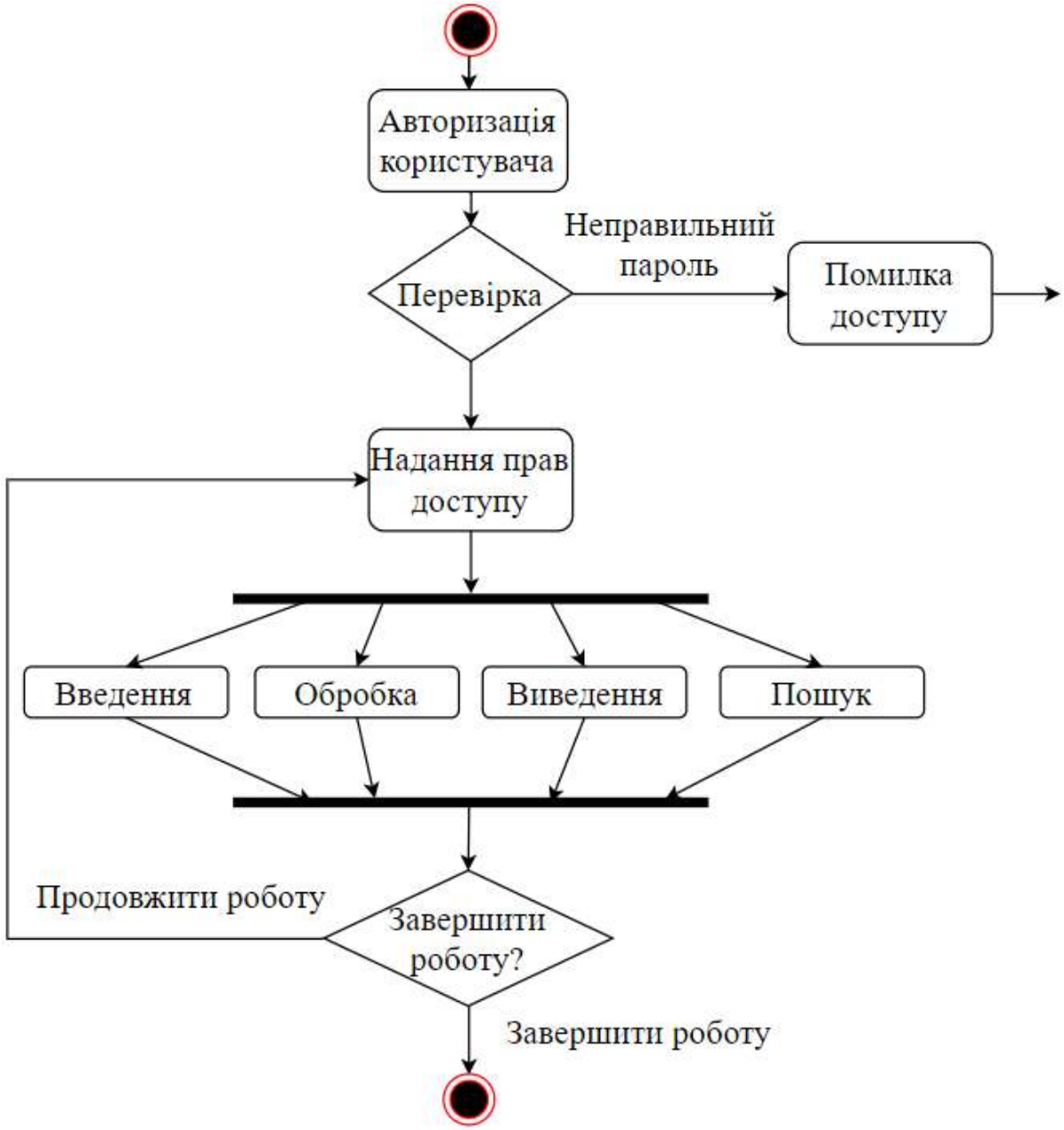

Рисунок 2.3 - Діаграма діяльності

Для створення діаграми діяльності використовуються наступні вузли:

- Вузол управління (control node) - це абстрактний вузол дії, яке координує потоки дій
- Початковий вузол діяльності (або початковий стан діяльності) (activity initial node) є вузлом управління, в якому починається потік (або потоки) при виклику даної діяльності ззовні

- Кінцевий вузол діяльності (або кінцевий стан діяльності) (activity final node) є вузлом управління, який зупиняє (stop) все потоки даної діаграми діяльності. На діаграмі може бути більш одного кінцевого вузла
- Кінцевий вузол потоку (або кінцевий стан потоку) (flow final node) є вузлом управління, який завершує даний потік. На інші потоки і діяльність даної діаграми це не впливає

Об'єкт, над якими виконуються дії. Це не обов'язковий елемент діаграми, але в деяких випадках необхідно показати об'єкт ініціює виконання дій, або є результатом його. [11] Діаграма діяльності зображена на рисунку 2.3

2.4 Висновок

У даному розділі було проведено пошук та аналіз відомих систем тайм-менеджменту, в результаті чого було розглянуто наступні системи:

- піраміда Франкліна
- матриця Стівена Кові
- система GTD Девіда Аллена
- інтуїтивний фокус Марка Форстера.

Також наведено схему алгоритму організації часу на основі обраних систем, діаграму послідовностей, діаграму діяльності та описано математичну модель.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ПРОДУКТИВНОЇ ОРГАНІЗАЦІЇ ЧАСУ

3.1 Обґрунтування вибору мови та середовища програмування

Для розробки інформаційної технології продуктивної організації часу було обрано наступні засоби:

- Середовище програмування Microsoft Visual Studio
- Інструмент Xamarin.Forms
- Об’єктно-орієнтована мова програмування C#
- Система керування базами даних SQLite

Microsoft Visual Studio - це програмне середовище з розробки додатків для ОС Windows, як консольних, так і з графічним інтерфейсом.

У комплект входять наступні основні компоненти:

- Visual Basic.NET - для розробки додатків на VisualBasic;
- Visual C ++ - на традиційній мові C ++;
- Visual C # - на мові C # (Microsoft);
- Visual F # - на F # (Microsoft Developer Division).

Функціональна структура середовища включає в себе:

- редактор вихідного коду, який включає безліч додаткових функцій, як автодоповнення IntelliSense, рефракторінг коду і т. д .;
- відладчик коду;
- редактор форм, призначений для спрощеного конструювання графічних інтерфейсів;
- веб-редактор;
- дизайнер класів;
- дизайнер схем баз даних.

Visual Studio також дозволяє створювати і підключати сторонні додатки (плагіни) для розширення функціональності практично на кожному рівні,

включаючи додавання підтримки систем контролю версій вихідного коду (Subversion і VisualSourceSafe), додавання нових наборів інструментів (для редагування і візуального проектування коду на предметно-орієнтованих мовах програмування або інструментів для інших аспектів процесу розробки програмного забезпечення).

