Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____
(повна назва)

Кафедра _____Програмної інженерії_____
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

_____другий (магістерський)_____
(рівень вищої освіти)

Дослідження методів оптимізації систем керування робочим процесом
(тема)

Виконав: студент 2 курсу, групи ПЗСм-17-3
Спеціальності 121- Інженерія програмного забезпечення
(код і повна назва спеціальності)

Освітньо-професійної програми
Програмне забезпечення систем
(повна назва освітньої програми)

_____Болотов Д.В._____
(прізвище, ініціали)

Керівник _____проф. Бондарєв В.М._____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри, проф. _____ З.В.Дудар

2019 р.

Харківський національний університет радіоелектроніки

Факультет _Комп'ютерних наук_____

Кафедра _Програмної інженерії_____

Рівень вищої освіти _другий (магістерський)_____

Спеціальність _121-Інженерія програмного забезпечення_____
(код і повна назва)

Освітньо-професійна програма _Програмне забезпечення систем_____
(повна назва)

ЗАТВЕРДЖУЮ:

Зав.кафедри_____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові _Болотову Дмитру Володимировичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів оптимізації систем керування робочим процесом

затверджена наказом по університету від "___" _____20 ___ р № _____
заповнюється вручну після отримання наказу

2. Термін подання студентом роботи до екзаменаційної комісії
26 червня 2019 р.

3. Вихідні дані до роботи *методи керування проектами, процеси керування проектами, трикутник управління проектами, процеси в управлінні часом, методи оптимізації управління часом, матриця Ейзенхауера, метод Помадоро*

4. Перелік питань, що потрібно опрацювати в роботі *мета роботи, аналіз проблемної галузі і постановка задачі, огляд методів керування робочим процесом, дослідження процесів в управлінні часом, аналіз таких методів як оптимізації як матриця Ейзенхауера та метод Помадоро, архітектура та проектування програмного забезпечення для оптимізації робочого процесу, аналіз можливих застосувань, висновки*

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) *актуальність дослідження, керування проектами, трикутник управління проектами, управління часом, методи в управлінні часом, мета роботи, постановка задачі, аналіз аналогів, програмна реалізація, сфери застосування, можливі удосконалення, висновки*

Зворотний бік бланку завдання

6 Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта	
		підпис	дат
Снецчастина	проф. Бондарєв В.М.		

КАЛЕНДАРНИЙ ПЛАН

	Назва етапів роботи	Терміни виконання етапів роботи	Примітка*
1.	Аналіз предметної галузі	19 квітня 2019р.	
2.	Огляд існуючих методів керування проектами	27 квітня 2019р.	
3.	Процеси в керуванні часом		
4.	Підготовка пояснювальної записки	25 травня 2019р.	
5.	Спецчастина	26 травня 2019р.	
6.	Підготовка презентації та доповіді	28 травня 2019р.	
7.	Попередній захист	30 травня 2019р.	
8.	Нормоконтроль, рецензування	02 червня 2019р.	
9.	Занесення диплома в електронний архів	03 червня 2019р.	
10.	Допуск до захисту у зав. кафедри	25 червня 2019р.	
* заповнюється вручну після виконання чергового пункту			

Дата видачі завдання _____ _____2019 р.

Студент _____
(підпис)

Керівник роботи _____ __проф. Бондарєв В.М._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до атестаційної роботи: 81 с., 5 р., 18 рис., 2 табл., 3 дод., 20 джерел.

ДОСЛІДЖЕННЯ, ЕФЕКТИВНІСТЬ РОБОЧОГО ЧАСУ, КЕРУВАННЯ ПРОЕКТАМИ, МЕТОДИ ОПТИМІЗАЦІЇ, ПЛАНУВАННЯ, РОБОЧИЙ ПРОЦЕС.

Об’єктом дослідження атестаційної роботи є методи та практики оптимізації систем керування робочим процесом.

Метою дослідження є виявлення найважливіших аспектів у процесах керування проектами та аналіз існуючих рішень для оптимізації таких процесів.

Результатом досліджень є створення системи, що допоможе оптимізувати процеси у керуванні проектами застосовуючи вже існуючи практики та підходи.

Така система може бути застосована як інструмент для менеджерів проекту у сфері розробки програмного забезпечення.

RESEARCH, WORKING TIME EFFECTIVENESS, PROJECT MANAGEMENT, OPTIMIZATION METHODS, PLANNING, WORKING PROCESS.

The object of the examination of attestation work is the methods and practices of optimizing working management systems.

The purpose of the research is to identify the most important aspects of project management processes and to analyze existing solutions to optimize such processes.

The result of the research is the creation of a system that will help to optimize processes in project management by applying existing practices and approaches.

Such a system can be used as a tool for project managers in the field of software development.

ЗМІСТ

ВСТУП

Розробка програмного забезпечення на даний момент є однією з найприбутковіших та швидкозростаючих сфер діяльності. Сотні тисяч робочих місць та десятки тисяч проектів що спрямовані на розвиток майже кожної галузі .

Для того щоб мати контроль та змогу керувати такими ресурсами необхідно чітко розуміти процеси, що є основою для отримання бажаного результату. Адже за відсутності чітко виражених правил, послідовності дій, стратегій та алгоритмів неможливо вирішити усі проблеми на шляху від проекту до готового програмного продукту.

Від чого необхідно відштовхуватись щоб мати змогу розвивати сферу програмного забезпечення у вірному напрямку? Насамперед від потреб ринку (бізнесу) та проблем що виникають на протязі кожної зі стадій проекту, а саме планування, реалізація, підтримка.

Метою атестаційної роботи є дослідження та аналіз процесів у керуванні проектами. Виявлення які існують практики та методики організації процесів у керуванні проектами, що між ними спільного та яка між ними різниця, які фігурують поняття у процесі керування проектами, та які області є першочерговими для впровадження нововведень.

Задачі які необхідно вирішити для досягнення поставленої мети полягають у виявленні критичних місць в управлінні проектами [1], описанні доступних ресурсів, взаємозв’язки між складовими процесів у керуванні проектами .

Об’єктом у даному випадку є саме процеси що лежать в основі керування проектами а предметом є методи що спрямовані на оптимізацію систем створених для керування робочим процесом.

Методи дослідження базуються на аналізі вже існуючих рішень та виявленні критичних місць у практиках, підходах, інструментах.

Незалежно від предметної області та розмірів проекту фігурує два поняття – це бюджет та часові обмеження. На даний момент існує ряд систем та інструментів

що спрямовані на оптимізацію процесів керування робочим процесом. Частина з них зосередженні на управлінні командою, інша на можливість контролю проекту на кожній з його стадій. І в тому і в іншому випадку фігурує поняття часу, як дата завершення проекту, чи оцінка спеціалістом часу необхідного на виконання задачі.

Проаналізувавши предметну галузь було прийнято рішення сфокусуватись на такому понятті як управління часом. Адже саме час є тим фактором що впливає на фінансову успішність чи провал проекту.

У результаті проведення досліджень в сфері управління часом було обрано такі методи як матриця Ейзенхауера та метод Помадоро, головною задачею яких є оптимізація у процесах планування, виконання задач, оцінки задач.

Визначивши методи, що спрямовані на удосконалення вже існуючих процесів було відтворено програмну реалізацію кожного з них та об’єднано у додаток. Така система може бути застосована як інструмент для менеджерів проекту та команди у сфері розробки програмного забезпечення, використовуючи який у довгостроковій перспективі можна підвищити рівень точності в оцінках задач, оптимізувати процес планування.

Таким чином з економічної точки зору зменшиться ризик для проектів насамперед з фіксованим бюджетом. Зважаючи на те що розроблений продукт включає лише ряд існуючих практик та методів для оптимізації є доцільним продовження досліджень та розширення системи реалізаціями інших методів оптимізації систем керування робочим процесом, інтеграція з існуючими сервісами спрямованими на оптимізацію керування робочим процесом але сфокусованих на інших аспектах цієї діяльності.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Керування проектами

Процеси є невід’ємною частиною керування проектами та загалом будь-якої діяльності. Адже саме процеси допомагають упорядкувати, виділити головне та сформувати послідовність дій для вирішення тієї чи іншої задачі.

Для того щоб зрозуміти проблематику досліджуваної предметної галузі необхідно чітко усвідомлювати потреби сфери та кінцеву ціль керування проектами. Щоб мати можливість виділити фактори що впливають на успіх проекту необхідно виділити основні поняття та компоненти системи. Для початку необхідно ввести таке поняття як проект.

Проект — це процес обмежений часовими рамками, зазвичай обмежений датою, який має визначений початок та кінець, але також може обмежуватися фінансуванням або досягненням результатів [2]. Мета здійснення такого процесу спрямована на реалізацію унікальних цілей та завдань.

Маючи уяву що являє собою проект, можна ввести наступне поняття, що буде окреслювати саме керування відносно проекту.

Управління проектами – область знань з планування, організації та керування ресурсами з метою успішного досягнення цілей та завершення завдань проекту.

Тимчасова природа проектів контрастує з бізнесом (процесами) які є повторюваною, постійною або частково постійною діяльністю з виробництва продуктів або послуг. На практиці, управління вищезазначеними двома системами часто різниться і таким чином вимагає розвитку окремих технічних навичок та використання розподіленого управління ними.

Тепер, коли поняття проект та управління проектами є досить чіткими та зрозумілими, виникає питання, хто саме відповідає за керування проектами та налагодження процесів для цієї діяльності?

Менеджер проекту — це спеціаліст у сфері керування проектами. Менеджер проекту може бути відповідальним за планування, супровід, виконання та

завершення будь-якого проекту. Це особа відповідальна за виконання визначених завдань проекту. Головними завданнями менеджера проектів є визначення чітких та об’єктивних завдань проекту, визначення вимог до проекту та управління головними обмеженнями проекту: часом, вартістю, межами та обсягом завдань.

Менеджер проекту деколи є представником замовника і має визначити та сформулювати чіткі потреби й вимоги замовника, базуючись на знаннях про організацію, яку він представляє. Вміння використовувати внутрішні процедури підрядника та створювати тісні зв'язки з призначеними представниками замовника є життєвою необхідністю для забезпечення виконання ключових завдань щодо ціни, часу реалізації, якості та задоволення вимог замовника.

Головною метою проектного управління є досягнення всіх цілей та виконання завдань та задач проекту, у той же час виконуючи зобов'язання щодо визначених обмежень проекту [3]. Типовими обмеженнями для проекту є межі та зміст, час, бюджет. Другорядним завданням є оптимізація, розподілення та інтеграція завдань, необхідних для досягнення наперед визначених цілей.

На даний момент враховуючи головні цілі керування проектом, є можливість провести декомпозицію щодо поняття проект, та виділити головні компоненти що є складовою частиною процесів у керуванні проектами.

Отже, враховуючи обмеження, цілі, ресурси можна зробити наступні висновки.

Основуючись на ідеї проекту формуються вимоги до цього проекту. Вимоги включають у себе як функціональні так і бізнес-вимоги. По суті це є поверховий опис кінцевого продукту.

Далі, вимоги розбиваються на задачі, що в кінцевому рахунку є детальним описом окремо взятої частини функціоналу.

Як результат задачі розподіляються на команду, і за кожну задачу відповідає конкретна людина (чи група людей, в залежності від дій що необхідно зробити для завершення задачі, наприклад розробити, протестувати).

І результатом такої роботи є готовий продукт. Послідовність дій що необхідні для досягнення цілей проекту, враховуючи доступні ресурси зображено на рисунку 1.1.

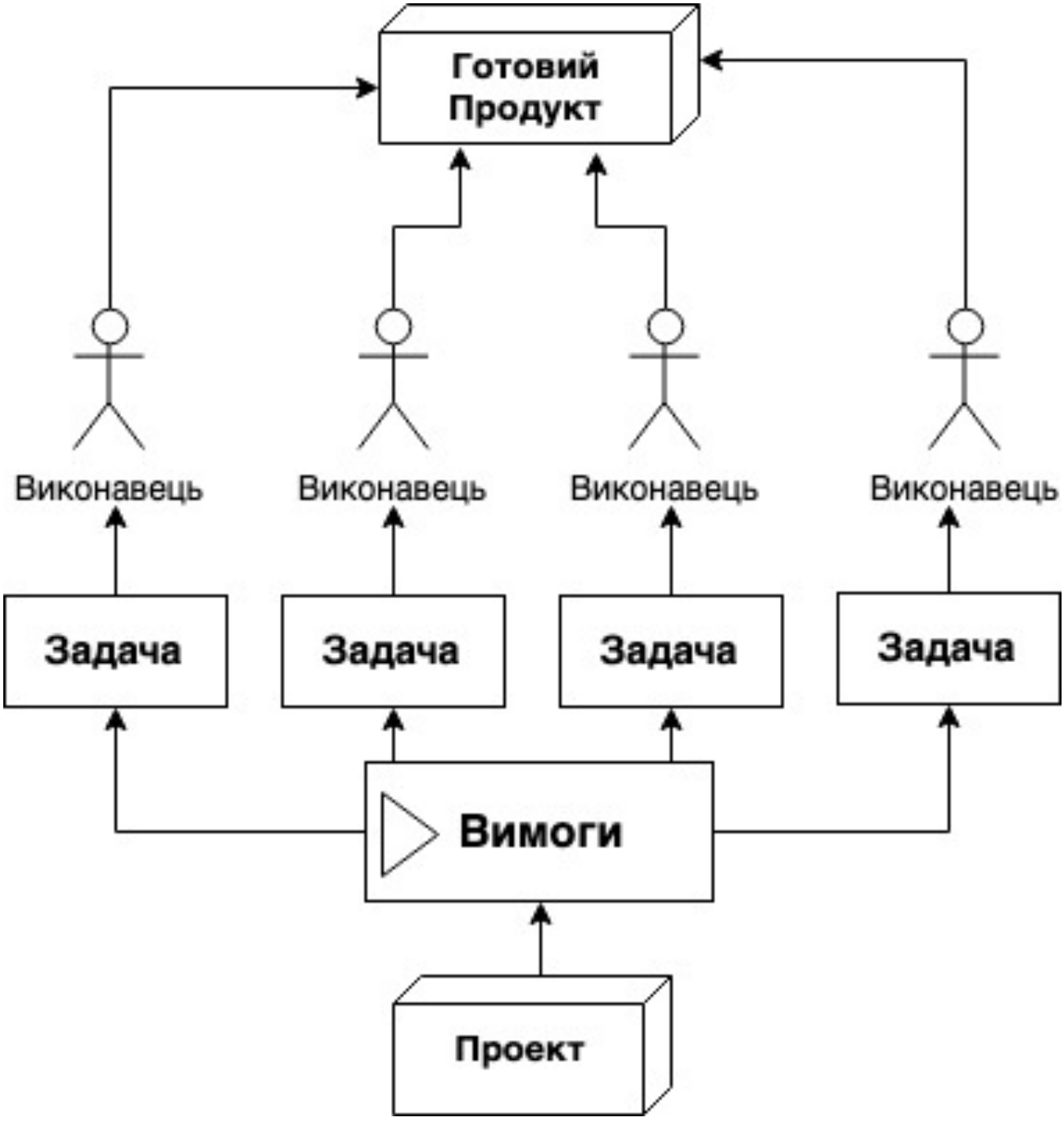

Рисунок 1.1 – Декомпозиція поняття проекту

Тепер, коли виявлено основні компоненти у процесі керування проектами та зв’язки між ними, необхідно з’ясувати які підходити та практики існують для

керування такими ресурсами, провести порівняльну характеристику, та виявити область в керуванні проектами яка є оптимальною для пошуку методів оптимізації.

1.2 Дослідження методів керування проектами

Існує ряд методів управління проектними ресурсами та процесами, включаючи гнучкі ітеративні, послідовні та методи розподілу на етапи.

Незважаючи на метод, що використовується для керування робочим процесом, необхідно детально розглядати загальні цілі проекту, календарний план, вартість (витрати), одночасно з ролями та відповідальністю усіх виконавців та зацікавлених сторін.

Розглянемо більш детально існуючі методи керування проектами, та опираючись на ці дослідження виявимо ряд проблем що існують у сфері керування проектами та рішень що є основою у цих методах.

Першим у списку буде традиційний метод поділу на етапи. Цей метод керування робочим процесом передбачає визначення послідовності дій, що мають бути завершені для досягнення цілей проекту.

В розробці програмного забезпечення цей підхід відомий під назвою модель водоспаду, через те що кожен наступний етап завершується лише після закінчення попереднього, тобто етапи мають лінійну залежність. Також для моделі водоспаду можна використовувати іншу назву послідовна модель.

В послідовній моделі керування робочим процесом можна визначити 5 складових проекту таких як 4 етапи та контроль розвитку проекту:

- ініціювання;
- планування та розробка;
- виконання та впровадження;
- моніторинг та контроль;
- завершення.

На рисунку 1.2 зображено етапи у послідовній моделі керування проектами та взаємозв’язки між ними.

Рисунок 1.2 – Етапи у традиційному методі

Не всі проекти проходять кожен з етапів, така ситуація може бути у разі дострокового припинення проекту. Деякі проекти не мають етапів структурованого планування або моніторингу. Деякі з проектів проходять стадії 2, 3, 4 по декілька разів.