Інтегроване середовище розробки (Integrated Development Environment - IDE) Visual Studio пропонує ряд високорівневих функціональних можливостей, які виходять за рамки базового управління кодом. Нижче перераховані всі переваги:

- Вбудований веб-сервер
- Підтримка багатьох мов при розробці
- Менше коду для написання
- Інтуїтивний стиль кодування
- Більш висока швидкість розробки
- Можливості налагодження.[12]

Xamarin.Forms представляє платформу, яка націлена на створення кроссплатформенних додатків під Android, iOS і Windows 10. Навіщо використовувати саме цю платформу, які переваги вона несе? Є певні статистичні дані, що значна частина мобільних додатків створюється більш ніж для однієї платформи, наприклад, для Android і iOS. Однак неминуче розробники стикаються з наступними труднощами:

- відмінність в підходах побудова графічного інтерфейсу так чи інакше впливає на розробку. Розробники змушені підлаштовувати додаток під вимоги до інтерфейсу на конкретній платформі
- різні API - розбіжність у програмних інтерфейсів і реалізації тих чи інших функціональностей також вимагає від програміста облік цих специфічних особливостей
- різні платформи для розробки. Наприклад, щоб створювати додатки для iOS нам необхідна відповідне середовище - Mac OS X і ряд

спеціальних інструментів, типу XCode. А в якості мови програмування вибирається Objective-C або Swift. Для Androida ми можемо використовувати найрізноманітніший набір середовищ - Android Studio, Eclipse і т.д. Але тут для переважної більшості додатків застосовується Java або Kotlin.

А для створення додатків під Windows використовується Visual Studio, а в якості мов - C #, VB.NET, C++. Xamarin дозволяє створювати одну єдину логіку додатка із застосуванням C # і .NET відразу для всіх трьох платформ - Android, iOS, UWP.

Переваги використання Xamarin.Forms:

- В процесі розробки створюється єдиний код для всіх платформ
- Xamarin надає прямий доступ до нативним API кожної платформи
- При створенні додатків ми можемо використовувати платформу .NET і мова програмування C # (а також F #), який є досить продуктивним, і в той же час ясним і простим для освоєння і застосування
- Xamarin Forms підтримує кілька платформ. Основні платформи: Android, iOS, UWP, Tizen. Додаткові платформи в стані превью: MacOS, WPF, GTK #.

Функціонально платформа Xamarin представляє ряд субплатформ. Зокрема:

- Xamarin.Android - бібліотеки для створення додатків на ОС Android
- Xamarin.Mac - бібліотеки для створення додатків на Mac OS X
- Xamarin.iOS - бібліотеки для створення додатків для iOS

Ці субплатформи відіграють велику роль - через них додатки можуть напрявляти запити до прикладних інтерфейсів на пристроях під управлінням ОС Android або iOS.

Завдяки цим платформам ми можемо створювати окремо додатки для Android, окремо для iOS, але найбільш важливою особливістю платформи є можливість створювати кросплатформені додатки - тобто одна логіка для всіх

платформ. Тобто ми один раз можемо визначити візуальний інтерфейс, один раз до нього прив'язати якусь логіку на C #, і все це буде працювати на Android, iOS і Windows. Дана можливість представлена технологією Xamarin.Forms.[13]

Для розробки також було використано об’єктно-орієнтовану мову C#, яка має свої переваги та недоліки.

Переваги:

- підтримки корпорації Майкрософт. На відміну від Java, якої не пішов на користь перехід у власність Oracle, C # добре розвивається завдяки зусиллям Microsoft;
- останнім часом багато вдосконалюється. Так як C # був створений пізніше, ніж Java та іншими мовами, то потрібно дуже багато доопрацювати. Також це стосується популяризації і безкоштовності - було обіцяно відкрити вихідний код, а інструменти (Visual Studio, Xamarin) стали безкоштовними для приватних осіб і невеликих компаній;
- багато синтаксичного цукру. Синтаксичний цукор - це такі конструкції, які створені для полегшення написання і розуміння коду (особливо якщо це код іншого програміста) і не грають ролі при компіляції;
- середній поріг входження. Синтаксис схожий на C, C ++ або Java полегшує перехід для інших програмістів. Для новачків це також один з найперспективніших мов для вивчення;
- Xamarin. Завдяки покупці Xamarin на C # тепер можна писати під Android та iOS. Це, безсумнівно, великий плюс, так як їх власна мобільна ОС (Windows Phone) не завоювала велику популярність;
- додано функціональне програмування (F #);
- велика спільнота програмістів;
- багато вакансій на посаду C # програміста в будь-якому регіоні.

Недоліки

- орієнтованість, в основному, тільки на .NET (на Windows платформу);
- безкоштовність тільки для невеликих компанії, учнів і програмістів-одинаків. Для великих команд покупка ліцензій обійдеться недешево. Тому якщо у вас є своя фірма, то доведеться розщедритися;
- зберегли оператор go to.[14]

SQLite - це внутрішня бібліотека, яка реалізує автономну, безсерверну, нульову конфігурацію, транзакційний двигун бази даних SQL. Код для SQLite знаходиться у відкритому доступі і тому є вільним для використання для будь-яких цілей, комерційних чи приватних. SQLite - це найбільш широко розгорнута база даних у світі з більшою кількістю додатків, ніж ми можемо порахувати, включаючи декілька гучних проектів.

SQLite - це вбудований двигун бази даних SQL. На відміну від більшості інших баз даних SQL, у SQLite немає окремого серверного процесу. SQLite читає і записує безпосередньо у звичайні файли диска. Повна база даних SQL з кількома таблицями, індексами, тригерами та переглядами міститься в одному файлі диска. Формат файлу бази даних є кросплатформенним - ви можете вільно копіювати базу даних між 32-бітовою та 64-бітовою системами або між архітектурами big-endian та little-endian . Ці функції роблять SQLite популярним вибором як Формат файлу додатків. Файли баз даних SQLite - це рекомендований формат зберігання Бібліотекою Конгресу США. Не думайте про SQLite не як про заміну Oracle але замість fopen ()

SQLite - це компактна бібліотека. Якщо ввімкнено всі функції, розмір бібліотеки може бути менше 600 Кбіт, залежно від цільової платформи та налаштувань оптимізації компілятора. (64-розрядний код більший. А деякі оптимізації компілятора, такі як агресивна функція вбудовування функцій та розкручування циклу, можуть призвести до того, що об'єктний код буде значно більшим.) Існує компроміс між використанням пам'яті та швидкістю. Зазвичай SQLite працює швидше, чим більше пам'яті ви йому надаєте. Тим не

менш, продуктивність, як правило, досить хороша навіть в умовах низької пам’яті. Залежно від способу його використання, SQLite може бути швидшим, ніж прямий вхід / вивід файлової системи. SQLite дуже ретельно перевіряється перед кожним випуском і має репутацію дуже надійної. Більшість вихідних кодів SQLite присвячена виключно тестуванню та верифікації. Автоматизований набір тестів запускає мільйони та мільйони тестових випадків із залученням сотень мільйонів окремих SQL-висловлювань та досягає 100% охоплення галузевим тестом. SQLite витончено реагує на збої в розподілі пам'яті та помилки вводу/виводу диска. Операції є кислотними навіть якщо вони перервані збою системи або відключенням електроживлення. Все це підтверджено автоматизованими тестами за допомогою спеціальних тестових джгутів, що імітують збої в системі. Звичайно, навіть при всьому цьому тестуванні все ж є помилки. Але на відміну від деяких подібних проектів (особливо комерційних конкурентів), SQLite відкрито і чесно ставиться до всіх помилок і надає списки помилок та хронології змін коду щохвилини. [15]

3.2 Реалізація програмного забезпечення продуктивної організації часу в Visual Studio

Для розробки кросплатформених додатків на Xamarin нам потрібне середовище розробки. Для Windows таким середовищем є Visual Studio. При установці Visual Studio 2019 програмою для установника обов'язково треба вибрати пункт "Розробка мобільних додатків на .NET" як зображено на рисунку 3.1

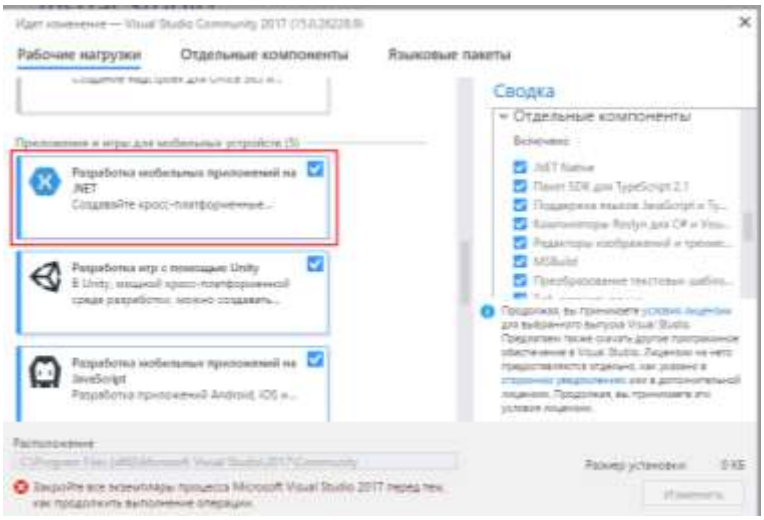

Рисунок 3.1 – Установка Visual Studio для розробки мобільних додатків

Після інсталяції ми зможемо в меню Help -> About Microsoft Visual Studio побачити позначку про Xamarin як на рисунку 3.2

Для початку необхідно створити проект, для створення кросплатформених додатків в Visual Studio 2019 є шаблон проекту Mobile App (Xamarin.Forms). Після цього необхідно обрати шаблон проекту і встановити ряд налаштувань. Нам доступно 4 шаблони:

- Blank App: порожній шаблон, який створює проект з мінімальною функціональністю
- Tabbed App: проект програми, яка використовує вкладки для навігації між сторінками
- Shell: шаблон односторінкового додатку
- Master Detail: проект спеціально для тих випадків, коли повинна бути функціональність для подання списку об'єктів і функціональність для виведення інформації по кожному окремому елементу списку. Крім того при створенні проекту є можливість обрати для яких платформ буде створюватись проект, зображено на рисунку 3.3

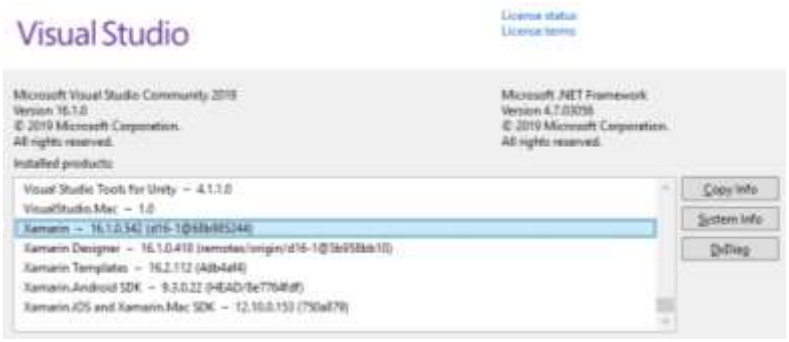

Рисунок 3.2 – Вікно About середовища Microsoft Visual Studio

Рисунок 3.3 – Вікно вибору шаблону проекту і налаштування підтримки платформ

Створений проект буде містити 4 проекти:

- HelloApp: головний проект бібліотеки, яка і буде містити всю основну логіку програми
- HelloApp.Android: проект для Android
- HelloApp.iOS: проект для iOS
- HelloApp.UWP: проект для Windows 10

Головним буде самий верхній проект (в даному випадку HelloApp). Він буде містити весь код і визначення інтерфейсу, яку потім будуть використовувати всі інші проекти.

Якщо ми звернемося до головного проекту, то в ньому вже буде чотири основних файли:

- App.xaml: файл, який визначає ресурси, загальні для всієї програми

- App.xaml.cs: файл з кодом C#, з якого починається виконання програми
- MainPage.xaml: файл з візуальним інтерфейсом для єдиної сторінки MainPage у вигляді xaml
- MainPage.xaml.cs: файл, який містить логіку MainPage на мові C #

Крім звичайних файлів для зберігання даних ми можемо використовувати локальні бази даних. Найбільш популярною системою баз даних для локальних додатків є SQLite.

Для роботи з SQLite можна використовувати різні підходи і бібліотеки. Рекомендованою бібліотекою є SQLite.NET, яка представляє просте ORM-рішення (Object Relational Mapping) для Xamarin. Вона дозволяє працювати з базою даних як зі сховищем об'єктів і маніпулювати даними як об'єктами стандартних класів C# без використання виразів на мові SQL. Спочатку додамо в головний проект у вузол References за допомогою менеджера NuGet цю бібліотеку sqlite-net-pcl, зображено на рисунку 3.4

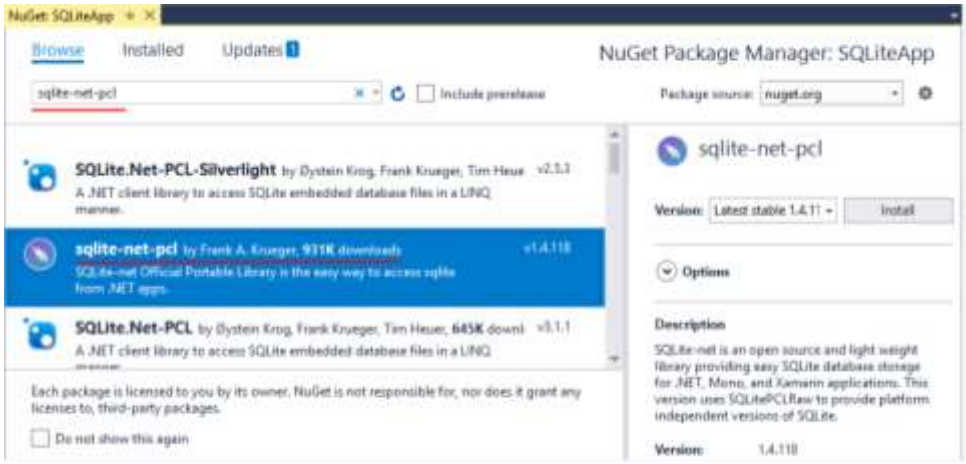

Рисунок 3.4 – Вікно додавання бібліотеки sqlite-net-pcl

Основні операції з SQLite.NET

Для початку визначимо клас, об’єкти якого будуть зберігатись в базі даних. Додамо в головний проект наступні класи:

- User
- Target

Ці класи використовують атрибути, які дозволяють налаштувати його відображення на таблицю в базі даних. Для налаштування ми використовуємо наступні атрибути:

- [PrimaryKey]: застосовується до властивості типу int і вказує, що стовпець в таблиці, який відповідає цій властивості, буде виконувати роль первинного ключа. Складові ключі не підтримуються
- [AutoIncrement]: застосовується до властивості типу int і вказує, що стовпець в таблиці, який відповідає цій властивості, буде інкрементувати значення на одиницю при додаванні нового елемента
- [Column (name)]: задає зіставлення властивості зі стовпцем в таблиці, який має ім'я name
- [Table (name)]: встановлює назву таблиці, яка буде відповідати даному класу
- [MaxLength (value)]: встановлює максимальну довжину для строкових властивостей
- [Ignore]: вказує, що властивість буде ігноруватися. Це може бути корисно, якщо значення даного властивості не треба зберігати в базі даних, і ця властивість не повинно зіставлятися за допомогою стовпців з таблиці в бд
- [Unique]: гарантує, що стовпець, який відповідає властивості з цим атрибутом, матиме унікальні неповторювані значення