З метою адаптації послідовної моделі для сфери розробки програмного забезпечення багато організацій використовують методологію Раціональних уніфікованих процесів (англ. Rational Unified Process RUP).

Використання послідовної моделі управління проектами ефективне для невеликих, чітко визначених проектів, але для проектів більш великих, невизначених та нових, описана модель у більшості випадків призводить до негативних результатів [4].

Конус невизначеності пояснює таке явище тим, що планування, яке виконується на початкових етапах проекту не є ефективним через значний ступінь невизначеності з точки зори вимог. Це особливо актуально для розробки нових продуктів у сфері програмного забезпечення, оскільки протягом виконання роботи з’ясовується багато не врахованих до цього факторів.

Наступний метод керування проектами є критичний шлях управління проектом.

Критичний шлях управління проектом (англ. Critical Chain Project Management CCPM) — це метод планування та управління проектами, який на перше місце ставить управління ресурсами (фізичними та людськими), необхідними для виконання завдань проекту. Фактично цей метод є доповненням теорії обмежень для проектів. Основним завданням цього методу є зріст продуктивності (або збільшення відсотку завершених завдань). На рисунку 1.3 представлено приклад графу критичного шляху.

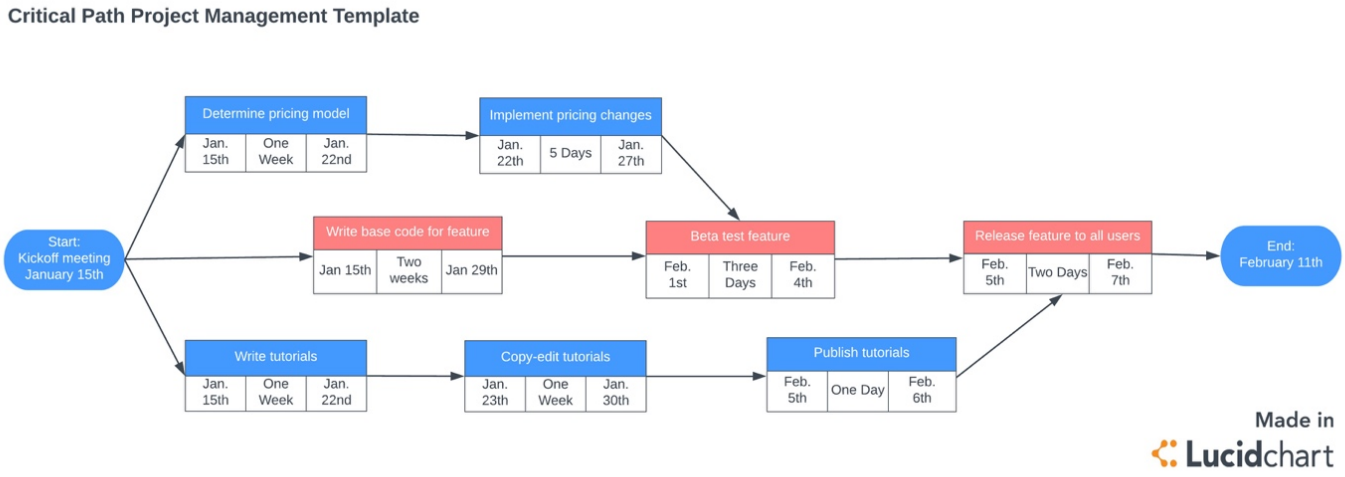

Рисунок 1.3 – Приклад критичного шляху

Системні обмеження для проектів зазначаються як ресурси. Для того щоб застосовувати обмеження, задачам на критичному шляху задається пріоритет вище ніж іншим завданням. У цілому, проекти плануються та керуються таким чином, щоб ресурси були доступні, коли задачі критичного шляху переходять у активну фазу, підпорядковуючи усі інші ресурси задачам критичного шляху.

Незалежно від типу проекту, план проекту має задавати розподілення ресурсів на рівні. Найдовша послідовність ресурсно-обмежених задача має бути відмічена, як критичний шлях. В середовищах, де існує не один проект а декілька, розподілення ресурсів на рівні необхідно використовувати в усіх проектах.

Загалом, достатньо обрати один наскрізний ресурс, такий ресурс що виступає як обмеження в усіх проектах та послідовно розташувати проекти відповідно до доступності цього ресурсу.

Ще одним методом керування проектами є екстремальне управління проектами. У критичних оглядах управління проектами відзначалося, що декілька методів управління проектами, які базуються на методиці програми оцінки та контролю (англ. Program Evaluation and Review Technique PERT), не в повній мірі задовольняють потреби багато-проектного середовища сучасних компаній.

Більшість з таких компаній орієнтовані на масштабні, короткострокові, не повторювані проекти, в яких усі види керування використовують інструменти проектного управління. Використання проектної моделі для «проектів», чи скоріше «задач», що тривають декілька тижнів, на практиці призводить до непотрібних витрат та слабкої гнучкості.

Замість використання класичного управління проектами, спеціалісти з керування проектами намагаються застосувати різні спрощенні методи, такі як гнучка методологія управління проектами.

Останнім методом керування проектів є такий підхід як проекти в контрольованому середовищі. Це упорядкований підхід до управління проектами, який був створений в 1996 році, як типовий метод керування проектами. Фактично це комбінація методології PROMPT (що еволюціонувала в методологію PRINCE) з методологію IBM MITP— управління впровадженням усього проекту. PRINCE2 пропонує метод управління проектами в рамках чітко визначеної структури організації.

PRINCE2 описує процедури координації людей та ресурсів у проекті, як проходить процес розробки та контролю проекту, дії що необхідно виконати у разі внесення змін до проекту у зв'язку з відхиленням від плану впровадження.

Кожен процес описується з ключовими вхідними та вихідними даними, а також цілями та діями, які необхідно виконати для досягнення цілей. Приклад такої моделі процесів подано на рисунку 1.4. Це надає можливість автоматично контролювати будь-яке неочікуване відхилення від плану.

Рисунок 1.4 – Модель процесів у методі проектів в контрольованому середовищі

Розподілення на етапи, якими можливо керувати, забезпечує ефективний контроль ресурсів. Впровадження проекту відбувається структуровано та контрольовано, завдяки інтегрованому контролю за виконанням.

1.3 Аналіз процесів керування проектами

Після дослідження методів, що лежать в основі керування проектами, було виявлено ряд процесів. Наступним кроком в деталізації проблематики досліджуваної області є аналіз процесів у керуванні проектами.

Зазвичай, керування проектами включає такий перелік елементів: чотири групи процесів та систему контролю. Незалежно від методології чи підходів, що застосовується, застосовуються одні й ті самі базові процеси управління проектами.

Групи процесів зазвичай включають:

– ініціювання;

– планування чи розробка;

– експлуатація чи виконання;

– моніторинг і контроль;

– завершення.

У проектному середовищі з дослідницьким нахилом (наприклад, дослідження та проектування) перераховані етапи можуть бути доповнені точками прийняття рішень, де визначається чи потрібно продовжувати чи припинити впровадження проекту.

Одним з процесів що має дуже важливий вплив на вартість проекту та успішність його загалом є саме етап планування, саме тому виникає необхідність розглянути його більш детально.

Після етапу ініціювання, проект планується з необхідним в залежності від типу проекту рівнем деталізації. Ціль такого планування – спланувати час, витрати та ресурси з метою об’єктивної оцінки роботи, яку необхідно зробити, та ефективного управління ризиками протягом виконання проекту [5]. Так само як для групи процесів ініціювання, недостатньо точний план значно знижує шанси проекту успішно завершити поставлені перед ним завдання.

Додаткові процеси, такі як планування комунікацій як між собою так і зі стороною замовника, визначення ролей у команді та зони відповідальності кожного члена команди, а також проведення попередньої зустрічі учасників проекту є запорукою того, що на старті проекту буде менше проблем з організацією процесу.

Для проектів з розробки нових продуктів, концептуальна розробка продукту може проводитися одночасно з плануванням, такий підхід може допомогти групі з планування, шляхом надання інформації про визначені результати проекту та заплановані задачі. Загальна схема процесів планування зображена на рисунку 1.5.

Рисунок 1.3 – Взаємозв’язок процесів планування

Планування проекту загалом складається з:

– визначення рівня деталізації та етапів впровадження;
– розробка документу, що окреслює зміст та межі проекту;
– визначення відповідальних за планування проекту;
– оцінка результатів проекту та структури декомпозиції робіт;
– визначення завдань необхідних для виконання проектних цілей та задання послідовності для таких завдань;
– оцінка вимог до ресурсів для забезпечення виконання завдань;
– оцінка часу та витрат на виконання завдань;
– розробка календарного плану;
– оцінка бюджету;

– планування можливих ризиків;
– отримання формальної згоди на початок роботи;

Опираючись на необхідність отримувати актуальну інформацію на кожному з етапів проекту, чітко розуміти процеси та компоненти що приймають участь у таких процесах, контролювати ресурси, знизити ризики у ході проекту, з'явилось таке поняття як системи контролю проектів.

Контроль проекту — це фактор, що гарантує відповідність проекту графіку виконання за часом та бюджету закладеного на цей проект. Контроль проекту розпочинається з планування робіт над проектом та закінчується детальним звітом з виконання проекту, враховуючи кожен елемент процесу управління проектом. Кожен проект має бути оціненим відносно рівня необхідного над ним контролю. Існує припущення, підтверджене багаторічним досвідом фахівців у галузі керування проектами, що забагато контролю у більшості випадків є причиною втрати часу, але у той же час замалий рівень контролю веде до збільшення ризиків.

Якщо контроль проекту застосований некоректно, то вартість для замовника та бізнесу пояснюється у термінах помилок, виправлень та додаткових витрат на аудит.

Системи контролю необхідні для моніторингу витрат, ризиків, якості, комунікацій, часу, змін, та людських ресурсів. У той же час аудитори як відповідальні за оптимізацію чи усунення недоліків у процесах мають визначити наскільки проекти впливають на фінансову звітність, наскільки достовірну інформацію отримують замовники і скільки точок контролю існує.

Аудитори мають розглянути процес розробки на предмет способу впровадження. Процес розробки та якість кінцевого продукту також можуть бути оцінені, якщо така виникає потреба.

Замовник може запросити від аудиторської фірми фіксування проблем на ранніх етапах з метою зменшення зусиль, часу, грошей, необхідних на виправлення. Аудитор може виступати як консультант з контролю, частина проектної команди або як окремий аудитор.

Бізнес іноді використовує формалізовані процеси розробки систем. Це допомагає досягти успішності розробки. Формальний процес ефективніший у створенні сильних точок контролю. Аудитори мають перевірити такий процес та підтвердити його якісну організацію та відповідність практиці. Ефективний формальний план впровадження системи характеризує контролю процесу:

– стратегія, задля приведення розробки до загальніших цілей організації;
– стандарти для нових систем;
– політики управління проектом щодо часу та бюджету;
– процедури, що описують процес;
– оцінка якості змін.

1.4 Трикутник управління проектами

Як будь-яке починання, проект виконується та завершується відповідно до заданих обмежень. У практиці керування проектами такими обмеженнями вважаються «зміст та межі», «час» та «вартість».

Такі обмеження отримали назву «Трикутник управління проектами» що зображений на рисунку 1.6, де кожна сторона є певним обмеженням. Жодна з сторін трикутника не може бути змінена, щоб не вплинути на інші сторони. Подальша деталізація обмежень відділяє «якість» чи «продуктивність» впровадження проекту від «змісту та меж» і перетворює якість в четверте обмеження.

Обмеження за часом — це час, за який необхідно завершити проект.

Вартість — це розмір бюджету, який виділений на реалізацію проекту.

Обмеження за змістом та межами — це завдання, які мають бути завершені для досягнення кінцевого результату проекту.

Зазначені три обмеження досить часто взаємопов'язані: збільшення меж та обсягів завдань, зазвичай призводить до збільшення часу та вартості, обмежений час може означати збільшення вартості чи зменшення змісту та меж проекту.

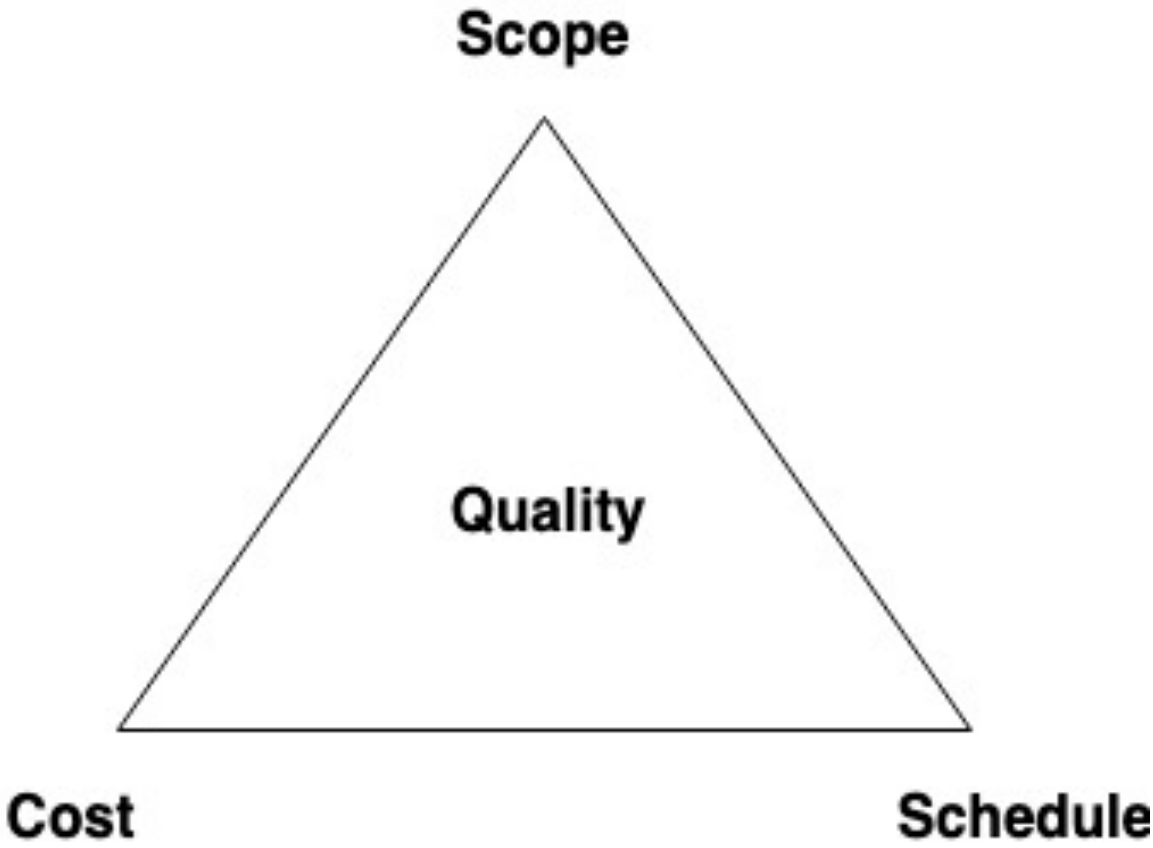

Рисунок 1.6 – Взаємозв’язок процесів планування

Наука про управління проектами забезпечує інструменти та методи, що надають змогу команді проекту (не лише менеджеру проекту) організувати свою роботу таким чином, щоб виконати вимоги щодо обмежень.

2 ДОСЛІДЖЕННЯ ПІДХОДІВ В УПРАВЛІННЯ ЧАСОМ

2.1 Опис процесів в управлінні часом

З усіх доступних метрик що фігурують у процесах керування проектами, незалежно від обраного метода керування проектами, чи процесами що впроваджені в ході оптимізації одним з найважливіших параметрів ефективності робочого процесу є час. Адже саме час фігурує у всіх напрямках робочого процесу, починаючи від процесу планування задачі до дати завершення проекту.

Вибравши напрямок для оптимізації робочого процесу, необхідно з’ясувати які існують методи та практики в управлінні часом, за рахунок чого можна здобути кращих результатів у плануванні, оцінки задач, досягнення цілей завдяки заданню пріоритетів. Для початку необхідно ввести поняття управління часом.

Управління часом, організація часу, тайм-менеджмент — технологія організації часу і підвищення ефективності його використання [6]. Це дія або процес тренування контролю над часом, витраченим на конкретні види діяльності, основною ціллю якого є збільшення ефективності та продуктивності.

В управлінні часом може допомогти ряд практик, інструментів і методів, що використовуються у ході виконання конкретних завдань у контексті проектів задля досягнення основних цілей та головної мети проекту.

Набір таких практик там методів, спрямованих на оптимізацію робочого процесу включає в себе такі дії як: планування, аналіз витрат, моніторинг, організацію, розподіл, постановку цілей, делегування, складання списків та розстановку пріоритетів.

На початку управління часом приписувалося тільки бізнесу або трудовій діяльності, але з часом термін розширився, включивши особисту діяльність.

Система управління часом становить поєднання процесів, інструментів, технік і методів. Зазвичай управління часом є необхідністю в розвитку будь-якого проекту, оскільки визначає час завершення проекту і масштаб.

Базовими підходами в процесі керування часом є постановка пріоритетів, декомпозиція великих завдань та проектів на окремі задачі. До управління часом належать також методики впливу з точки зору мотивації та повсякчасний прогрес а також контроль результатів.

Основні ідеї управління часом полягають в тому, що основну увагу необхідно приділяти саме тим задачам, що є важливішими, вміння раціонально розподіляти час, а не тому, як швидше робити справи. По суті, це досить важливе і складне завдання постановки цілей, визначення пріоритетів, яке зачіпає всі галузі управлінської діяльності, пов'язані з прийняттям рішень.

В управлінні часом можна виділити наступні процеси:

– аналіз;
– моделювання стратегій з урахуванням проведеного аналізу;
– цілепокладання: постановка мети або визначення ключового напрямку розвитку. Визначення та формулювання мети (цілей);
– планування і розстановка пріоритетів. Розробка плану досягнення поставлених цілей і виділення пріоритетних (першорядних) завдань для виконання;
– реалізація. Конкретні кроки і дії відповідно до наміченого плану і порядком досягнення мети.
– контроль досягнення мети, виконання планів, підбиття підсумків за результатами.

Також в разі, якщо особа або група осіб, які практикують управління часом, планують і далі здійснювати проекти, то доцільно вести хронометраж і фіксувати результати аналізу хронометражу у вигляді «карток проекту» (запис за параметрами різного характеру показників витрат часу на окремі завдання) для їх подальшого застосування в процесі будь-яких проектів або програм.

Для того, щоби визначити як наразі використовується час рекомендується протягом певного періоду вести щоденник, в якому нотується кількість часу необхідна для виконання типових задач. Це дозволяє точніше розраховувати власні

сили у плануванні. Що б оцінювати свій прогрес ефективним методом є щотижневий аналіз виконаного та того, що очікується на наступному тижні.

Для того, щоб план був ефективний необхідно дотримуватись простих правил.

Основні принципи складання плану:

– регулярність;

– системність;

– послідовність.

Для забезпечення реальності планування слід планувати такий обсяг завдань, з яким фахівець може реально впоратися.

Необхідно на регулярній основі фіксувати витрачений на задачі час. При цьому важливо відмічати, скільки часу і на яку саме задачу було витрачено. У результаті такої практики, працівник, маючи досить чітке представлення про затрачений час, може вносити корективи у свій план на майбутнє; для складання інформативного плану з детальним зазначенням використаного часу є необхідність розподілити свої завдання на довго, середньо і короткострокові.

Існує безліч різних засобів, що використовуються для планування та моніторингу використаного часу. Вибираючи найбільш відповідний для зазначених потреб метод, необхідно орієнтуватися на цілі та очікування щодо обраного методу. Процес організацій часу для кожного члена команди окремо, у сумі допоможе налагодити більш стабільний та чіткий робочий процес. Адже похибки у оцінці задач та не відповідність затраченого часу в порівнянні з очікуваним є причиною відсутності достатнього рівня контролю у минулому.

Іншою проблемою у досягнені успішності проекту є не коректна оцінка задач з точки зору пріоритетів, адже саме потреби замовника та бізнесу є основою для формування першочергових задач.

Саме тому виникає необхідність планування, складання списку завдань, визначення способів досягнення поставлених цілей. У щоденній метушні так легко забути про якісь події або справи. У великій кількості справ, які необхідно зробити

– деякі губляться і залишаються невиконаними. Цілі, здаються недосяжними, тому що не мають конкретного плану з їх досягнення.

На сьогодні існує чимало різних додатків, які у різній формі допомагають вирішити проблеми за розподілом часу на виконання задачі. Проаналізувавши ринок систем спрямованих на оптимізацію процесів керування робочим часом, можна розділити існуючі рішення на декілька категорій.

Програми для створення списків задач – найбільш доречні у застосуванні для персонального використання, допомагають не забути виконати важливі задачі, та візуалізують ці задачі, що спрощуэ оцінку об’єму роботи.

Дошки для завдань – такі системи зручно використовувати для пріоритезації завдань, та відслідковуванню прогресу, часто застосовуються на підприємствах.

Сервіси повідомлень – особливістю є можливість отримувати нотифікації-нагадування про найближчі події, які ви занесли до списку задач.

2.2 Аналіз використання робочого часу

Отримавши розуміння які саме процеси присутні в управлінні часом, виникає необхідність аналізу робочого часу, що допоможе виділити саме ті області що потребують застосування методів оптимізації.

Робочий час – час, що витрачається на виконання роботи, або час активного перебування на роботі, присвячене виконанню прямих службових обов'язків [7]. Реально робочим часом вважається час, зазначений в системі, призначеній для контролю використання робочого часу.

Виділивши поняття робочого часу, та опираючись на задачі, що виникають на протязі робочого дня, можна вивести наступні залежності.

Коефіцієнт екстенсивного використання робочого часу.

$$К_э = (Ф – Р)/Ф = 1 – Р/Ф$$

де Ф – фонд робочого часу, хв;

Р – регламентовані і нерегламентовані перерви в роботі, хв.

Коефіцієнт втрат робочого часу, які залежать від працівників.

$$К_п = П_з/Ф$$

де $П_з$ – втрати робочого часу, що залежать від працівника, хв.

Коефіцієнт втрат робочого часу, які обумовлені організаційно-технічними причинами і не залежать від працівника.

$$К_п = П_о/Ф$$

де $П_о$ – втрати з організаційно-технічних причин, які не залежать від працівників, хв.

Коефіцієнт витрат робочого часу на відпочинок і особисті потреби працівників.

$$К_{ол} = ОЛ/Ф$$

де ОЛ – витрати часу працівника на особисті потреби (обідня перерва, виробнича гімнастика, гігієна і т. п.), хв.

Аналогічно можна проаналізувати витрати часу працівника на виконання властивих йому функцій ($К_{св}$), невластивих йому функцій ($К_{нсв}$), на виконання творчих робіт ($К_т$), організаційно-адміністративної роботи ($К_{оа}$) і т. д.

Дефіцит робочого часу – нестача часового ресурсу, викликана неправильною організацією працівником своєї діяльності, або не документованою організацією діяльності керівництвом, що призводить до поспіху, затягування виконання робіт, завдань, неякісній роботі, втрат у виробництві,. Що в кінцевому підсумку істотно впливає на ефективність і результати роботи всього підприємства.

Одним з методів вдосконалення управління на підприємстві є аналіз витрат робочого часу керівника підприємства і керівників функціональних підрозділів, в даному випадку саме менеджера проекту, що відповідає за процес керування проектом.

Для виявлення причин браку часу необхідно періодично проводити аналіз часу за кілька робочих днів.

Найбільш характерні причини дефіциту часу :

- безплановість роботи як результат роботи не тільки самого менеджера, але і стилю роботи всієї команди;
- невідповідність працівника і займаної ним посади, ролі що він виконує на проекті;
- неадекватна оцінка, швидкості роботи, результативності;
- відсутність конкретно сформульованих задач і цілей.

Беручи до уваги вищеописаний ряд причин, що можуть стати причиною ризиків в успішності проекту, виникає необхідність дослідити методи та практики, що спрямовані на усунення цих проблем, а також на оптимізацію процесу керування часом та керування проектом.

2.3 Дослідження методів оптимізації використання робочого часу

Для раціонального використання часу командою менеджеру необхідно, перш за все, чітко усвідомити основні функції, цілі, завдання та бюджет в контексті виконуваного проекту. При плануванні слід враховувати ряд основних правил.

При складанні плану на день залишити 40% часу вільним, тобто 60% часу відвести на планові роботи, 20% на непередбачені, 20% на задачі що виникають спонтанно.

Необхідно постійно фіксувати витрачений на задачі час. При цьому слід відмічати, на які саме задачі час працівника було витрачено, на скільки витрачений

час співпадає з запланованим та причини що призвели до розбіжності цих показників. Адже проаналізувавши саме ці характеристики менеджер може зробити висновки та уникати подібних ситуацій у майбутньому.

Для складання хорошого плану з чітким зазначенням використаного часу необхідно розподілити задачі з точки зору довго строковості, пріоритету, та провести оцінку задач з точки зору спеціалістів що будуть відповідальні за ці задачі. Врахувати залежності та можливі ризики.

Для забезпечення реальності планування слід планувати такий обсяг завдань, з яким команда може реально впоратися, беручи до уваги непередбачувані обставини та рівень кваліфікації кожного з фахівців.

Основою плану використання часу та прогресу проекту може служити його перспективний план. З урахуванням плану на період виконання проекту також складають план на коротші ітерації, який охоплює цілі на коротші відрізки часу. І таким чином відбувається детальне планування, базуючись на основних потребах проектної діяльності.

Ітеративні плани можуть бути скоординовані з глобальним планом і поділені на відрізкові плани. План на робочий день являє собою найважливішу сходинку в плануванні робочого часу, він дозволяє контролювати та вносити корективи з урахуванням обстановки.

Планування означає підготовку до реалізації цілей і упорядкування робочого часу. З практики відомо, що при витраті 10 хвилин на планування робочого часу можна щодня заощадити до двох годин.

Виявивши ряд проблем, що існують в управлінні часом, необхідно розглянути існуючі методи що допомагають вирішити такі проблеми.

Упорядкування планів дня за допомогою методу «Альп». Цей метод охоплює п'ять стадій:

- впорядкування завдань;
- оцінка тривалості дій;
- резервування часу (у співвідношенні 60:40);
- прийняття рішень по пріоритетах і передоручення;

– контроль обліку виконаного.

Черговість виконання справ можна встановлювати за допомогою принципу Парето (в співвідношенні 80:20). Цей принцип означає, що в середині цієї групи або множини окремі малі частини є більш значущими, ніж в загальному в цій групі.

Відповідно до цієї теорії можна зробити висновок щодо використання робочого часу фахівця: за перші 20% витраченого часу досягається 80% результату. Решту 80% витраченого часу дають лише 20% загального результату.

Ще одним методом у плануванні задача є встановлення пріоритетності завдань за допомогою АВС-аналізу [8].

Ця техніка заснована на тому, що частки у відсотках найбільш важливих і найменш важливих справ в сумі залишаються незмінними. Всі завдання поділяються на три класи відповідно до їх значимості. Аналіз АВС базується на трьох закономірностях.

Найбільш важливі справи становлять 15% загальної їх кількості, якими займається фахівець. Внесок цих задач для досягнення цілі складає близько 65%.

Важливі завдання становлять 20% загальної їх кількості, значимість їх для досягнення мети приблизно дорівнює 20%.

Менш важливі (малоістотні) завдання становлять 65% загальної їх кількості, а їх значущість дорівнює 15%.

Для використання АВС-аналізу необхідно дотримуватись наступних правил:

– скласти список всіх майбутніх завдань;
– систематизувати їх за важливістю та встановити черговість;
– пронумерувати ці завдання;
– оцінити завдання відповідно за категоріями А, В і С;
– завдання категорії А (15% загальної їх кількості) вирішуються у першу чергу;
– завдання категорії В (20%) підлягають другорядному виконанню;
– завдання категорії С відкладаються та переплануються на наступний день.

Для того щоб досягти максимальної ефективності при плануванні робочого дня та пріоритезації задач, слід також звернути увагу на такі сучасні методи планування як матриця Ейзенхауера та метод Помодоро.

Прискорений аналіз за принципом Ейзенхауера. Цей принцип є допоміжним у тих випадках, коли необхідно терміново прийняти рішення про пріоритетність виконання завдань. Пріоритети встановлюються за такими критеріями, як терміновість і важливість завдання. Вони підрозділяються на чотири групи.

Президент США Дуайт Девід Ейзенхауер запропонував просту методику для визначення пріоритетів серед списку всіх поточних дій. Це здійснюється за допомогою градації задач стосовно їхньої терміновості та важливості як наведено у таблиці 1.1.

Таблиця 1.1 — Матриця Ейзенхауера

Важливість/Терміновість	Терміново	Не терміново
Важливо	Розрішення кризових ситуацій, невідкладні справи, проекти у яких підходять терміни здачі.	Планування нових проектів, оцінка отриманих результатів, налагодження відносин, розгляд нових перспектив.
Не важливо	Декотрі телефонні дзвінки, перерви, певні наради, соціальна діяльність.	Рутинна робота, пошта.

Як інструмент, матриця Ейзенхауера працює і, більш того, добре організує, якщо їй слідувати. Найважливішим аспектом роботи команди є квадрат Важливо та Терміново. Необхідно щоб Терміново був частіше порожнім, а це цілком реально, якщо частіше займатися справами квадрата Важливо, але не Терміново.

Як використовувати цю матрицю? Просто розподіліть туди проектні задачі у відповідності з їх важливістю і терміновістю.

Метод Помодоро — це метод управління часом, розроблений Франческо Чірілло в кінці 1980-х. Цей метод використовує таймер для того, щоб розбити роботу на 25-хвилинні інтервали, що надає необхідні умови для більш точної оцінки задач у майбутньому, які називаються «pomodori» (італійське слово, що означає «помідори») і розділені короткими перервами.

Є п'ять кроків до виконання методу:

– оберіть задачу, яку потрібно зробити;
– встановіть таймер на 25 хвилин;
– працюйте, поки таймер не продзвенить; запишіть це позначкою «х»;
– зробіть коротку перерву (від 3 до 5 хвилин);
– кожні чотири «pomodori» робіть довшу перерву (15–30 хвилин).

Етапи планування, моніторингу, запису, обробки та візуалізації є фундаментальними для методу. У фазі планування задачі пріоритизуються за допомогою запису їх в щоденний список задач. Це дозволяє користувачам оцінити зусилля потрібні для задачі. Коли «pomodori» закінчуються, вони записуються, додаючи відчуття досягнення та надаючи чисті дані для подальшого аналізу та чіткого освідомлення які задачі вимагаються більше часу на виконання.

Для потреб методу термін «помодоро» позначає неподільний 25-хвилинний відрізок часу. Після завершення задачі весь час, що залишився в «помодоро», присвячується перенавчанню. Регулярні перерви робляться з метою асиміляції.

Короткий (від 3 до 5 хвилин) відпочинок розділяє послідовні «помодорі». Чотири «помодорі» утворюють набір. Між наборами помодорі роблять довшу перерву (на 15-30 хвилин).

Найголовнішою метою методу є зменшення впливу внутрішніх та зовнішніх відволікаючих факторів на увагу та потік. «Помодоро» неподільний. Якщо під час «помодоро» роботу переривають, то або інша діяльність повинна бути занотованою та відкладеною або ж «помодоро» повинен бути відкинутим.

3 ОПИС ПРОБЛЕМ ЩО ВИРІШУЮТЬСЯ

3.1 Постановка задачі

Дослідивши проблеми що існують у процесах керування проектами, розглянувши методи, що спрямовані на оптимізацію таких процесів було вирішено створити систему, яка буде зручним інструментом для контролю часових меж у роботі команди та зможе допомогти менеджеру команди як відповідальній за процеси людині контролювати задачі які займають найбільшу частину, враховувати ризики, проводити аналіз ефективності роботи команди, та зосередитися на виконанні пріоритетних задач.

Метою є розробка веб-орієнтованої системи, призначеної для додання задач, та визначенню їх пріоритету на основі такого методу як Матриця Ейзензауера, можливість відстежувати час затрачений на певну справу використовуючи підхід Pomador.

З точки зору менеджера проекту головною задачею є контроль слідування правилам планування зі сторони усіх членів команди. У разі якщо всі необхідні кроки в процесі планування будуть витримані, менеджер проекту отримає можливість проводити аналіз ефективності роботи команди, враховувати можливі ризики, більш точно проводити оцінку нових задач.

Система має давати змогу для управління задачами у розкладі. Найбільш адаптивним та легкодоступним способом взаємодії з системою є робота в браузері.

Функціонал повинен забезпечувати базовий функціонал для управління та зручного сприйняття користувачами.

Створення програмної системи складається з реалізації наступних задач:

- реєстрація користувача у системі;
- можливість додання нових задач;
- можливість вибору категорій (meeting, task), пріоритету, дати виконання задачі;

– пріоритезація задач за мітками матриці Ейзенхауера (терміново/не терміново), (важливо/не важливо);
– можливість приступити к виконанню задачі, для якої задана кількість часу у pomodor (задаються на етапі створення задачі);
– перегляд задач у різних видах, як календар, та список;
– можливість перевести задачу у стан виконаної;
– пошук за фільтрами;
– бали продуктивності (нараховуються за кожну виконану за день задачу).

Для реалізації системи було обрано середовище розробки WebStorm. Так як засіб буде розроблено у вигляді веб-орієнтованої системи, це забезпечить кросплатформеність та можливість роботи у будь якій системі – з сімейства Windows або Unix-подібних систем.