При запитах до бази даних буде відбуватися зіставлення типів даних з SQLite з типами даних з C#. Зіставлення типів можна описати таблицею зображеною на рисунку 3.5. Приклад класу наведено нижче:

```
public class User
   {
      [PrimaryKey, AutoIncrement]
      public int Id { get; set; }
      public string userName { get; set; }
      public string name { get; set; }
```

```
[MaxLength(12)]
public string password { get; set; }
[MaxLength(10)]
public string phone { get; set; }
public User()       {
}    }
```

C#	SQLite
int, long	integer, bigint
bool	integer (1 = true)
enum	integer
float	real
double	real
decimal	real
string	varchar, text
DateTime	numeric, text
byte[]	blob

Рисунок 3.5 – Зіставлення типів даних SQLite і C#

Також необхідно створити клас репозиторію, через який будуть йти всі операції з даними. В конструкторі класу відбувається створення підключення і бази даних (якщо вона відсутня). Для всіх операцій з даними використовуються методи, визначені в класі SQLiteConnection:

- Insert: додає об'єкт в таблицю
- Get <T>: дозволяє отримати елемент типу T по id
- Table <T>: повертає всі об'єкти з таблиці
- Delete <T>: видаляє об'єкт по id
- Update <T>: оновлює об'єкт
- Query <T>: виконує SQL-вираз і повертає рядки з таблиці у вигляді об'єктів типу T (відноситься до виразів SELECT)

- Execute: виконує SQL-вираз, але нічого не повертає (відноситься до операцій. Де не треба повертати результат - UPDATE, INSERT, DELETE)

Фрагмент коду наведено нижче:

```
public class UserDB
   {
      private SQLiteConnection _SQLiteConnection;
      public UserDB()
      {
         var fileName = "UserDatabase.db3";
         var documentPath =
System.Environment.GetFolderPath(System.Environment.SpecialFolder.Personal);
         var path = Path.Combine(documentPath, fileName);
         _SQLiteConnection = new SQLiteConnection(path, false);
         _SQLiteConnection.CreateTable<User>();
      }
      public IEnumerable<User> GetUsers()
      {
         return (from u in _SQLiteConnection.Table<User>()
               select u).ToList();
      }
      public User GetSpecificUser(int id)
      {
         return _SQLiteConnection.Table<User>().FirstOrDefault(t => t.Id == id);
                  }
```

У Xamarin.Forms візуальний інтерфейс складається з сторінок. Сторінка являє собою об'єкт класу Page, вона займає весь простір екрану. Тобто те, що ми бачимо на екрані мобільного пристрою - це сторінка. Додаток може мати

одну або кілька сторінок. За замовчуванням весь інтерфейс створюється в класі App, який розташовується в файлі App.xaml.cs і який представляє поточний додаток.

Робота класу App починається з конструктора, де спочатку викликається метод InitializeComponent(), який виконує ініціалізацію об'єкта, а потім встановлюється властивість MainPage. Через це властивість клас App встановлює головну сторінку додатка. В даному випадку вона визначається класом HelloApp.MainPage, тобто тим класом, який визначений в файлах MainPage.xaml і MainPage.xaml.cs.

XAML представляє мову розмітки на основі xml для створення об'єктів декларативним чином. Власне тому при створенні проекту вже за замовчуванням в нього додаються два файли MainPage.xaml і MainPage.xaml.cs. Використання XAML має деякі переваги.

По-перше, за допомогою XAML ми можемо відокремити графічний інтерфейс від логіки додатку, завдяки чому над різними частинами додатка можуть одночасно працювати різні фахівці: над інтерфейсом - дизайнери, над кодом логіки - програмісти.

По-друге, XAML дозволяє описати інтерфейс більш ясним і зрозумілим способом, такий код набагато простіше підтримувати і оновлювати. В цілому XAML дозволяє організувати весь призначений для користувача інтерфейс у вигляді набору сторінок подібно до того, як це робиться в HTML.

Файл MainPage.xaml містить розмітку візуального інтерфейсу сторінки, що зображено на рисунку 3.6

Рисунок 3.6 – Вигляд файлу MainPage.xaml

Файл з розміткою на xaml є звичайний файл xml, і першим рядком йде стандартне визначення xml-файлу. Далі тут визначено елемент ContentPage, який представляє сторінку і всередині якого визначена мітка з текстом. У визначенні кореневого елемента ContentPage підключаються чотири простору імен за допомогою атрибутів xmlns. Простір імен http://xamarin.com/schemas/2014/forms визначає більшість типів з Xamarin Forms, які застосовуються для побудови графічного інтерфейсу. Другий простір імен http://schemas.microsoft.com/winfx/2009/xaml визначає ряд типів XAML і типи CLR.

Для забезпечення користувацького інтерфейсу було використано наступні елементи:

1. Button – є найбільш фундаментальним інтерактивним елементом управління в усіх Xamarin. Forms. Користувач натискає Button за допомогою пальця або клацає його за допомогою миші, щоб запустити цю команду. Button визначає подія Clicked, яка виникає при відпуску кнопки пальця або миші з поверхні Button. Властивість IsEnabled для Button повинно мати значення true, щоб воно відповідало на торкання. Кнопка успадковує або визначає кілька властивостей, що впливають на його зовнішній вигляд:

- TextColor - колір тексту Button
- BackgroundColor - колір фону для цього тексту
- BorderColor - це колір області, навколишнього Button

- FontFamily - це сімейство шрифтів, що використовується для тексту
- FontSize - це розмір тексту
- FontAttributes вказує, що текст виділений курсивом чи напівжирним шрифтом
- BorderWidth - ширина кордону
- CornerRadius є радіусом кута Button
- CharacterSpacing інтервал між символами Buttonого тексту

2. Label - використовується для відображення тексту як з одним, так і з кількома рядками. Мітки можуть мати оформлення тексту, кольоровий текст і використовувати призначені для користувача шрифти (сімейства, розміри і параметри).

- Оформлення тексту з підкресленням і закресленням можна застосовувати до екземплярів Label, присвоївши властивості Label.TextDecorations один або кілька TextDecorations членів перерахування: None, Underline і Strikethrough
- Мітки можна налаштувати для використання призначеного для користувача кольору тексту за допомогою властивості BIND TextColor.
- Мітки можуть бути налаштовані на обробку тексту, який не може поміститися в один рядок одним з декількох способів, що надаються властивістю LineBreakMode.
- Число рядків, що відображаються Label, можна вказати, задавши для властивості Label.MaxLines значення int.
- Мітки надають властивість FormattedText, яке дозволяє переглядати текст з декількома шрифтами і кольорами в одному і тому ж поданні.

3. TimePicker - викликає елемент управління вибору часу і дозволяє користувачеві вибрати час. TimePicker визначає наступні властивості:

- Time типу TimeSpan, певні години за замовчуванням дорівнює TimeSpan 0. Тип TimeSpan вказує тривалість часу з півночі.
- Format типу string, стандартної або користувальницької рядка форматування .NET, яка за замовчуванням має значення t, це короткий шаблон часу.
- TextColor типу Color, який використовується для показу обраного часу, за замовчуванням Color.Default.
- FontAttributes типу FontAttributes, значення за замовчуванням FontAtributes.None.
- FontFamily типу string, значення за замовчуванням null.
- FontSize типу double, значення за замовчуванням - 1,0.
- CharacterSpacing типу double - інтервал між символами TimePickerого тексту.