Мовою написання серверної частини був обраний Node.js з використанням веб-фреймворку express. В якості сховища даних була обрана СУБД MongoDB та ORM для неї Mongoose. Для реалізації клієнтської частини обрані технології HTML5/CCS3, JavaScript, з використанням фреймворків React та Redux та бібліотеки Bootstrap.

У ході виконання роботи необхідно зробити такі завдання:
– провести концептуальне та UML-моделювання предметної галузі;
– розробити схему бази даних для зберігання даних;
– розробити алгоритми пріоритезації задач
– виконати програмну реалізацію веб-системи;
– провести тестування розробленого програмного продукту.

3.2 Аналіз аналогів

Розглянувши існуючі на ринку засоби для підтримки процесу керування часом, було знайдено багато готових рішень. Основними їх перевагами є надання

можливості користувачу вести персональні електронні органайзери, мати завжди до них доступ з комп’ютера, планшета чи смартфона.

Але зважаючи на усі зазначені плюси, що мають такі програмні продукти, слід зауважити, що вони вирішують не всі задачі. А саме керування задачами у контексті одного проекту, детальний моніторинг виконання задач, та можливість відстежувати стан проекту на конкретний момент.

Отже, виникає наочна необхідність у створенні програмної системи, що дозволить спростити та пришвидшити виконання цих завдань, шляхом їх автоматизації та візуалізації.

Ці програми мають як і ряд переваг, так і деякі недоліки. Більшість з цих програмних засобів створені тільки для візуалізації та зберіганнях власного розкладу у електронному вигляді. Розглянемо найбільш популярні на ринку програми з них.

Rescuetime, приклад інтерфейсу наведено на рисунку 3.1, представляє собою платформу, для поліпшення процесу планування робочого дня, та забезпечення умов, що сприятимуть втіленню цього плану у життя.

З безліччю сторонніх справ у цифровому житті легко розсіятися. RescueTime допомагає вам зрозуміти щоденні звички, щоб надати можливість зосередитись і бути більш продуктивними.

Основні можливості:

- встановити сповіщення, щоб ви знали коли витрачено певну кількість часу на активності, можливо більше ніж розраховувалось;
- відмітки про виконані протягом дня справи;
- блокування відволікаючих сайтів, вибираючи кількість часу, щоб зосередитися;
- аналіз часу витраченого на зустрічі та мітинги, ну роботу з поштою;
- таймери та лічильники на виконання задач.

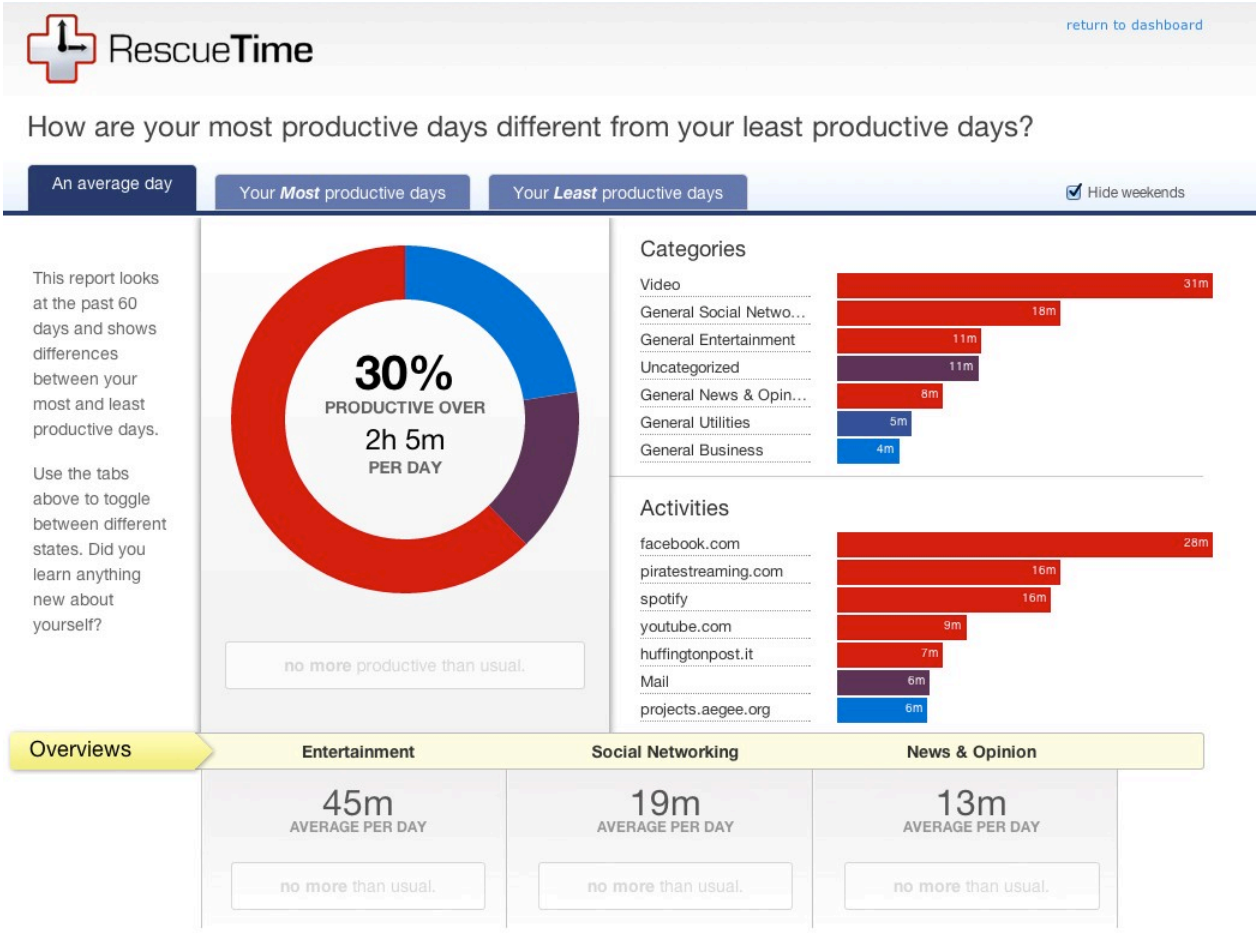

Рисунок 3.1 – Інтерфейс RescueTime

Існує безкоштовна та преміум версія сервісу. Використовуючи безкоштовну версію, ви отримуєте можливість задавати цілі на день, відстежувати активності на сайтах.

Для того щоб отримати можливість використовувати основний функціонал, нотифікаціїї, повідомлення на пошту, необхідно придбати преміум версію.

Особливістю сервісу є можливість запустити додаток у фоні, та відстежувати час витрачений на різноманітні сайти, і як результат отримати звіт що базується на активності за день. Сприяти росту продуктивності.

Головним недоліком є відсутність веб версії, відсутність інструментів для пріоритезації задач. Також до недоліків можна віднести занадто високу ціну.

Mylifeorganized, представлено на рисунку 3.2. На відміну від попереднього рішення цей додаток більш направлений на візуалізацію задач та побудову графіків ефективності. Сервіс надає можливість використовувати окремі додатки для

персонального комп'ютера, мобільними пристроями під управлінням систем Android та IOS. Головним недоліком є відсутність веб версії.

Цікавою функцією на мобільних додатках є можливість отримувати нагадування коли знаходишся в певній локації, за умови що у тебе увімкнений навігатор, та ти додав адресу до задачі. Синхронізація, додавання задач через пошту та геопозиція для завдання доступні лише у преміум версії.

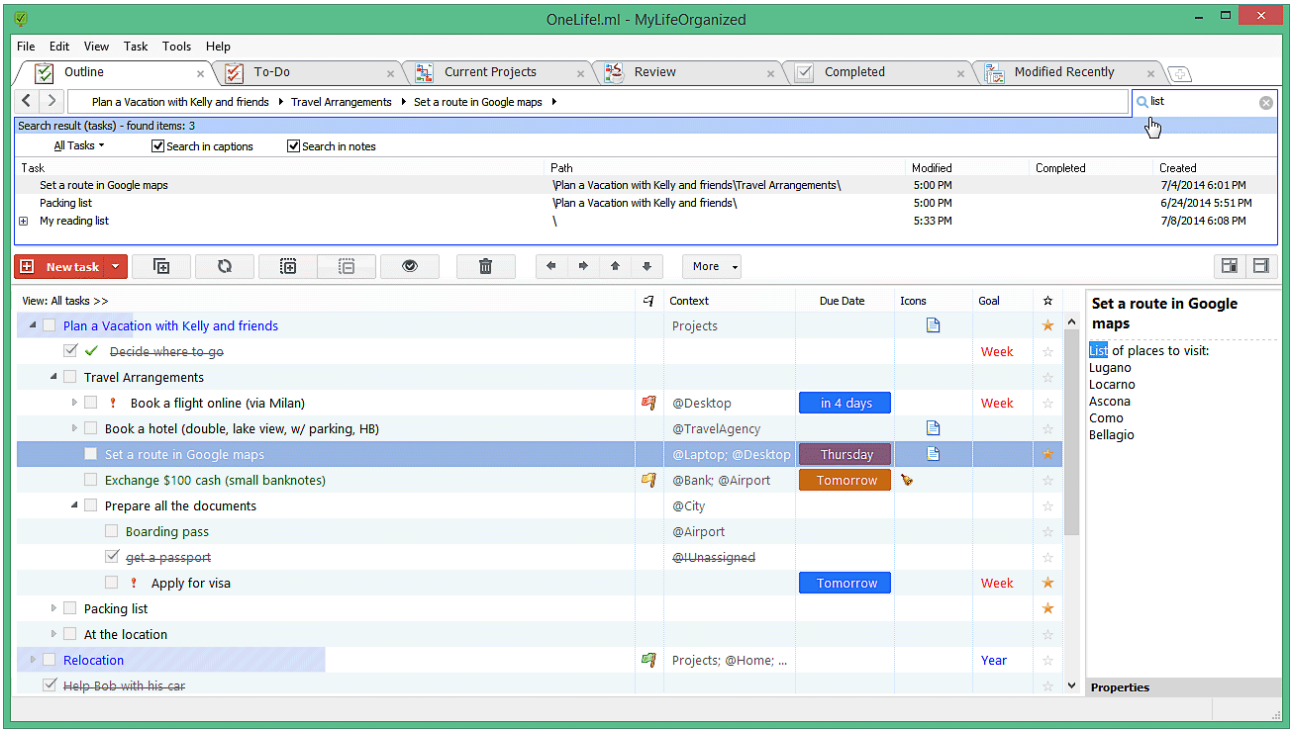

Рисунок 3.2 – Інтерфейс MyLifeOrganized

Основні можливості:

– простий інтерфейс перетягування дозволяє переставити завдання в простому списку або організувати їх в дерево;
– MLO дозволяє розбити задачу на підзадачі , підзадачу ще на підзадачу і так далі;
– ви можете створювати гнучкі ієрархічні списки і додати залежності між завданнями;

- після того, як ви додали дату, контекст і залежності, MLO буде автоматично генерувати список елементів дій, які вимагають негайної уваги;
- отримати потрібну інформацію в потрібному місці. MLO призначений для відправки вам нагадування, коли ви приїдете до одного з зазначених місць;
- MLO синхронізується на всі пристрої;
- можливо створювати нові завдання, відправивши по електронній пошті лист на вашу поштову скриньку MLO. Потім ви можете конвертувати їх в задачі.

Основним недоліком є необхідність окремо купувати додаток для кожної з платформ, тобто якщо у вас є необхідність користуватись додатком на персональному комп'ютері та на смартфоні, разом вийде близько 90 доларів.

Також додаток має дуже складний інтерфейс, і приступити до роботи з ним без перегляду інструкції не можливо, що в свою чергу створює проблему іншого роду, а саме незручність в аналізі виконаних задач та формування загального прогресу для проекту.

RemeberTheMilk подано на рисунку 3.3 – багаторічний помічник прихильників планування, сервіс, що дозволяє не допустити ситуації що є причинами дедлайнів, авралів і прикрих випадковостей.

Однією в важливих сторін є можливість використання абсолютно на любій платформі а також веб-додаток. Синхронізація з поштовими сервісами. Назначати відповідальних, ділитись задачами, отримувати нотифікації можна лише використовуючи преміум версію.

Зареєструвавшись на сайті, користувач може почати записувати завдання (ті, що потрібно зробити). Завдання бувають особистими, освітніми, робочими і різними іншими. Так чи інакше, всі вони поділені на групи, які можна додавати і видаляти самостійно. Крім того, кожній задачі можна привласнити дату і час закінчення виконання, мітку, за якою її буде легше знайти, і місце, де ця задача повинна бути виконана.

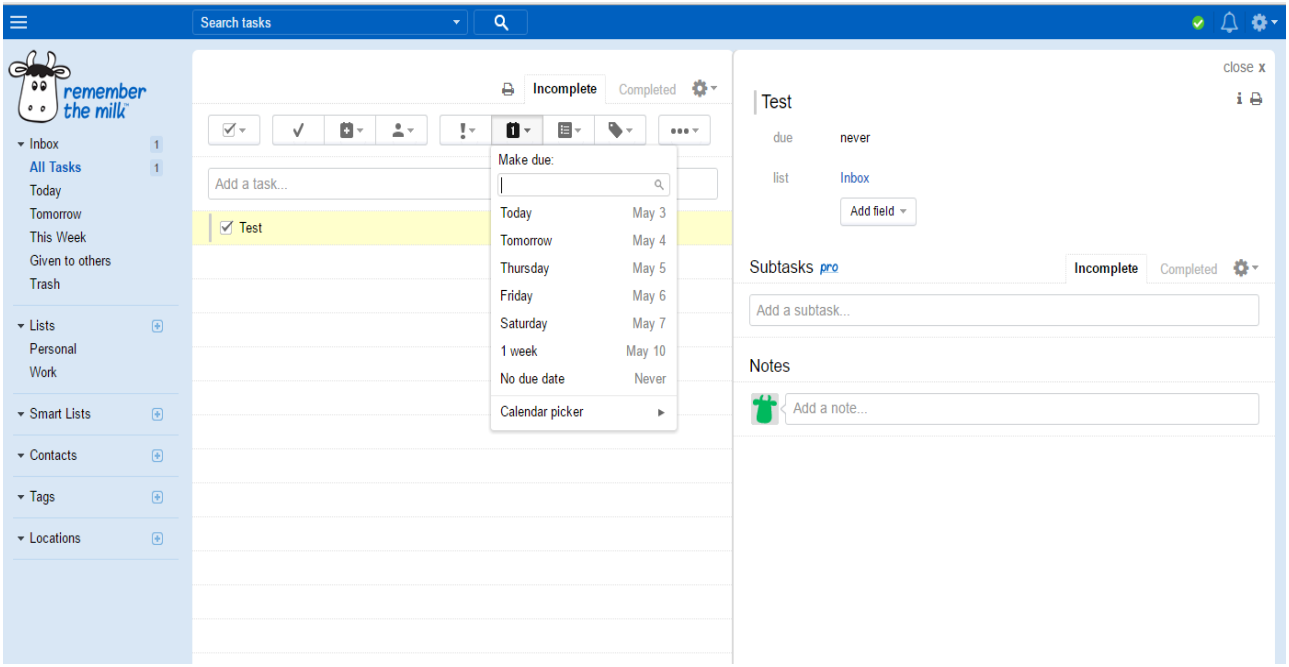

Рисунок 3.3 – Інтерфейс Remember the milk

Основні можливості:

– створення списку справ;
– синхронізація між пристроями;
– нагадування у скайп, пошту, твіттер;
– можливість ділитися списками, та назначати виконавців, якщо хтось з ваших друзів також використовує додаток;
– велика кількість характеристик для завдання (пріорітет, кінцева дата виконання, повторюваність, теги та інше);
– пошук по фільтрам (по тегам, по даті, по виконавцю).

Щоб пам'ятати про терміни, в які завдання повинно бути виконано, можна налаштувати повідомлення, які будуть приходити на вказану поштову скриньку, номер ICQ, Skype або телефонний номер (за SMS). Завдяки цьому користувач звільняється від постійного відвідування сайту з метою дізнатися, що і коли заплановано.

Підсумовуючи результати аналізу аналогів, основні недоліки та переваги додатків можна винести у таблицю 1.2:

Таблиця 1.2 – Порівняльна характеристика аналогів

	Rescue time	My life organized	Remember the milk
Зручність інтерфейсу та навігація	Висока зручність	Складний інтерфейс	Складна навігація
Нотифікації через інші сервіси	+	-	+
Веб-версія	-	-	+
Ціна на преміум версію	72$	60$, 30$	40$

Кожен з додатків має ряд особливостей, та досить не погано справляється з поставленими задачами, та жоден з аналогів не є вирішенням описаних раніше проблем, вони більш зосереджені на візуалізації та синхронізації задач.

Таким чином жодна з існуючих систем не покриває повність поставлені задачі і жодна із систем не є універсальною, оскільки в додатках наявний чіткий баланс між ціною та функціоналом. Кожний сервіс вдало виконує певні завдання у своїй предметній області, однак не може максимально повно вирішити усі проблеми, що стають перед користувачем.

Слід зауважити, що аналіз схожих за призначенням додатків проводився за допомогою публічної інформації та безкоштовних періодів використання сервісів, тобто більша частина недоліків була знайдена емпірично та не може бути виявлена повністю через брак достовірних даних.

Враховуючи вищезазначені проблеми, постає питання про те, як можна полегшити процеси керування часом а як наслідок і керування проектом, спростити процес планування, та досягти максимальної ефективності протягом дня.

Розглянувши існуючі на ринку засоби для підтримки керування часом, було знайдено багато готових рішень. Основними їх перевагами є надання можливості користувачу вести персональні електронні органайзери, мати завжди до них доступ з комп’ютера, планшета чи смартфона. Але зважаючи на усі плюси, що мають такі програмні продукти, слід зазначити, що вони вирішують не всі задачі, описані вище.

Отже, виникає наочна необхідність у створенні програмної системи, що дозволить спростити та пришвидшити виконання цих завдань, шляхом їх автоматизації та візуалізації.

3.3 Призначення розробки

Отже, у результаті проведеного аналізу предметної області було вирішено створити програмний додаток «Система керування робочим часом», що допоможе вирішувати вищезазначені проблеми в організації процесу планування, оцінки задач та пріоритезації.

Перед початком роботи над програмною системою необхідно визначитися з вимогами до неї. У загальному випадку під вимогами розуміють сукупність властивостей, які повинна мати система, що реалізується. У ході роботи треба реалізувати програмну систему що вирішуватиме поставлені задачі.

Враховуючи складність розробки рішень для організації робочого процесу було поставлено завдання виявити ті вимоги, виконання яких принесе максимальну користь під час використання системи.

Даний етап роботи вимагає чіткого розуміння кінцевого результату та переліку необхідних дій задля його успішного та своєчасного досягнення. Таким чином, був розглянутий процес управління часом в цілому та проведена декомпозиція пов’язаних з ним процесів

Основна увага була приділена розкладанню дизайну на окремі функціональні або логічні компоненти, певні інтерфейси, що надають чітко окреслений функціонал в рамках їх відповідальності, методи, події і властивості [9].

Користувацький інтерфейс (англ. User Interface, UI) та взаємодія (англ. User Experience, UX) мають бути лаконічним та розробленими згідно гнучкого веб-інтерфейсу, що адаптується під мобільні платформи відповідно. Дизайн веб-сторінок має забезпечувати оптимальне відображення та взаємодію сайту з користувачем незалежно від роздільної здатності та формату пристрою, з якого здійснюється перегляд сторінки. Метою розробки дизайну є практичне відображення інформації та зручна навігація на всіх пристроях із доступом до інтернету (від стаціонарних ПК до мобільних телефонів).

Додаток «Система керування робочим часом» призначений для проектів та команд де є необхідність формалізувати потреби.

Дана система має простий у розумінні інтерфейс, тому не потребує багато часу для навчання користування, і , як наслідок, навпаки зменшує час, що потрібен користувачу для виконання задач. Маючи можливість зазначити в розкладі справи на конкретний час, користувач отримує змогу запобігти накладанням у розкладі.

Загальні відомості про розробку системи полягають у наступному: веб додаток «Система керування робочим часом» має бути розроблено за допомогою мови програмування JavaScript, а саме, платформи Node.js, фреймворку express. та технологій HTML5/CCS3, JavaScript, з використанням фреймворку React та Redux й бібліотеки Bootstrap. В якості сховища даних була обрана СУБД MongoDB та Mangoose ORM.

Засіб було вирішено створити у вигляді веб-додатку (сайту), що підтримує роботу лише з авторизованими користувачами. Якщо користувач вже зареєстрований, після вводу свого логіну і паролю він має змогу переглянути свій розклад та оновити його у відповідності до вимог. У іншому разі необхідно пройти етап реєстрації, для того щоб отримати доступ до системи

Основна функціональність даної системи полягає у наступному:

– реєстрація користувача у системі;

- можливість додання нових задач;
- можливість вибору категорій (meeting, task), пріоритету, дати виконання задачі;
- пріоритезація задач за мітками матриці Ейзенхауера (терміново/не терміново), (важливо/не важливо);
- можливість приступити к виконанню задачі, для якої задана кількість часу у pomador (задаються на етапі створення задачі);
- перегляд задач у різних видах, як календар, та список;
- можливість перевести задачу у стан виконаної;
- пошук за фільтрами;
- бали продуктивності (нараховуються за кожну виконану за день справу).

При створенні системи повинно бути взято до уваги:

- зручність використання сайту – інтерфейс має бути зручним та інтуїтивно зрозумілим, не вимагати додаткової підготовки користувачів;
- довідкова інформації щодо використання системи повинна бути розташована у відповідному розділі на сайті.

Система розробляється у вигляді веб-додатку, тож вона не потребує специфічних умов для того, щоб користувач зміг працювати з нею. Корисним для користувача буде знайомство з відповідними техніками для оптимізації робочого процесу, а саме матриця Ейзенхауера та методом Помодоро.

Серед необхідних вимог для початку роботи можна назвати лише:

- комп’ютер з підключенням до мережі Інтернет;
- наявність програми для перегляду веб-сторінок – браузеру.

4 АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Концептуальне моделювання предметної області

Метою концептуального моделювання є огляд предметної області та створення концептуальної моделі, що об'єднує уявлення про вміст бази даних, отримані в результаті вивчення предметної області.

Іншими словами, концептуальна модель – це опис структури БД, що виконаний з використанням природної мови, таблиць, діаграм і інших засобів, зрозумілих всім людям, що працюють над проектуванням бази даних [10]. Основним компонентом концептуальної моделі є опис об'єктів предметної області і зв'язків між ними [11].

Для даної предметної галузі основними поняттями є проект, задача, користувач. Загальна схема зв’язків об'єктів предметної області відображена на рисунку 4.1. Щоб визначити допоміжні сутності необхідно провести аналіз цих понять та виявити їх характеристики.

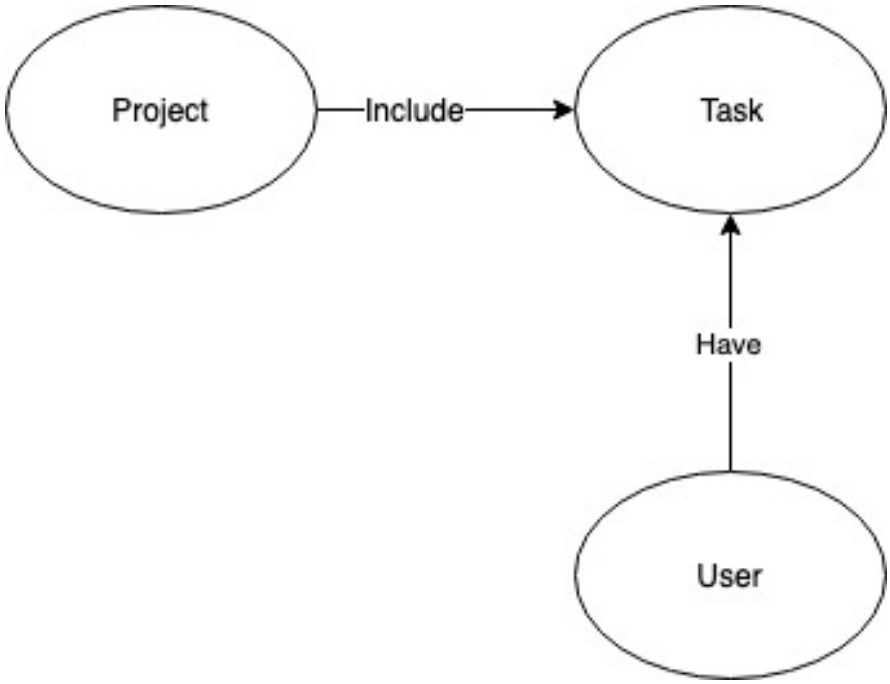

Рисунок 4.1 – Загальна схема взаємозв’язку об’єктів предметної області

Користувач системи матиме змогу реєструватися у системі, редагувати свої дані, переглядати свій розклад. Також користувач має змогу зіставляти свій розклад, виконувати задачі, чи виконувати їх застосовуючи метод Помодоро.

Отже, основними сутностями предметної галузі є «Користувач», «Задача», «Проект», У ході проектування між цими сутностями були виявлені наступні взаємозв’язки:

– користувач має задачі;

– проект містить багато задач.

Отже, у ході аналізу та концептуального моделювання були виявлені основні сутності системи та їх атрибути, що мають стати базовими при моделюванні та розробці системи.

Такими сутностями є:

– користувач;

– задача;

– проект.

Сутність «Користувач» має наступні атрибути:

– унікальний ідентифікатор;

– персональні дані (прізвище, ім’я);

– email;

– логін для входу у систему;

– пароль.

Сутність «Задача» повинна мати такі атрибути:

– унікальний ідентифікатор;

– назва;

– категорія (task, meeting);

– дата;

– тривалість;

– кількість pomodoro закладену на виконання;

– кількість балів;

– флаг нагадування;

– важливо/не важливо за матрицею Ейзенхауера;
– терміново/не терміново за матрицею Ейзенхауера.
– унікальний ідентифікатор користувача;
– унікальний ідентифікатор проекту;
– статус.

Сутність «Проект» має такі атрибути:

– унікальний ідентифікатор;
– кількість задач;
– набрані бали.

Моделювання розроблюваної системи проводиться з використанням мови UML для побудови діаграм, що допоможуть відобразити функціональність та внутрішню структуру системи.

UML – мова графічного опису для об'єктного моделювання в області розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, який використовує графічні позначення для створення абстрактної моделі системи, що називається UML-моделлю [12].

Можна виділити наступні переваги UML:

– UML об'єктно-орієнтований, в результаті чого методи опису результатів аналізу і проектування семантично близькі до методів програмування на сучасних об'єктно-орієнтованих мовах;
– діаграми UML порівняно прості для читання після досить швидкого ознайомлення з його синтаксисом;
– UML розширює і дозволяє вводити власні текстові та графічні стереотипи.

При проектуванні системи були розроблені діаграми наступних типів: діаграма прецедентів та діаграма розгортання.

Діаграма прецедентів описує функціональне призначення системи або те, що система повинна робити [13]. Розробка діаграми має такі цілі:

– визначити контекст предметної області, яка моделюється;
– сформулювати загальні вимоги до функціонального поведінки проектованої системи;

– розробити вихідну концептуальну модель системи для її подальшої деталізації у формі логічних і фізичних моделей.

Суть діаграми прецедентів полягає в наступному. Проектована система представляється у вигляді безлічі сутностей або акторів, що взаємодіють з системою за допомогою варіантів використання. При цьому актором або дійовою особою називається будь-яка сутність, що взаємодіє з системою ззовні.

Діаграма прецедентів для даної системи відображена на рисунку 4.2.

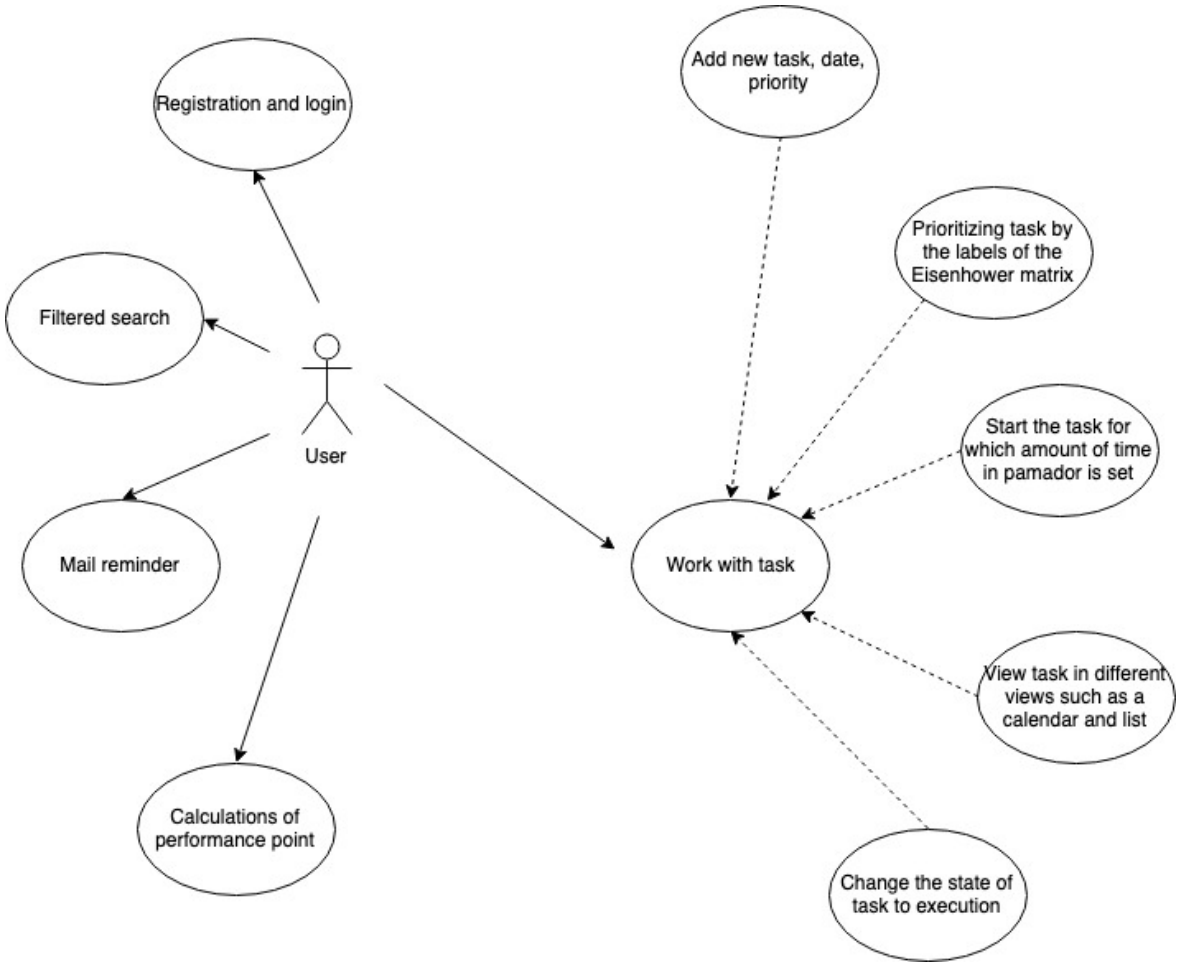

Рисунок 4.2 – Діаграма прецедентів

Діаграма прецедентів моделює систему з точки зору користувача і відображає доступні йому можливості. Діаграми цього типу допомагають при аналізі вимог до системи та проектуванні інтерфейсу користувача.

На діаграмі зображені основні функції системи, що доступні користувачам. При переході на сайт користувач потрапляє на головну сторінку, де користувачу

пропонується зареєструватися у системі, щоб розпочати роботу з основними можливостями системи.

Основний функціонал, доступний користувачу системи включає в себе наступні можливості:

- реєстрація користувача у системі;
- можливість додання нових задач;
- можливість вибору категорій (meeting, task), пріоритету, дати виконання задачі;
- пріоритезація задач за мітками матриці Ейзенхауера (терміново/не терміново), (важливо/не важливо);
- можливість приступити к виконанню задачі, для якої задана кількість часу у pomador (задаються на етапі створення задачі);
- перегляд задач у різних видах, як календар, та список;
- можливість перевести задачу у стан виконаної;
- пошук за фільтрами;
- бали продуктивності (нараховуються за кожну виконану за день справу).

Фізичне представлення програмної системи разом з інформацією про те, на якій платформі вона реалізована, відображається на діаграмі розгортання.

Додаток «Система керування робочим часом» складається з таких основних частин:

- серверу (Node.js);
- веб–кліенту (React & Redux);
- бази даних (MongoDB & Mangoose).

Діаграма розгортання – діаграма в UML, на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах.

Компоненти відповідають представленню робочих екземплярів одиниць коду. Компоненти, що не мають представлення під час роботи програми на таких діаграмах не відображаються; натомість, їх можна відобразити на діаграмах компонент.

Взаємодія компонентів системи представлена на діаграмі розгортання, що зображена на рисунку 4.3. На ній показані всі зв’язки та деякі нефункціональні вимоги.