4. CheckBox - є типом кнопки, яка може бути або встановлена, або порожньою. Якщо прапорець встановлений, він вважається включеним. Якщо прапорець порожній, він вважається відключеним. CheckBox визначає властивість bool з ім'ям IsChecked, яке вказує, чи встановлений CheckBox. Це властивість також підтримується об'єктом BindableProperty, що означає, що його можна привласнити стилю і бути цільовим об'єктом прив'язок даних. CheckBox визначає подія CheckedChanged, яке виникає при зміні властивості IsChecked або за допомогою маніпуляції користувача, або коли додаток задає властивість IsChecked. Об'єкт CheckedChangedEventArgs, що супроводжує подія CheckedChanged, має одну властивість з ім'ям Value типу bool. При спрацьовуванні події в якості значення властивості Value встановлюється нове значення властивості IsChecked.

5. Switch - tлемент управління Xamarin. Switch Forms є горизонтальною вимикачем, за допомогою якого користувач може перемикатися між станами включення і виключення, які представлені boolean

значенням. Клас успадковує від View. Switch Елемент управління визначає дві властивості:

- IsToggled значення, яке вказує, чи включено в. Switch boolean
- OnColorпараметр, який впливає на те Switch, як об'єкт відображається в перемиканні або включеному стані. Color
- ThumbColor значення Color параметра бігунка Switched.

Ці властивості підтримуються BindableProperty об'єктом. Це означає, що можна використовувати стиль і ціль прив'язок даних. Елемент управління визначає подія, яка IsToggled виникає при зміні властивості за допомогою маніпуляції користувача або коли додаток задає властивість IsToggled Toggled Switch Об'єкт, Value який супроводжує подія, має одну властивість з ім'ям типу bool. Toggled ToggledEventArgs. При спрацьовуванні події значення Value властивості відображає нове значення IsToggled властивості.

6. Picker - відображає короткий список елементів, з якого користувач може вибрати елемент. Picker визначає наступні властивості:

- Title типу string, значення за замовчуванням null.
- TitleColor типу Color- колір, який використовується для виведення Title тексту.
- ItemsSource типу IList, вихідний список елементів для виведення, який за замовчуванням має значення null.
- SelectedIndex типу int, індекс обраного елемента, який за замовчуванням має значення-1.
- SelectedItem типу object обраний елемент, який за замовчуванням має значення null.
- TextColor типу Color, колір, використовуваний для виведення тексту, за замовчуванням - Color.Default.
- FontAttributes типу FontAttributes, значення за замовчуванням FontAtributes.None.
- FontFamily типу string, значення за замовчуванням null.

- FontSize типу double, значення за замовчуванням - 1,0.
- CharacterSpacing типу double - інтервал між символами елемента, що відображається Picker.

Для забезпечення роботи сповіщень було використано плагін Xam.Plugins.Notifier 3.0.1 - послідовний та простий спосіб показу локальних сповіщень у додатках Xamarin та Windows. Плагін потрібно завантажити з нугета, встановити, використання його відбувається через using Plugin.LocalNotifications; Необхідно викликати CrossLocalNotifications.Curerent з будь-якого проекту чи PCL, щоб отримати доступ до API.

- Показати місцеве сповіщення негайно

CrossLocalNotifications.Current.Show("title", "body");

- Показати місцеве сповіщення у заплановану дату / час

CrossLocalNotifications.Current.Show("title","body",101,DateTime.Now.AddSeconds(5));

- Скасувати місцеве сповіщення

CrossLocalNotifications.Current.Cancel(101); [16]

3.3 Тестування та аналіз результатів роботи програмного засобу продуктивної організації часу

Для початку роботи з додатком необхідно провести його установку на пристрій та запустити. При запуску додатку на мобільному пристрої відриється головне вікно, на якому необхідно буде авторизуватися, це вікно зображено на рисунку 3.7

Після авторизації відкриється вікно на якому будуть знаходитись меню програми. В меню Ви побачите наступні пункти: Сьогоднішні цілі, Плани на місяць, Статистика, Виконані завданні, Налаштування. Дане вікно має вигляд зображений на рисунку 3.8

Рисунок 3.7 – Головне вікно програми

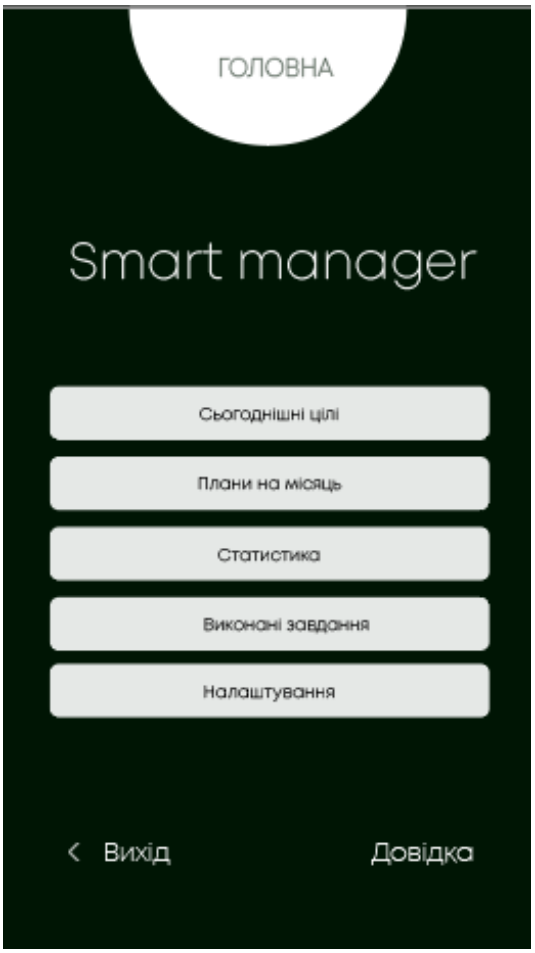

Рисунок 3.8 – Вікно з меню програми

Одним з головних елементів програми є вікно «Сьогоднішні цілі», яке зображено на рисунку 3.9, воно містить:

1. Навігацію по дням тижня
2. Списки справ розподілені за категоріями в порядку пріоритетності
3. Містить елементи управління справами
4. Містить кнопку додавання нової справи

Рисунок 3.9 -Вікно програми «Сьогоднішні цілі»

Для того, щоб додати нову справу є кнопка «+», після натиснення якої відкривається вікно «Нове завдання», зображене на рисунку 3.10

На етапі додавання нової справи є можливості:

- Здійснити опис завдання
- Встановити категорію до якої відноситься завдання
- Встановити пріоритетність завдання
- Встановити час сповіщення
- Увімкнути звук для сповіщень

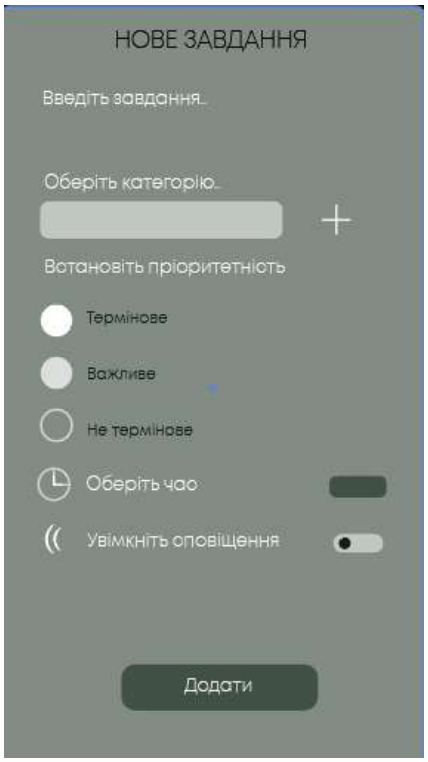

Рисунок 3.10 – Вікно «Нове завдання»

Окрім додавання справ, є також можливість редагування вже створених завдань. В головному меню також є пункт «Плани на місяць», при переході за даним пунктом відкривається вікно зображене на рисунку 3.11. У даному вікні є можливість переглядати завантаженість днів справами на місяць вперед.