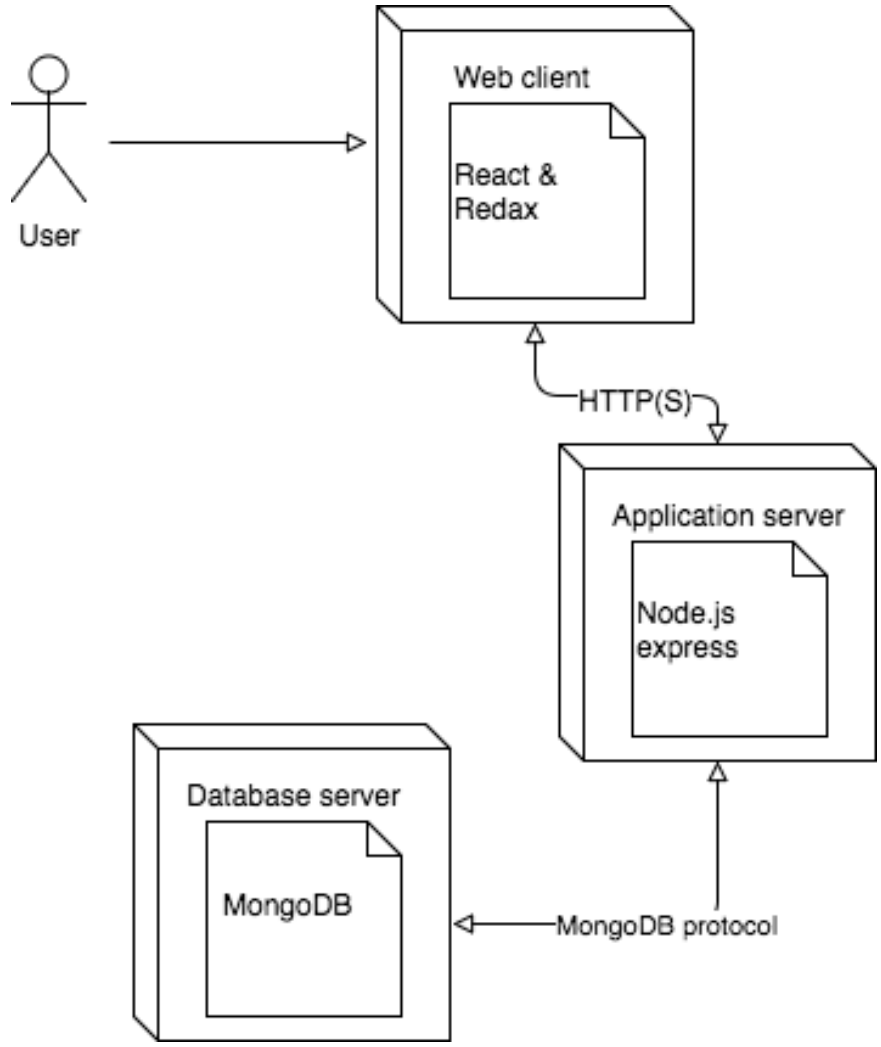

Рисунок 4.3 – Діаграма розгортання

Діаграма розгортання відображає робочі екземпляри компонент. Вона застосовується для представлення загальної конфігурації і топології розподіленої програмної системи і містить розподіл компонентів по окремих вузлах системи.

Крім того, діаграма розгортання показує наявність фізичних сполук-маршрутів передачі інформації між апаратними пристроями, задіяними в реалізації системи. Діаграма компонентів допомагає виявити вузькі місця системи і реконфігурувати її топологію для досягнення необхідної продуктивності.

Веб клієнт був побудований за допомогою сучасних технологій HTML5/CCS3, JavaScript, а також з використанням фреймворку React та Redux та бібліотеки Bootstrap. Серверна частина додатку була розроблена за допомогою Node.js з використанням фреймворку express. Node.js був обраний у зв’язку з тим, що розробка як backend частини, так і frontend ведеться однією мово програмування, та дані між частинами додатку передаються у JSON–форматі, що є нативним для JavaScript. В якості сховища даних була обрана СУБД MongoDB з використанням ORM Mangoose.

4.2 Проектування архітектури

На даному етапі можна виділити такі основні частини, необхідні для функціонування додатку та забезпечення виконання усіх необхідних операцій як клієнт, сервер, та база даних. Node js для серверної частини, React для клієнтської частини, MongoDB як база даних.

При проектуванні програмного продукту була обрана двошарова архітектура, яка забезпечує роботу системи при злагодженій роботі WEB-клієнта та сервера баз даних [14].

Модель клієнт-серверної взаємодії визначається перш за все розподілом обов'язків між клієнтом та сервером.

Наступним кроком є визначення зон відповідальності кожного з рівнів, що присутні в обраному підході

З логічної точки зору можна відокремити такі рівні операцій:

– рівень представлення даних, який по суті являє собою інтерфейс користувача і відповідає за представлення даних користувачеві і введення від нього керуючих команд;
– прикладний рівень, який реалізує основну логіку додатку і на якому здійснюється необхідна обробка інформації;

– рівень управління даними, який забезпечує зберігання даних та доступ до них.

Дворівнева клієнт-серверна архітектура передбачає взаємодію двох програмних модулів клієнтського та серверного.

В залежності від того, як між ними розподіляються наведені вище функції, розрізняють:

– модель тонкого клієнта, в рамках якої вся логіка застосунку та управління даними зосереджена на сервері. Клієнтська програма забезпечує тільки функції рівня представлення;
– модель товстого клієнта, в якій сервер тільки керує даними, а обробка інформації та інтерфейс користувача зосереджені на стороні клієнта. Товстими клієнтами часто також називають пристрої з обмеженою потужністю: кишенькові комп'ютери, мобільні телефони та інші.

В ході розробки була обрана модель товстого клієнта, оскільки вона забезпечує більш гнучке налаштування та масштабованість додатка. Реалізація сервісів, доступних із веб-браузерів, як правило, базується на трьохрівневій архітектурі.

Серверна частина побудована на класичній концепції MVC, для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

Контролер одержує вхідні дані й перетворює їх на команди для моделі чи вигляду.

Модель інкапсулює ядро даних і основний функціонал їхньої обробки і не залежить від процесу вводу чи виводу даних.

Вигляд може мати декілька взаємопов'язаних областей, наприклад різні таблиці і поля форм, в яких відображаються дані.

Архітектура клієнт-сервер є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних додатків і передбачає взаємодію та обмін даними між ними [15].

Сервери є незалежними один від одного. Клієнти також функціонують паралельно і незалежно один від одного. Немає жорсткої прив'язки клієнтів до серверів. Більш ніж типовою є ситуація, коли один сервер одночасно обробляє запити від різних клієнтів; з іншого боку, клієнт може звертатися то до одного сервера, то до іншого. Клієнти мають знати про доступні сервери, але можуть не мати жодного уявлення про існування інших клієнтів.

Дуже важливо ясно уявляти, хто або що розглядається як «клієнт». Можна говорити про клієнтський комп'ютер, з якого відбувається звернення до інших комп'ютерів. Можна говорити про клієнтське та серверне програмне забезпечення. Нарешті, можна говорити про людей, які бажають за допомогою відповідного програмного та апаратного забезпечення отримати доступ до тієї чи іншої інформації.

Загальноприйнятим є положення, що клієнти та сервери — це перш за все програмні модулі. Найчастіше вони знаходяться на різних комп'ютерах, але бувають ситуації, коли обидві програми — і клієнтська, і серверна, фізично розміщуються на одній машині; в такій ситуації сервер часто називається локальним.

Модель клієнт-серверної взаємодії визначається перш за все розподілом обов'язків між клієнтом та сервером. Логічно можна відокремити три рівні операцій: рівень представлення даних, який по суті являє собою інтерфейс користувача і відповідає за представлення даних користувачеві і введення від нього керуючих команд; прикладний рівень, який реалізує основну логіку застосунку і на якому здійснюється необхідна обробка інформації; рівень управління даними, який забезпечує зберігання даних та доступ до них.

Веб клієнт був побудований за допомогою сучасних технологій HTML5/CCS3, JavaScript, а також з використанням фреймворків React та Redux та бібліотеки Bootstrap.

JavaScript – динамічна, об’єктно-орієнтовна мова програмування. Представляє собою стандарт ECMAScript. Найчастіше використовується як частина браузера, що надає можливість коду на стороні клієнта (такому, що

виконується на пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки. JavaScript класифікують як прототипну скриптову мову програмування з динамічною типізацією.

React.js, здебільшого називають React — відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка була створена для вирішування проблеми часткового оновлення вмісту веб-сторінки, з якими стикаються в розробці односторінкових додатків. Розробляється Facebook, Instagram і спільнотою індивідуальних розробників.

React дозволяє розробникам створювати великі веб додатки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки.

Його мета полягає в тому, щоб бути швидким, простим, масштабованим. React обробляє тільки користувацький інтерфейс у додатка. Це відповідає видові у шаблоні модель-вид-контролер (MVC), і може бути використане у поєднанні з іншими JavaScript бібліотеками або в великих фреймворках MVC, таких як AngularJS. Він також може бути використаний з React на основі надбудов, щоб піклуватися про частини без користувацького інтерфейсу побудови веб-застосунків.

В даний час React використовують Khan Academy, Netflix, Yahoo, Airbnb, Sony, Atlassian та інші.

Twitter Bootstrap був обраний як фронтенд фреймворк. Twitter Bootstrap – це набір інструментів від Twitter (відноситься до класу інструментів: CSS-фреймворк), створений для полегшення розробки веб-застосунків та сайтів. Він включає CSS та HTML для типографії, форм, кнопок, таблиць, сіток, навігації тощо, а також додаткові розширення JavaScript [16].

Основними перевагами Twitter Bootstrap 3 є наступне: економія часу розробки; масштабованість на різноманітні пристрої та роздільну здатність екрану без жодних змін у розмітці сторінки; усі компоненти платформи використовують єдиний стиль та шаблон за допомогою центральної бібліотеки, що дозволяє

узгоджувати дизайн сторінок; простота у використанні фреймворку; сумісність з усіма популярними браузерами.

Серверна частина додатку була розроблена за допомогою Node.js з використанням фреймворку express. Node.js був обраний у зв’язку з тим, що розробка як backend частини, так і frontend ведеться однією мовою програмування, та дані між частинами додатку передаються у JSON – форматі, що є нативним для JavaScript.

Node.js – платформа з відкритим кодом для написання серверної частини веб-реалізації на JavaScript, автором якої є Раян Дал. Node.js характеризується такими властивостями, як асинхронна однопотокова модель виконання запитів, неблокуючий ввід/вивід, система модулів CommonJS, JavaScript Google V8 [17].

Для управління модулями використовується пакетний менеджер npm. Node.js призначений для відокремленого виконання високопродуктивних мережевих застосунків на мові JavaScript. Функції платформи не обмежені створенням серверних скриптів для веб, платформа може використовуватися і для створення звичайних клієнтських і серверних мережевих програм. Для забезпечення обробки великої кількості паралельних запитів Node.js задіює асинхронну модель запуску коду, засновану на обробці подій в неблокуючому режимі і визначенні обробників зворотніх викликів.

Node.js характеризується такими властивостями:

– асинхронна однопотокова модель виконання запитів;
– неблокуючий ввід/вивід;
– система модулів CommonJS;
– рушій JavaScript Google V8.

Для управління модулями використовується пакетний менеджер npm (node package manager).

Для забезпечення обробки великої кількості паралельних запитів у Node.js використовується асинхронна модель запуску коду, заснована на обробці подій в неблокуючому режимі та визначенні обробників зворотніх викликів (callback). Як способи мультиплексування з'єднань підтримується epoll, kqueue, /dev/poll і select.

Для мультиплексування з'єднань використовується бібліотека libuv, для створення пулу потоків (thread pool) задіяна бібліотека libeio, для виконання DNS-запитів у неблокуючому режимі інтегрований c-ares. Всі системні виклики, що спричиняють блокування, виконуються всередині пула потоків і потім, як і обробники сигналів, передають результат своєї роботи назад через неіменовані канали (pipe).

Хоча Node.js особливо гарний у контексті додатків, що працюють в реальному часі, він цілком підходить і для надання інформації з об'єктних баз даних (наприклад MongoDB). Дані, збережені у форматі JSON, дозволяють Node.js функціонувати без втрати відповідності та без перетворення даних.

Наприклад, якщо ви використовуєте Rails, то вам довелося б перетворювати JSON в двійкові моделі, а потім знову надавати їх у вигляді JSON по HTTP, коли дані будуть споживатися Backbone.js, Angular.js або навіть звичайними викликами jQuery, AJAX. Працюючи з Node.js можна просто надавати ваші об'єкти JSON клієнту через REST API, щоб клієнт обробляв їх. Крім того, не доводиться турбуватися про перетворення між JSON і чим завгодно ще при зчитуванні бази даних і запису в неї (якщо ви використовуєте MongoDB). Отже, ви обходитеся без безлічі перетворень, використовуючи універсальний формат серіалізації даних, що застосовується і на клієнтові і на сервері, і в базі даних.

MongoDB — документо-орієнтована система управління базами даних (СУБД) з відкритим кодом, яка не потребує опису схеми таблиць. Займає нішу між швидкими і масштабованими системами, що оперують даними у форматі ключ/значення, і реляційними СУБД, функціональними і зручними у формуванні запитів.

Код MongoDB написаний на мові C++ і поширюється в рамках ліцензії AGPLv3. MongoDB підтримує зберігання документів в JSON-подібному форматі, має досить гнучку мову для формування запитів, може створювати індекси для різних збережених атрибутів, ефективно забезпечує зберігання великих бінарних об'єктів, підтримує журналювання операцій зі зміни і додавання даних в БД, може працювати відповідно до парадигми Map/Reduce, підтримує реплікацію і побудову відмовостійких конфігурацій.

При розробці автори виходили з необхідності спеціалізації баз даних, завдяки чому їм вдалося відійти від принципу «один розмір під усе». За рахунок мінімізації семантики для роботи з транзакціями з'являється можливість вирішення цілого ряду проблем, пов'язаних з нестачею продуктивності, причому горизонтальне масштабування стає простішим.

Використовувана модель документів зберігання даних (JSON/BSON) простіше кодується, простіше управляється (у тому числі за рахунок застосування так званого безсхемного стилю (англ. schemaless style), а внутрішнє угруповання релевантних даних забезпечує додатковий виграш в швидкодії. Нереляційний підхід досить зручний для створення баз даних, у яких горизонтальне масштабування означає розгортання на множині машин. Можливість забезпечувати найкращу продуктивність повинна існувати паралельно з підтримкою більшої функціональності, ніж це дозволяє використання пар «ключ-значення» (у чистому вигляді). Технологія баз даних має працювати скрізь, починаючи з серверів користувача та віртуальних машин і закінчуючи хмарними технологіями.

MongoDB, на думку розробників, має заповнити розрив між простими сховищами даних типу «ключ-значення» (швидкими і легко масштабованими) і великими РСКБД (зі структурними схемами і потужними запитами). У MongoDB є вбудовані засоби із забезпечення шардінга (розподіл набору даних по серверах на основі певного ключа), комбінуючи який реплікацією даних можна побудувати горизонтально масштабований кластер зберігання, в якому відсутня єдина точка відмови (збій будь-якого вузла не позначається на роботі БД), підтримується автоматичне відновлення після збою і перенесення навантаження з вузла, який вийшов з ладу. Розширення кластера або перетворення одного сервера в кластер проводиться без зупинки роботи БД простим додаванням нових машин.

У результаті аналізу потреб, поставлених задач, вибору інструментів та виділення функціональних частин веб-додатку була отримана архітектура системи, що зображена на рисунку 4.4.

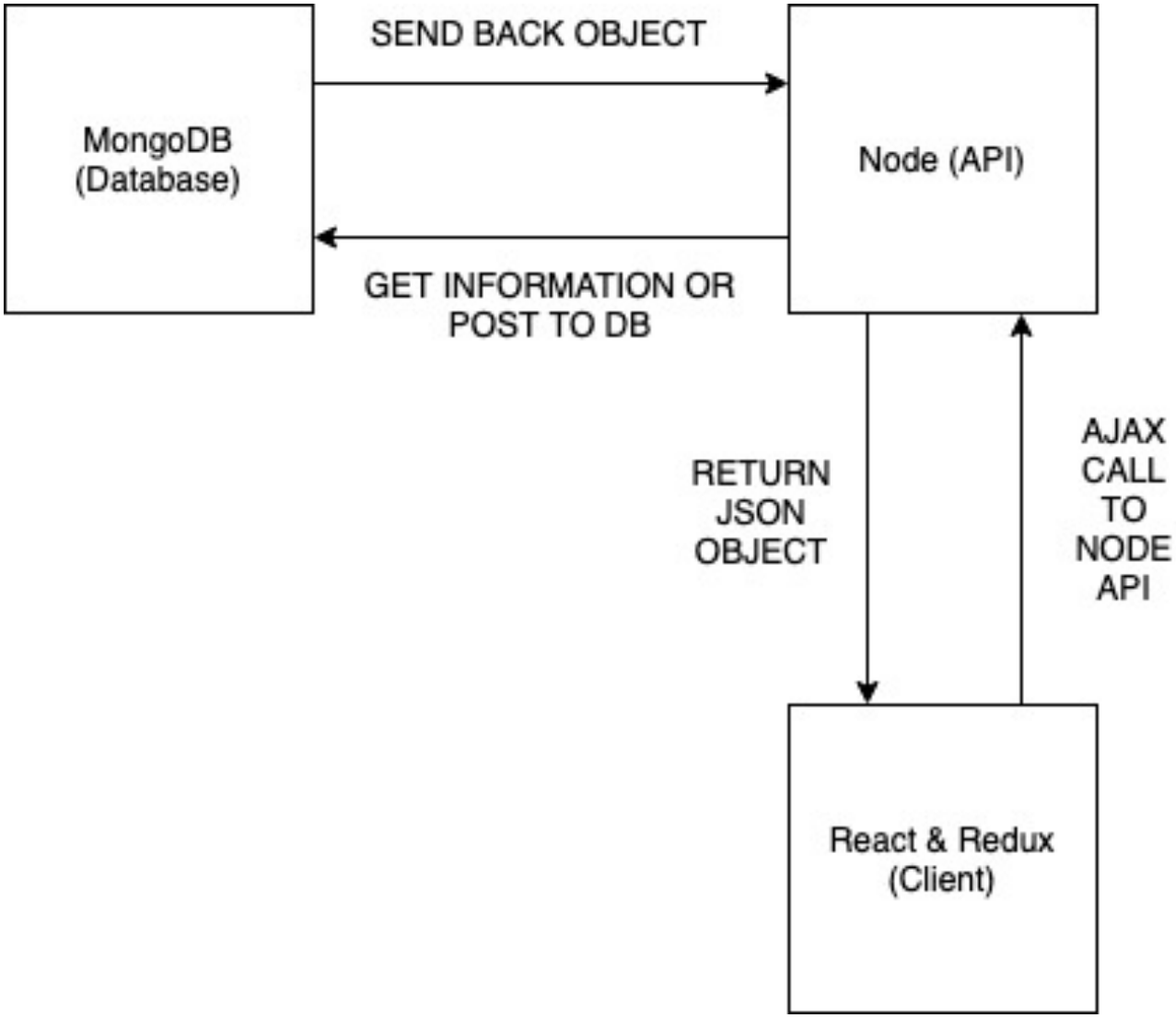

Рисунок 4.4 – Архітектура додатку

Завдяки використанню адаптивного веб–дизайну, забезпечується коректне відображення веб-сторінок на різних пристроях. Метою адаптивного веб-дизайну є універсальність веб-сайту для різних пристроїв. Для того, щоб веб-сайт було зручно переглядати з пристроїв різних роздільних здатностей та форматів, за технологією адаптивного веб-дизайну не потрібно створювати окремі версії веб-сайту для окремих видів пристроїв. Один сайт може працювати на смартфоні, планшеті, ноутбуці та телевізорі з виходом в інтернет, тобто на всьому спектрі пристроїв.

Git був використаний для контрою версій під час розробки додатку. Git – розподілена система керування версіями файлів та спільної роботи [18]. Git є однією з найефективніших, надійних і високопродуктивних систем керування

версіями, що надає гнучкі засоби нелінійної розробки, що базуються на відгалуженні і злитті гілок. Для забезпечення цілісності історії та стійкості до змін заднім числом використовуються криптографічні методи, також можлива прив'язка цифрових підписів розробників до тегів і комітів.

Інструмент «Система керування робочим часом» призначений, для того щоб допомагати ефективно розподілити свій час, застосовуючи вже існуючі методики, та надати можливість менеджеру провести аналіз використання робочого часу, що дозволить зменшити ризики критичних моментів на проекті.

Інтерфейс для додатку «Система керування робочим часом» створено з використанням технології RWD (респонсів веб-дизайн), дизайну веб-сторінок, що забезпечує оптимальне відображення та взаємодію сайту з користувачем незалежно від роздільної здатності та формату пристрою, з якого здійснюється перегляд сторінки.

Метою респонсів веб-дизайну є практичне відображення інформації та зручна навігація на всіх пристроях із доступом до інтернету. За технологією респонсів веб-дизайну не потрібно створювати окремі версії веб-сайту [19].

Даний принцип базується на використанні медіа-запросів в css (каскадні таблиці стилів).

Для створення інтерфейсу було використано сучасний стек технологій: HTML5, CSS3 та JavaScript з використанням бібліотеки jQuery.

Ця бібліотека є бібліотекою JavaScript та фокусується на взаємодії JavaScript і HTML. Бібліотека jQuery допомагає легко отримувати доступ до будь-якого елементу DOM, звертатися до атрибутів і вмісту елементів DOM, маніпулювати ними.

Використання цих інструментів дало можливість створити зручний та зрозумілий інтерфейс користувача.

Дизайн сайту був розроблений враховуючі останні тенденції: великі блоки, яскравий контент, динамічне довантаження блоків.

4.3 Опис інтерфейсу користувача

Базовою функціональністю системи є надання можливості зареєстрованому користувачеві переглядати власний розклад, додавати задачі, проводити моніторинг витраченого на задачу часу за методом Помадоро, пріоритезувати задачі за матрицею Ейзенхауера.

Після відкриття сайту користувач потрапляє на головну сторінку сайту зображену на рисунку 4.5, на якій представлено дві форми.

Форма реєстрації, для нових користувачів, та форма входу, для користувачів, які вже мають акаунт.

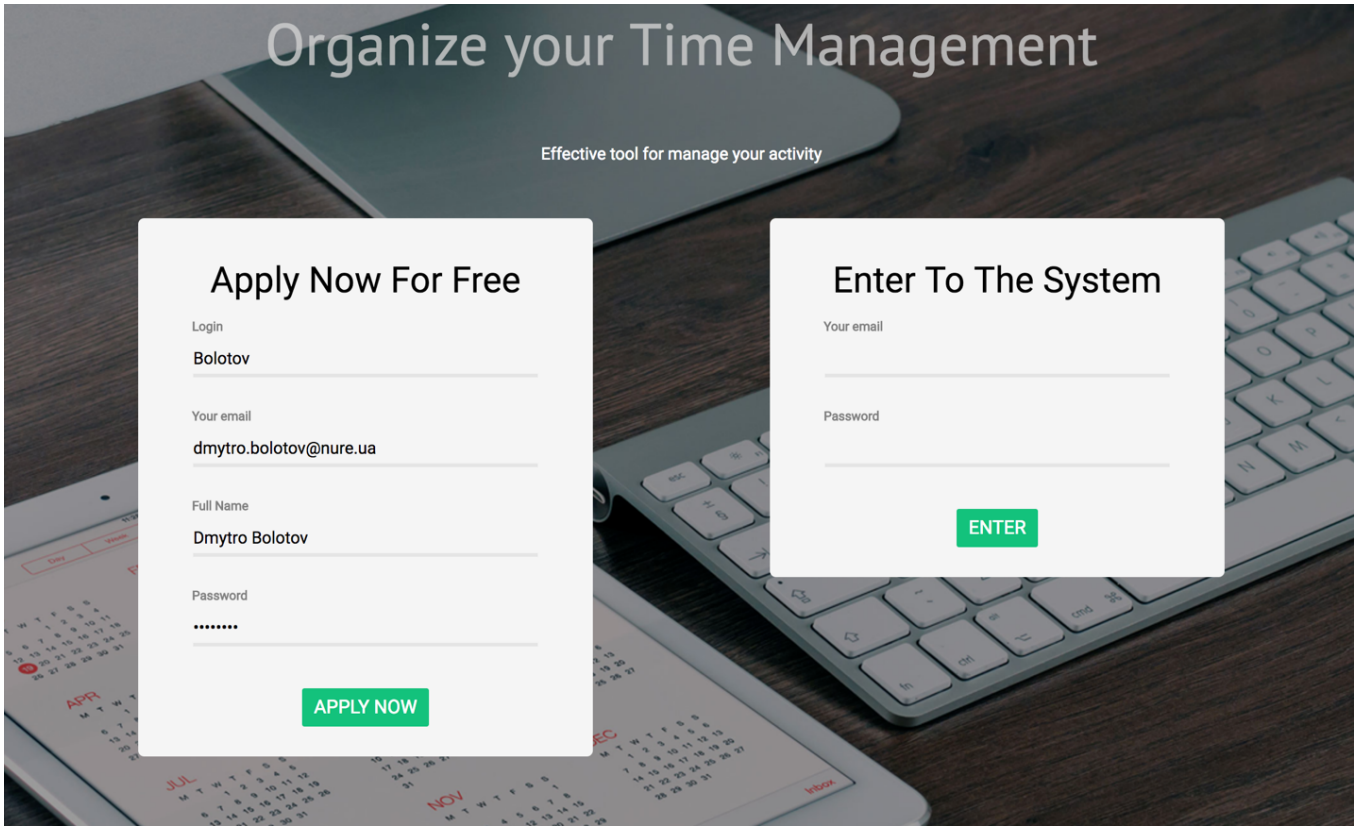

Рисунок 4.5 – Головна сторінка сайту

Після авторизації користувач потрапляє на сторінку, з персональним розкладом. Інтерфейс систем складається з таких основних компонентів як навігаційне меню, та інформативна частина системи, де відображається інформація згідно з обраним розділом. У навігаційному меню представлені такі пункти:

– персональна інформація;

– список задач;

– пріоритезація;

– моніторинг активності.

Розглянемо детальніше сторінку, на котру користувач потрапляє одразу після авторизації, а саме сторінку зі списком задач, пункт навігаційного меню Task List. Приклад розкладу користувача наведений на рисунку 4.6.

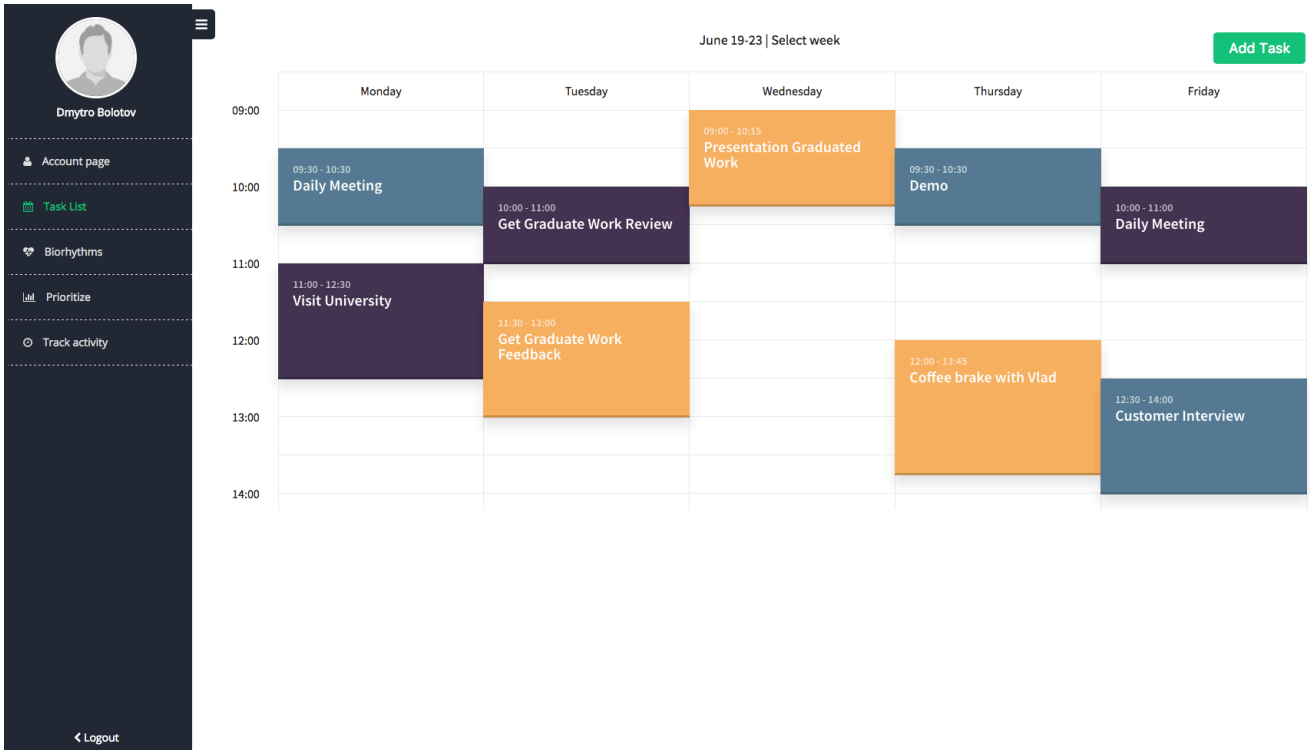

Рисунок 4.6 – Приклад персонального розкладу

В залежності від пріоритету задач, виходячи з принципу розрахунку згідно правил матриці Ейзенхауера, значення яких введені користувачем на етапі створення задачі, вони позначені різним кольором. Задачі розташовані у календарі згідно з заданими часовими межами.

Для того, що переглянути ті самі задачі у вигляді пріоритезованого за матрицею Ейзенхауера списку, можна перейти на сторінку Prioritize що зображена на рисунку 4.7.

Задачі розбиті на чотири основні розділи, згідно з підходом, що пропонує матриця Ейзенхауера. До кожної задачі на моменті створення було додано мітки,

важливо чи не важливо, терміново чи не терміново. В результаті, у кінцевому списку задачі розставлені згідно пріоритетів, тобто до їх виконання слідує підходити у тому порядку, у якому вони розташовані (в рамках одного блоку послідовність може змінюватись).

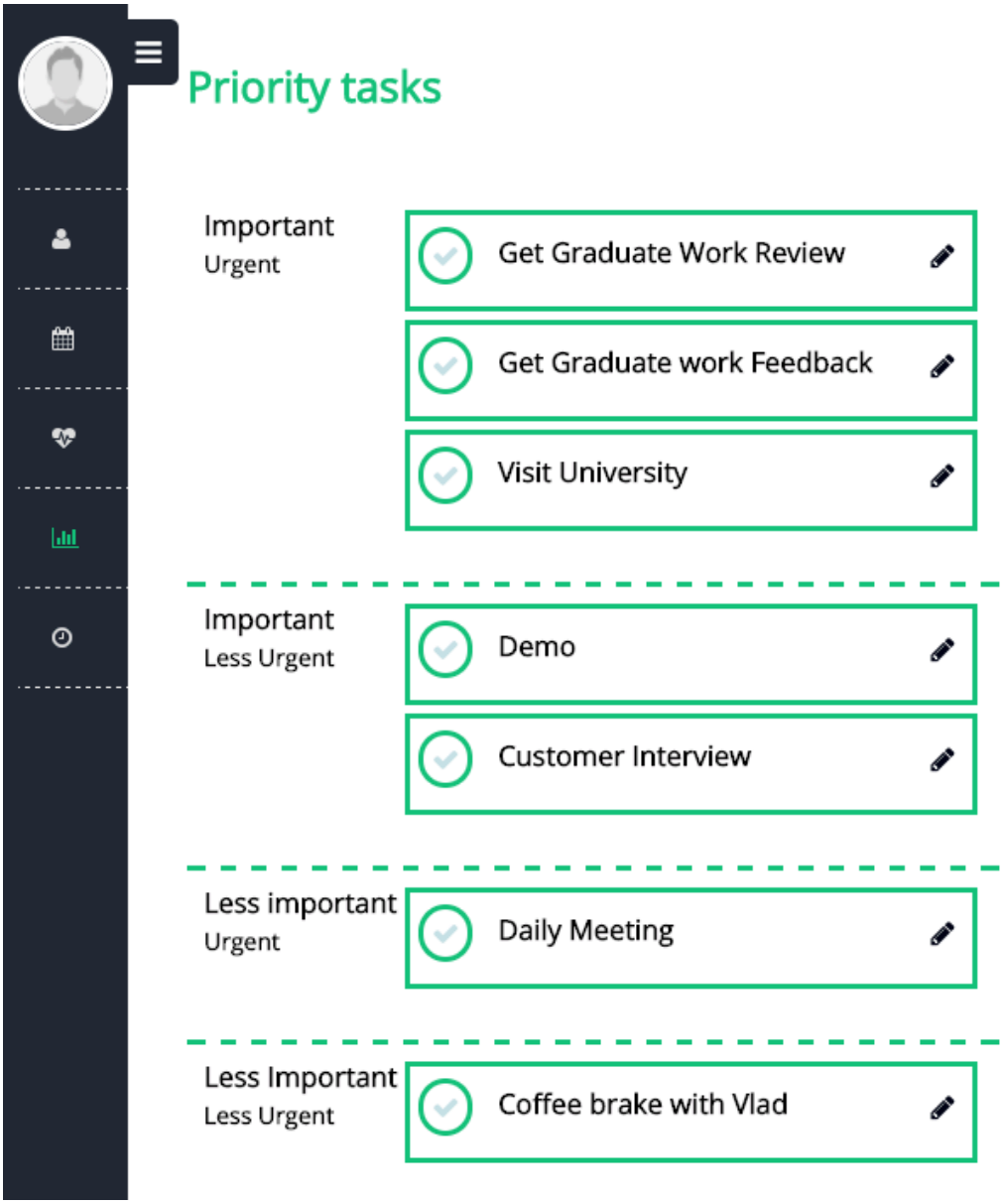

Рисунок 4.7 – Приклад пріоритезованих задач

Використовуючи такий метод пріоритезації можна запобігти створенню критичних ситуацій.

Кожну з задач можна легко перевести в стан виконаних просто поставивши відмітку у відповідному полі. Іконка олівця дозволяє перейти у режим редагування задачі, адже можлива ситуація, коли на етапі створення користувач помилився.

Доречі, щоб продемонструвати зовнішній вигляд системи на мобільних платформах, сторінка була представлена у інтерфейсі під мобільні платформи.