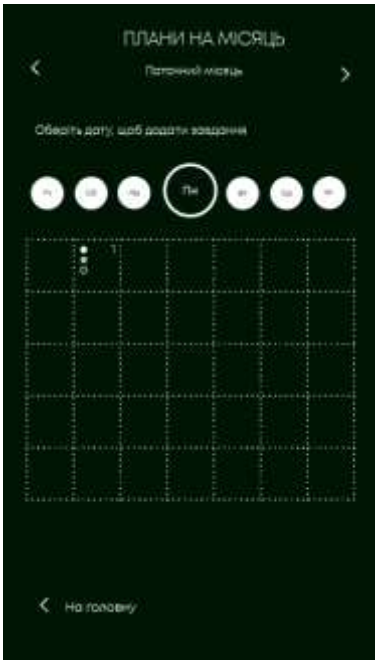

Рисунок 3.11 – Вікно «Плани на місяць»

Наступним пунктом головного меню є «Статистика», яка дозволяє переглянути які справи забирають найбільше часу, відповідно да категорій на які вони поділені. Дане вікно має наступний вигляд, зображений на рисунку 3.12. Наступний пункт меню «Виконані завдання» містить усі завдання, які були відмічені як завершені, що зображено на рисунку 3.13. Також є можливість відновлення завдань кошика.

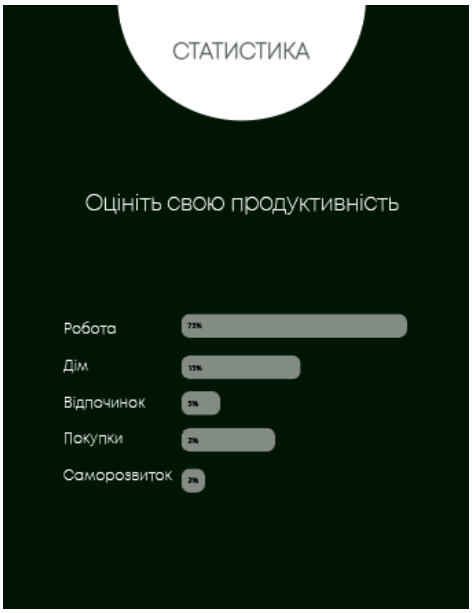

Рисунок 3.12 – Вікно «Статистика»

Рисунок 3.13 – Вікно «Виконані завдання»

Також є вікно «Налаштування», зображене на рисунку 3.14 в якому є можливість:

- Обрати мову додатку
- Встановити звук сповіщень
- Увімкнути відображення завдань на робочому столі

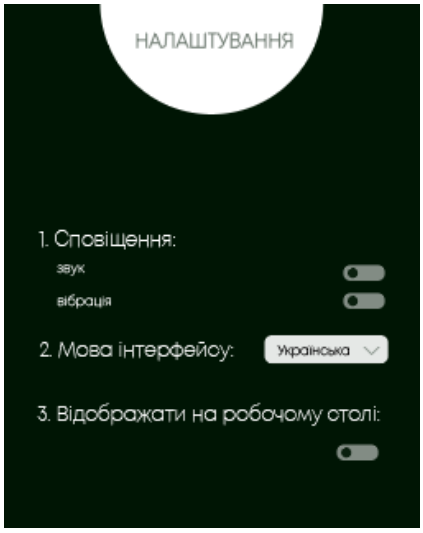

Рисунок 3.14 – Вікно «Налаштування»

В результаті розробки було досягнено поставленої мети за рахунок реалізації функцій, що відсутні в програмах аналогах, а саме:

- Наявність роботи з календарем
- Розподіл завдань за категоріями
- Резервне копіювання
- Пріоритизація завдань
- Фільтр завдань
- Зручний інтерфейс
- Статистика про завдання
- Користувацькі налаштування

Ці функції описані в таблиці 3.1

Таблиця 3.1 – Таблиця порівняння наявності функцій в програмах

	Наявність роботи з календарем	Розподіл за категоріями	Резервне копіювання	Пріоритизація завдань	Фільтр завдань
Розроблюваний продукт	+	+	+	+	+
Microsoft To Do	-	-	-	-	-
Мої Завдання: Планувальник завдань	+	-	-	+	+
Завдання: список справ, додаток списку завдань	-	+	-	-	-

За результатами таблиці можна побачити, що розроблений програмний засіб має покращений функціонал, принаймні на 2 можливості у порівнянні з програмами-аналогами, який забезпечить вищу продуктивність користувача.

3.4 Висновок

В третьому розділі було здійснено програмну реалізацію інформаційної технології продуктивної організації часу. Описано покрокову розробку додатку та засоби, які використовувалися для цього. Здійснено тестування програмного засобу, а також описано результат реалізації поставленої мети магістерської роботи, а саме підвищення продуктивності користувача за рахунок продуктивної організації часу з використанням інформаційної технології.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки. Для проведення технологічного аудиту було залучено 2-х незалежних експертів. Такими експертами будуть Озеранський В.С. та Яровий А.А.

Здійснюємо оцінювання комерційного потенціалу розробки за 12-ма критеріями за 5-ти бальною шкалою.

Результати оцінювання комерційного потенціалу розробки наведено в таблиці 4.1.

Таблиця 4.1 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта	
	1. Експерт 1	2. Експерт 2
	Бали, виставлені експертами:	
1	4	4
2	3	3
3	3	4
4	4	3
5	3	4
6	4	4
7	3	3
8	4	4
9	4	3
10	4	3
11	3	4
12	3	4
Сума балів	$СБ_1 = 43$	$СБ_2 = 43$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{2} = 43$	

Отже, з отриманих даних таблиці 4.1 видно, що нова розробка має високий рівень комерційного потенціалу.

4.2 Прогнозування витрат на виконання науково-дослідної роботи та конструкторсько–технологічної роботи.

Для розробки нового програмного продукту необхідні такі витрати.

Основна заробітна плата для розробників визначається за формулою (4.1):

$$З_о = \frac{М}{Т_р} \cdot t, \qquad (4.1)$$

де М- місячний посадовий оклад конкретного розробника;

$Т_р$ - кількість робочих днів у місяці, $Т_р = 22$ дні;

t - число днів роботи розробника, t = 50 днів.

Розрахунки заробітних плат для керівника і програміста наведені в таблиці 4.2.

Таблиця 4.2 – Розрахунки основної заробітної плати

Працівник	Оклад М, грн.	Оплата за робочий день, грн.	Число днів роботи, t	Витрати на оплату праці, грн.
Науковий керівник	6000	272,72	5	1363,5
Інженер-програміст	3400	154,54	50	7727
Всього:				9090,5

Розрахуємо додаткову заробітну плату:

$$З_{дод} = 0,1 \cdot 9090,5 = 909,05 \text{ (грн.)}$$

Нарахування на заробітну плату операторів НЗП розраховується як 37,5…40% від суми їхньої основної та додаткової заробітної плати:

$$Н_{зп}= (З_о + З_р)\cdot \frac{\beta}{100}, \quad (5.2)$$

$$Н_{зп}= (9090,5+909,05)\cdot \frac{36,3}{100} = 3629,83 \text{ (грн.)}.$$

Розрахунок амортизаційних витрат для програмного забезпечення виконується за такою формулою:

$$А=\frac{Ц\cdot Н_а}{100}\cdot \frac{Т}{12}, \quad (4.3)$$

де Ц – балансова вартість обладнання, грн;

$Н_а$ – річна норма амортизаційних відрахувань % (для програмного забезпечення 25%);

Т – Термін використання (Т=3 міс.).

Таблиця 4.3 – Розрахунок амортизаційних відрахувань

Найменування програмного забезпечення	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань,грн
Персональний комп’ютер	10000	25	3	625
Всього:				625

Розрахуємо витрати на комплектуючі. Витрати на комплектуючі розрахуємо за формулою:

$$K = \sum_1^n Н_і \cdot Ц_і \cdot К_і, \quad (4.4)$$

де n – кількість комплектуючих;

Ні - кількість комплектуючих і-го виду;

Ці – покупна ціна комплектуючих і-го виду, грн;

Кі – коефіцієнт транспортних витрат (приймемо Кі = 1,1).

Таблиця 4.4 - Витрати на комплектуючі, що були використані для розробки ПЗ.

Найменування матеріалу	Одиниці виміру	Ціна, грн.	Витрачено	Вартість витрачених матеріалів, грн.
Флешка	шт.	180	1	180
Пачка паперу	уп.	130	1	130
Ручка	шт.	10	1	10
Всього з урахуванням транспортних витрат				352

Витрати на силову електроенергію розраховуються за формулою:

$$В_е=В\cdot П\cdot Ф\cdot К_п ; \quad (4.5)$$

де В – вартість 1кВт-години електроенергії (В=1,7 грн/кВт);

П – установлена потужність комп'ютера (П=0,6кВт);

Ф – фактична кількість годин роботи комп'ютера (Ф=200 год.);

$К_п$ – коефіцієнт використання потужності ($К_п< 1$, $К_п = 0,7$).

$$В_е= 1,7 \cdot 0,6 \cdot 200 \cdot 0,7 = 142,8 \text{ (грн.)}$$

Розрахуємо інші витрати $В_{ін}$.

Інші витрати $І_в$ можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які були виконували дану роботу, тобто:

$$В_{ін} = (\mathbf{1}..\mathbf{3}) \cdot (З_о + З_р). \quad (4.6)$$

Отже, розрахуємо інші витрати:

$$В_{ін} = \mathbf{1} * (\mathbf{9090,5} + \mathbf{909,05}) = \mathbf{9999,55} \text{ (грн)}.$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини роботи:

$$В = З_о + З_д + Н_{зп} + А + К + В_е + І_в$$

В = 9090,5+909,05+3629,83+625+352+142,8+9999,55= 24748,73 (грн.)

Розрахуємо загальну вартість наукової роботи $B_{заг}$ за формулою:

$$В_{заг} = \frac{В_{ін}}{\boldsymbol{\alpha}} \quad (4.7)$$

Де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відн. одиницях = 1.

$$\mathbf{В_{заг}} = \frac{\mathbf{24748,73}}{\mathbf{1}} = \mathbf{24748,73}$$

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою:

$$ЗВ = \frac{В_{заг}}{\beta} \quad (4.8)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Отже, розрахуємо загальні витрати:

$$\mathbf{ЗВ = \frac{24748{,}73}{0{,}9} = 27498{,}58} \text{ (грн.)}$$

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки.

Спрогнозуємо отримання прибутку від реалізації результатів нашої розробки. Зростання чистого прибутку можна оцінити у теперішній вартості грошей. Це забезпечить підприємству (організації) надходження додаткових коштів, які дозволять покращити фінансові результати діяльності .

Оцінка зростання чистого прибутку підприємства від впровадження результатів наукової розробки. У цьому випадку збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою:

$$\mathbf{\Delta П_i = \sum_1^n (\Delta П_я \cdot N + П_я \Delta N)_i} \quad (4.9)$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження результатів розробки у даному році;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_я$ – основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

В результаті впровадження результатів наукової розробки витрати на виготовлення інформаційної технології зменшаться на 25 грн (що автоматично спричинить збільшення чистого прибутку підприємства на 25 грн), а кількість користувачів, які будуть користуватись збільшиться: протягом першого року – на 150 користувачів, протягом другого року – на 125 користувачів, протягом третього року – 100 користувачів. Реалізація інформаційної технології до впровадження результатів наукової розробки складала 500 користувачів, а прибуток, що отримував розробник до впровадження результатів наукової розробки – 400 грн.

Спрогнозуємо збільшення чистого прибутку від впровадження результатів наукової розробки у кожному році відносно базового.

Отже, збільшення чистого продукту $\Delta П_1$ протягом першого року складатиме:

$$\Delta П_1 = 25 \cdot 500 + (400 + 25) \cdot 150 = 76250 \text{ грн.}$$

Протягом другого року:

$$\Delta П_2 = 25 \cdot 500 + (400 + 25) \cdot (150 + 125) = 129375 \text{ грн.}$$

Протягом третього року:

$$\Delta П_3 = 25 \cdot 500 + (400 + 25) \cdot (150 + 125 + 100) = 171875 \text{ грн.}$$

4.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Визначимо абсолютну і відносну ефективність вкладених інвестором інвестицій та розрахуємо термін окупності.

Абсолютна ефективність $Е_{абс}$вкладених інвестицій розраховується за формулою:

$$Е_{абс} = (ПП - \mathbf{PV}), \qquad (4.10)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, 3 роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки 2, 3,4.

Рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, рисунок 4.1.

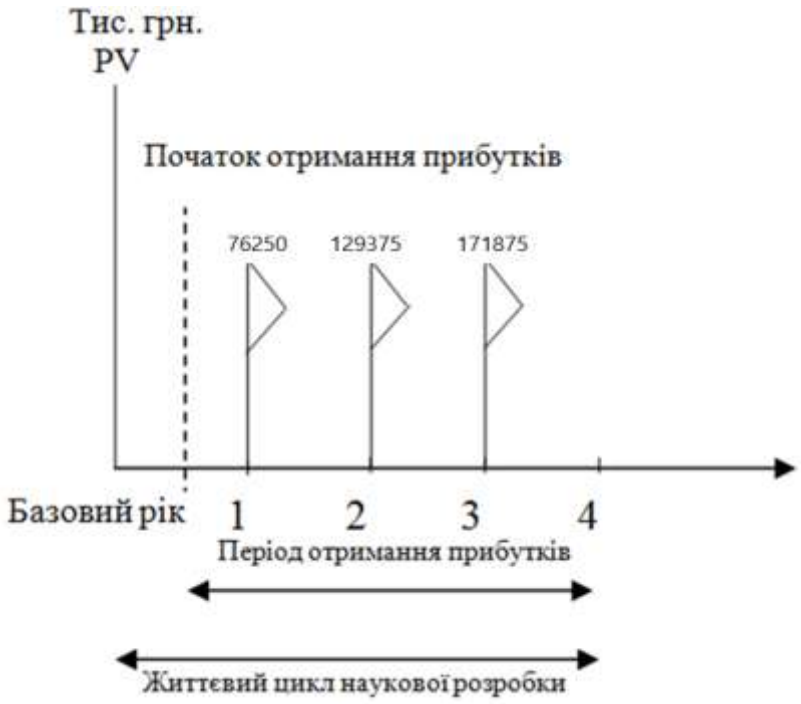

Рисунок 4.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Розрахуємо вартість чистих прибутків за формулою:

$$\mathbf{ПП} = \sum_{\mathbf{1}}^{\boldsymbol{m}} \frac{\boldsymbol{\Delta П_i}}{\boldsymbol{(1+\tau)^t}} \quad (4.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки.