Одразу видно, що навігаційне меню стало займати значно менше простору, та при бажанні можна натиснути на відповідну іконку, й вибрати необхідний розділ, у разі, якщо користувач ще не досить гарно орієнтується в системі.

Також користувач має можливість провести моніторинг використання часу на певні задачі як показано на рисунку 4.8.

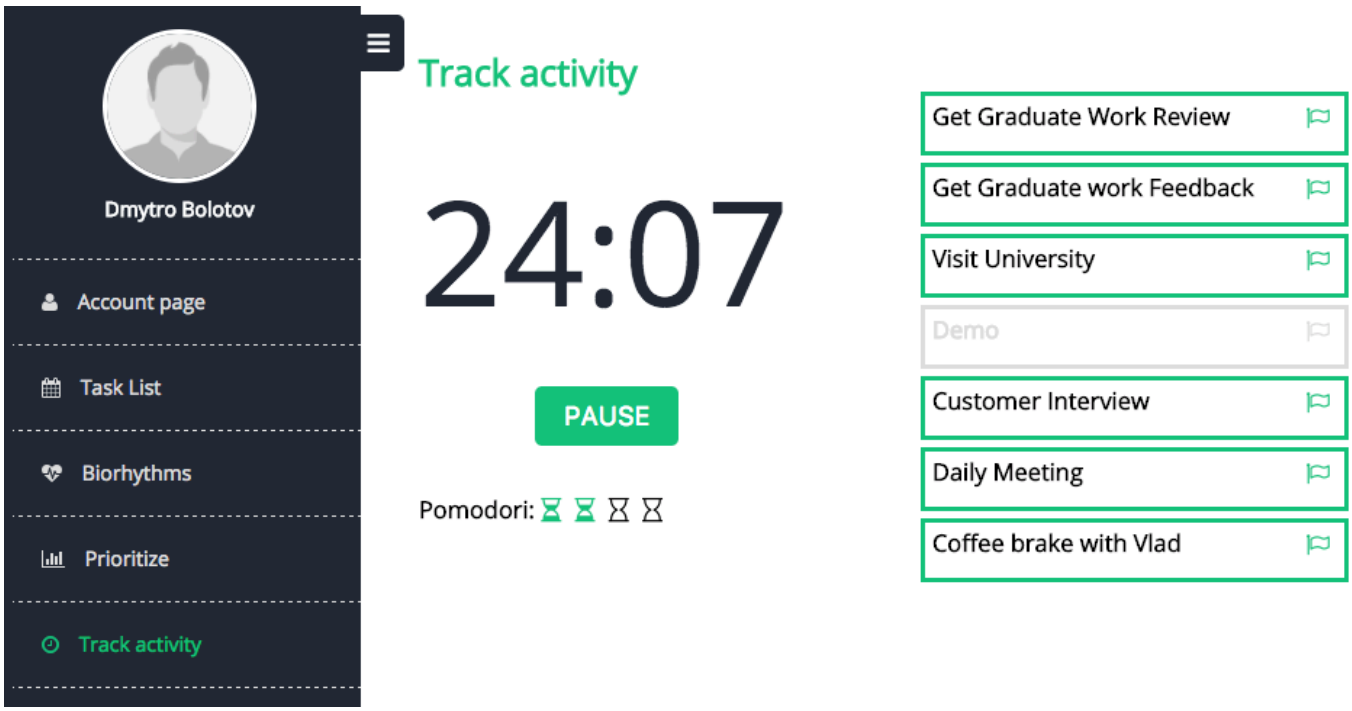

Рисунок 4.8 – Моніторинг витрат часу за системою Pomodor

Користувачеві необхідно вибрати задачу, для якої заповнена кількість Pomodor. Та розпочати відлік.

З кожним пройденим інтервалом користувач бачить затрачений на задачу час, та може зробити висновки що до ефективності роботи.

4.4 Тестування системи

У ході роботи було проведено тестування основних функцій системи.

Тестування програмного забезпечення – перевірка відповідності між реальним і очікуваним поведінкою програми, здійснювана на кінцевому наборі тестів, обраному певним чином.

Усі тести діляться на чотири групи: модульне тестування, інтеграційне тестування, тестування компонентів інтерфейсу, і системне тестування. Основні рівні в процесі розробки, як визначено в SWEBOK є модульне, інтеграційне і системне тестування, які відрізняються метою тестування, не маючи на увазі певну модель процесу.

Модульне тестування – відноситься до тестів, які перевіряють функціональність певного розділу коду, зазвичай на функціональному рівні. В об'єктно-орієнтованому середовищі, це, як правило, тестування на рівні класу, а мінімальні модульні тести містять у собі конструктори та деструктори.

Модульне тестування (юніт тестування) – полягає в ізольованій перевірці кожного окремого елемента шляхом запуску тестів в штучному середовищі. Модульне тестування спрямоване на усунення помилок проектування. Ця стратегія спрямована на підвищення якості одержуваного ПЗ, до такого рівня, як вимагає процес контролю якості.

Інтеграційне тестування є типом тестування ПЗ, яке прагне перевірити інтерфейси між компонентами від програмного дизайну. Програмні компоненти можуть бути інтегровані як в рамках ітеративного підходу, так і всі разом. Інтеграційне тестування працює над виявленням дефектів у інтерфейсах та взаємодії інтегрованих компонентів (модулів).

Системне тестування – тестує інтегровану систему для перевірки відповідності всім вимогам. Системне інтеграційне тестування перевіряє, чи система інтегрується в будь-яку зовнішню систему (або системи) відповідно до системних вимог. Основним завданням системного тестування є перевірка як

функціональних, так і не функціональних вимог до системи в цілому. При цьому виявляються дефекти, такі як невірне використання ресурсів системи, непередбачені комбінації даних користувача рівня, несумісність з оточенням, непередбачені сценарії використання, відсутня або невірна функціональність, незручність використання.

Тестування даної програмної системи вимагає комплексного підходу як з боку зовнішнього (мануального) тестування, тестування валідної роботи REST API так впровадження юніт та інтеграційних тестів з боку серверної частини. Була виконана перевірка відповідності між реальним і очікуваним поведінкою програми, що здійснюється на кінцевому наборі тестів, обраному певним чином в найбільш проблемних місцях системи.

Тестування програмного забезпечення охоплює цілий ряд видів діяльності, вельми аналогічний послідовності процесів розробки програмного забезпечення. Сюди входять постановка задачі для тесту, проектування, написання тестів, тестування тестів і, нарешті, виконання тестів і вивчення результатів тестування. Вирішальну роль відіграє проектування тесту [20].

Для підвищення ефективності випробування та його прискорення необхідно розробити науково обґрунтовані методи, засоби і методики, що дозволяють подолати недоліки підходу до тестування, недооцінку його ролі в забезпеченні необхідного рівня якості продукту. Ця мета може бути досягнута лише шляхом розробки технологічної схеми випробувань, що передбачає:

- розуміння, в яких умовах буде працювати система і вимоги до неї з боку користувачів;
- автоматизацію всіх найбільш трудомістких процесів і насамперед моделювання середовища функціонування;
- чітке уявлення мети і послідовності випробування;
- цілеспрямованість і ненадмірність випробування, що виключають або мінімізують повторення однорідних процедур при одних і тих же умовах функціонування системи;

– систематичний контроль за ходом, регулярне ведення протоколу та журналу випробування;
– чітке, послідовне визначення і виконання плану випробування;
– зіставлення наявних ресурсів з передбачуваним обсягом випробування.

Можливість забезпечення, а також об'єктивної кількісної оцінки повноти та достовірності результатів випробування на всіх етапах.

В результаті поставлених критеріїв тестування якості інформаційної системи було сформовано ряд вимог до найбільш критично важливих функцій задля оптимально сприйнятої роботи системи, а також можливість подальшого автоматизованого покриття тестами різних компонентів, що матимуть важливе значення в подальшому. Саме тому в першу чергу під час розробки програмного коду для закріплення результатів його написання були створені юніт-тести, щоб в ході розробки не втрачати якість роботи готових до застосування компонентів.

Це являє собою Модульне тестування, яке полягає в окремому тестуванні кожного модуля коду програми. Модулем називають найменшу частину програми, яка може бути протестованою. В об'єктно-орієнтованому програмуванні — інтерфейс, клас. Їх зазвичай застосовують для того, щоб упевнитися, що код відповідає вимогам архітектури та має очікувану поведінку. Саме тому новий функціонал не буде псувати написаний раніше.

Розроблена система має підтримувати наступні основні функції:

– реєстрація користувача у системі;
– можливість додання нових задач;
– можливість вибору категорій (meeting, task), пріоритету, дати виконання задачі;
– пріоритезація задач за мітками матриці Ейзенхауера (терміново/не терміново), (важливо/не важливо);
– можливість приступити к виконанню задачі, для якої задана кількість часу у pomador (задаються на етапі створення задачі);
– перегляд задач у різних видах, як календар, та список;
– можливість перевести задачу у стан виконаної;

– пошук за фільтрами;

– бали продуктивності (нараховуються за кожну виконану за день справу).

Було проведене як автоматизоване, так і ручне тестування системи. Для написання тестів використовувалася бібліотека Jest. Вручну було протестовано робочі області програмної системи.

Ручне тестування – це частина процесу тестування на етапі контролю якості в процесі розробки програмного забезпечення. Воно проводиться тестувальником без використання програмних засобів, для перевірки програми або сайту шляхом моделювання дій користувача.

У ролі тестувальників можуть виступати і звичайні користувачі, повідомляючи розробникам про знайдені помилки. В кінцевому рахунку також було проведене мануальне тестування для перевірки коректної взаємодії дизайну системи з серверними відповідями та даними, що мігрують поміж ними. В даному процесі тестування були усунені критичні помилки виконання та знайдені існуючі для їх подальшого усунення.

Тестування було проведено за допомогою динамічного та статичного підходів. Динамічний підхід включає в себе запуск програмної системи, статичний підхід перевірку синтаксису коду програми.

У ході тестування було виявлено, що система працює коректно та відповідно до постановки задачі. Таким чином, у ході проведеного тестування було встановлено, що всі основні функції системи працюють коректно.

5 АНАЛІЗ МОЖЛИВИХ ЗАСТОСУВАНЬ ТА УДОСКОНАЛЕНЬ

Розроблений додаток може стати корисним інструментом у роботі менеджера проекту, і допомогти оптимізувати такі процеси у керуванні проектами, що відносяться до планування об’єму робіт, проведення оцінки задач, зменшити ризик провалу строків.

Існує достатньо велика кількість напрямків для майбутнього розвитку додатку, реалізуючи які можна модернізувати розроблену систему. Одним з таких напрямків є створення окремого розділу додатку, де менеджер міг би отримати звіт у форматі графіків що до ефективності працівника, виявити членів команди що мають низький рівень ефективності та задачі що вимагають більше часу ніж було закладено на етапі планування. Створення сторінки зі статистикою виконаних задач та отриманих балів, що відображало би загальний прогрес для проекту.

Також важливим кроком стала би інтеграція з сервісами, які мають відкритий API, наприклад Outlook та Google Calendar, JIRA та Trello.

Ще одним напрямком для розширення може стати реалізація інших методів оптимізації процесів керування проектами.

Також у майбутньому існує можливість розвитку продукту у інших напрямах, по перше, можливе створення власного алгоритму, що відповідатиме за попередню оцінку задач беручи за основу попередні оцінки схожих за типом завдань. Такий крок у процесі керування проектами допоміг би знизити час що необхідно витрачати на оцінку задач, і вимагав би лише перегляду оцінених завдань спеціалістами. Другий варіант це зосередитись на створенні прошарку, для аналізу даних які ми отримуємо від стороннього сервісу, і третій варіант, це тісна інтеграція з іншими схожими за функціоналом додатками. Кожен з варіантів має право на життя і є як плюси так і мінуси того чи іншого підходу

ВИСНОВКИ

У ході атестаційної роботи було досліджено методи що спрямовані на оптимізацію систем керування робочим процесом та саме процеси у керуванні проектами.

На початку роботи було проведено аналіз предметної області, в процесі якого були визначені основні проблеми, що існують в розглянутій галузі. Проаналізувавши існуючі підходи для керування проектами та основні проблеми що існують у цій галузі виявлено що незалежно від підходу фігурує час. Саме управління часом було обрано як область для оптимізації.

Було проведено аналіз існуючих аналогів, які є популярними на ринку, та призначені для організації робочого часу, формування списків задач та створення розкладів, і виявлено їх основні можливості та недоліки. На підставі аналізу предметної області була проведена постановка завдання.

Беручи за основу такі методи оптимізації в управлінні часом як матриця Ейзенхауера для визначення пріоритетів та метод Помадоро для контролю витрат робочого часу, було створено додаток, що може стати корисним інструментом у роботі проектного менеджера, та допомогти підвищити ефективність команди, та запобігти критичним моментам на проекті.

Результатом дослідження стала розробка системи для управління проектом застосовуючи методи оптимізації керування проектами.

У ході виконання поставленої задачі було:

– проведено аналіз, концептуальне та UML-моделювання предметної галузі;
– розроблений алгоритми отримання нотифікацій з рекомендаціями;
– виконано програмну реалізацію веб-системи;
– проведено тестування розробленого програмного продукту.

Розроблений продукт можна використовувати у сфері менеджменту проектів розробки програмного забезпечення. Контроль за виконанням задач надає

можливість більш точно у майбутньому робити оцінку задач що сприяє зниженню ризиків і сприяє успішності проекту.

Застосування створеного додатку на практиці може допомогти знизити ризики, що в свою чергу покриває також фінансові ризики. Адже досить часто саме проблеми в оцінках часу необхідних на реалізацію проекту становлять фінансову загрозу успішності проекту.

Розроблений продукт повністю відповідає вимогам, що були поставлені раніше, та представляє доступ до методів оптимізації процесу керування проектами.

У той же час система потребує удосконалень. Серед них можна виділити такі напрямки як розширення функціоналу системи в основі якої можуть бути представлені інші методи оптимізації систем керування робочим процесом, підтримка мобільних пристроїв шляхом створення нативних додатків та оптимізації вже реалізованих рішень. Ще одним напрямком для розширення може стати створення окремого розділу додатку, де статистика отримана для кожного конкретного члена команди допоможе визначити ефективність, та врахувати помили допущенні на етапі планування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Портни, С. Управление проектами для чайников. – М.: Диалектика, 2006. – 368 с.

2. Ньюэлл, М. Управление проектами для профессионалов. Руководство по подготовке к сдаче сертифицированного экзамена. – М.: Кудиц-пресс, 2008. – 416 с.

3. ДеМарко, Т. Deadline. Роман об управлении проектами. – М.: Вершина, 2006. – 143 с.

4. Лапыгин, Ю. Управление проектами: от планирования до оценки эффективности. – М.: Омега-Л, 2008. – 252 с.

5. Вудок, М. Эффективное управление. – М.: Дело, 2012. – 378 с.

6. Моргенстерн, Д. Тайм-менеджмент. Искусство планирования и управления своим временем и своей жизнью. – М.: Добрая книга, 2014. – 112 с.

7. Берд, П. Тайм-менджмент. Планирование и контроль времени. – М.: ФАИР-ПРЕСС, 2011. – 288 с.

8. Халан, И. Управление временем. – М.: Издательство ДИЛЯ, 2014. – 96 с.

9. Никерсон, П. Ловушка. – М.: Альпина Паблишер, 2015. – 205 с.

10. Мартин, Р. Чистный код. Создание, анализ и рефактроринг. – М.: Питер, 2010. – 464 с.

11. Фаулер, М., Дейвид, Р. Архитектура корпоративных программных приложений. – М.: Вильямс, 2008. – 544 с.

12. Фаулер, М. UML. Основи. – СПб: Символ-Плюс., 2004. – 192 с.

13. Діаграма варіантів використання (use case diagram) [Електронний ресурс] / UML Теорія. – URL: http://www.info-system.ru/designing/methodology /uml/theory/use_case_diagram_theory.html (дата звернення: 22.04.2019).

14. Трьохрівнева архитектура [Електронний ресурс]/ Вікіпедія – URL: https://ru.wikipedia.org/wiki/Трёхуровневая_архитектура (дата звернення: 11.05.2019).

15. Архитектура REST [Електронний ресурс] / Хабрахабр. – URL: https://habrahabr.ru/post/38730 (дата звернення: 15.05.2019).

16. Duckett, J. HTML and CSS: Design and Build Websites. 1st Edition. – John Wiley & Sons, 2011. – 490 p.

17. Документація Nodejs [Електронний ресурс] / Офіційний сайт проекту Nodejs. – URL: https://wwwnodejs.org/ (дата звернення: 25.05.2019).

18. Скотт, Ч. Профессиональный Git. – М.: «Символ-плюс», 2015. – 1012 с.

19. Colborne, G. Simple and Usable Web, Mobile, and Interaction Design (Voices That Matter) 1st Edition. – New Riders, 2010. – 208 p.

20. Винниченко, И. Автоматизация процессов тестирования. – СПБ.: Питер, 2005. – 203 с.