Отже, розрахуємо вартість чистого прибутку:

$$ПП = \frac{\mathbf{27498{,}58}}{\mathbf{(1+0{,}1)^0}} + \frac{\mathbf{76250}}{\mathbf{(1+0{,}1)^2}} + \frac{\mathbf{129375}}{\mathbf{(1}+0{,}1)^3} + \frac{171875}{(1+0{,}1)^4} = 305109{,}38 \text{ (грн.)}$$

Тоді розрахуємо $E_{\text{абс}}$:

$$\text{Е}_{\text{абс}} = 305109{,}38 - 27498{,}58 = 277610{,}8 \text{ грн.}$$

Оскільки $E_{\text{абс}} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР буде доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $\text{Е}_\text{в}$ за формулою:

$$\text{Е}_\text{в} = \sqrt[\text{Т}]{1 + \frac{\text{Е}_{\text{абс}}}{\text{PV}}} - 1 \quad (4.12)$$

де $E_{абс}$ – абсолютна ефективність вкладених інвестицій, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$T_ж$ – життєвий цикл наукової розробки, роки.

Тоді будемо мати:

$$E_в = \sqrt[3]{1 + \frac{277610,8}{27498,58}} - 1 = 1,23 \text{ або } 123\ \%$$

Далі, розраховану величина $E_в$ порівнюємо з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau = d + f,$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2019 році в Україні d = 0,2;

f – показник, що характеризує ризикованість вкладень, величина f = 0,1.

$$\tau = 0,2 + 0,1 = 0,3$$

Оскільки $E_в$ = 123% > $\tau_{мін}$ = 0,3 = 30%, то у інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

Термін окупності вкладених у реалізацію наукового проекту інвестицій. Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в}$$

$$T_{ок} = \frac{1}{1,23} = 0,81 \text{ року}$$

4.5 Висновок

В результаті виконання економічної частини було проведено оцінювання комерційного потенціалу розробки, прогнозування витрат на виконання науково-дослідної роботи та конструкторсько–технологічної роботи, прогнозування комерційних ефектів від реалізації результатів розробки та розрахунок ефективності вкладених інвестицій та період їх окупності.

Оскільки $E_{\text{абс}} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР буде доцільним.

Обрахувавши термін окупності даної наукової розробки, який дорівнює 0,81 року, можна зробити висновок, що фінансування даної наукової розробки буде доцільним.

ВИСНОВКИ

В ході виконання магістерської кваліфікаційної роботи було розроблено інформаційну технологію продуктивної організації часу. Було виконано наступні завдання: проведено аналіз предметної області продуктивної організації, розроблено інформаційну технологію, здійснено програмну реалізацію, а також обґрунтовано економічну доцільність розробки даного програмного продукту.

В першому розділі магістерської кваліфікаційної роботи було проведено аналіз існуючих програмних засобів продуктивної організації часу, здійснено постановку задачі. В результаті було виявлено основні переваги та недоліки існуючих програм і визначено функціонал який має бути наявний в розроблюваному продукті.

В другому розділі обґрунтували вибір методі організації часу користувача на основі аналізу відомих систем тайм-менеджменту. Розроблено алгоритм продуктивної організації часу користувача.

В третьому розділі описали програмну реалізацію інформаційної технології, а саме особливості створення інтерфейсу, опис технологій та методів, що використовувались для створення. Для розробки було використано середовище Visual Studio, інструменти Xamarin Forms, реляційна СУБД SQLite і мова програмування C#. Розроблена програма має принаймні на 2 функціональні можливості більше ніж програми аналоги.

В четвертому розділі було обґрунтовано економічну доцільність розробки інформаційної технології продуктивної організації часу, а саме: оцінювання комерційного потенціалу розробки, прогнозування витрат на виконання науково-дослідних робіт, прогнозування комерційних ефектів від реалізації розробки, розрахунок ефективності вкладених інвестицій та період їх окупності. Обрахувавши термін окупності даної наукової розробки, який дорівнює 0,81 року, можна зробити висновок, що фінансування даної наукової

розробки буде доцільним. Отже всі завдання, які ставилися на магістерську кваліфікаційну роботу виконані, мета магістерського дослідження досягнута.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Волянська Є. В. / Інформаційна технологія продуктивної організації часу / Інформаційні технології і автоматизація–2019 : зб. доп. XII Міжнар. наук.-практ. конф., Одеса, 17–18 жовт. 2019 р. / Одес. нац. акад. харч. технологій, Ін-т комп'ютер. систем і технологій «Індустрія 4.0» ім. П. М. Платонова ; ред. кол.: С. В. Котлик, В. А. Хобін, В. М. Плотніков. – Одеса, 2019. – Ч. І. – с. 72. : – Бібліогр.: 3 назв.

2. Тайм-менеджмент – технологія підвищення ефективності використання часу - [Електронний ресурс]. – Режим доступу: http://market.avianua.com/?p=4069

3. Планирование дел: составление списков и приоритизация - [Електронний ресурс]. – Режим доступу: https://4brain.ru/time/plan.php

4. Матрица управления временем С. Кови - [Електронний ресурс]. – Режим доступу: https://4brain.ru/time/systems.php#3

5. Системы тайм менеджмента - [Електронний ресурс]. – Режим доступу: http://jeejoo.com/sistemy-tajm-menedzhmenta-vybirajte-chto-vam-po-dushe/

6. Microsoft To Do: список, завдання й нагадування - [Електронний ресурс]. – Режим доступу: https://play.google.com/store/apps/details?id=com.microsoft.todos

7. Ike - To-Do List, Task List - [Електронний ресурс]. – Режим доступу: https://play.google.com/store/apps/details?id=com.pocketuniverse.ike&hl=ru

8. Мої Завдання: Планувальник завдань - [Електронний ресурс]. – Режим доступу: https://play.google.com/store/apps/details?id=com.weekly.app

9. Завдання: список справ, додаток списку завдань - [Електронний ресурс]. – Режим доступу:

https://play.google.com/store/apps/details?id=com.tasks.android&showAllReviews=true

10. Теорія і практика UML. діаграма послідовності - [Електронний ресурс]. – Режим доступу: http://it-gost.ru/articles/view_articles/94

11. Теорія і практика UML. діаграма діяльності - [Електронний ресурс]. – Режим доступу: http://it-gost.ru/articles/view_articles/96

12. Описание среды разработки Microsoft Visual Studio - [Електронний ресурс]. – Режим доступу: https://studbooks.net/2258619/informatika/opisanie_sredy_razrabotki_microsoft_visual_studio

13. Xamarin и кросс-платформенная разработка - [Електронний ресурс]. – Режим доступу: https://metanit.com/sharp/xamarin/1.1.php

14. Язык программирования C# - [Електронний ресурс]. – Режим доступу: https://learn-code.ru/yazyki-programmirovaniya/c-sharp

15. About SQLite - [Електронний ресурс]. – Режим доступу: https://www.sqlite.org/about.html

16. Local Notifications Plugin for Xamarin and Windows - [Електронний ресурс]. – Режим