Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інтелектуальних інформаційних технологій та автоматизації
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна технологія для менеджменту стартапів»

Виконав: студент 2-го курсу, групи 1КН-21м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
Машницький Є.М.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
Арсенюк І.Р.
(прізвище та ініціали)
«15» 12 2022 р.

Опонент: к.т.н., доцент каф. АІІТ
Кабачій В.В.
(прізвище та ініціали)
«15» 12 2022 р.

Допущено до захисту
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.
(прізвище та ініціали)
«16» 12 2022 р.

Вінниця ВНТУ - 2022 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації Кафедра комп'ютерних наук
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 122 «Комп'ютерні науки»
Освітньо-професійна програма – «Системи штучного інтелекту»

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
Д.т.н., проф. Яровий А.А.
14.09 **2022 року**

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Машницький Євгеній Петрович
(прізвище, ім'я, по батькові)

1. Тема роботи інформаційна технологія для менеджменту стартапів

керівник роботи к.т.н., доцент кафедри КН Арсенюк І.Р.
затверджені наказом вищого навчального закладу від "14" 09 2022 року №203
2. Строк подання студентом роботи 18 листопада 2022 року

3. Вихідні дані до роботи:
Одночасна онлайн участь у редагуванні інформації по проекту – до 100 людей; тривалість проекту не менше 4 тижнів; кількість працівників у проекті не менше п'яти; наявність відкритого API; наявність multitenensi парадигми; підтримка кросплатформності; мова програмування – об`єктно орієнтована.
4. Зміст текстової частини:
Вступ, аналіз предметної області менеджменту стартапів, розробка інформаційної технології менеджменту стартапів, програмна реалізація інформаційної технології менеджменту стартапів, економічна частина, висновки, перелік використаних джерел, додатки.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень)
Алгоритм роботи програми менеджменту стартапів, UML діаграма класів програми менеджменту стартапів, робочі вікна програми менеджменту стартапів, результати тестування менеджменту стартапів.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
1-3	Арсенюк ІР., к.т.н., доц. каф. КН	14.09.2022	12.2022
4	Нікіфорова Л. О., к. е. н., доц. каф. ЕПВМ	19.09.2022	12.12.2022

7. Дата видачі завдання 14.09 2022 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	При-мітка
1	Аналіз сучасного стану розвитку інформаційних технологій управління стартапом	14.09.2022 - -1.10.2022р	
2	Розробка методу та інформаційної технології управління стартапом	2.10.2022р - -1610.2022р.	
3	Програмна реалізація розробленої інформаційної технології, тестування та оцінка параметрів	17.10.2022р - -07.11.2022р	
4	Підготовка економічної частини	08.11.2022р - -21.11.2022р	
5	Апробація та/або впровадження результатів дослідження	23.11.2022р - 01.12.2022р.	
6	Оформлення пояснювальної записки, графічного матеріалу та презентації	02.12.2022р - 14.12.2022р.	

Студент _____ (підпис) Машницький Є.П.

Керівник роботи _____ (підпис) Арсенюк І.Р.

АНОТАЦІЯ

УДК 004.8

Машницький Є.П. Інформаційна технологія управління стартапом. Магістерська кваліфікаційна робота зі спеціальності 122 – комп'ютерні науки, освітня програма - комп'ютерні науки. Вінниця: ВНТУ, 2022. 88 с.

На укр. мові. Бібліогр.: 17 назв; рис.: 26; табл. 7.

Магістерська кваліфікаційна робота присвячена розробці інформаційної технології для менеджменту стартапів. Оглянуто та проаналізовано існуючі аналоги для менеджменту стартапів, обгрунтовано доцільність створення відповідної технології з додатковими функціональними можливостями.

Модефіковано існуючу математичну модель та розроблено алгоритм для системи менеджменту стартапами, обрано експертну систему та інтегровано у систему. Обґрунтовано мову програмування та середовище розробки даної системи. Створено веб-застосунок на мові C# та JavaScript для управління стартапом.

Графічна частина складається з 5 плакатів.

У економічному розділі розраховано суму витрат на розробку та виготовлення нового технічного рішення, яка складає 676021 гривень, спрогнозовано орієнтовану величину витрат по кожній з статей витрат, розраховано чистий прибуток, термін окупності витрат для виробника 1,028 роки та економічний ефект для споживача при використанні даної розробки.

Ключові слова: стартап, математична формула, експертна система, аутентифікація, зворотнє поширення помилки.

ABSTRACT

Mashnytskyi E.P. Information technology of startup management. Master's qualification thesis on specialty 122 - computer science, educational program - computer science. Vinnytsia: VNTU, 2022. 88 p.

In Ukrainian speech Bibliography: 17 titles; Fig.: 26; table 7.

The master's thesis is devoted to the development of information technology for the management of startups. The existing analogs for startup management were reviewed and analyzed, the feasibility of creating a suitable technology with additional functionality was substantiated.

The existing mathematical model was modified and an algorithm was developed for the startup management system, an expert system was selected and integrated into the system. The programming language and development environment of this system are substantiated. Created a web application in C# and JavaScript to manage a startup.

The graphic part consists of 5 posters.

In the economic section, the amount of costs for the development and production of a new technical solution is calculated, which is 676,021 hryvnias, the estimated amount of costs for each of the cost items is predicted, the net profit is calculated, the payback period for the manufacturer is 1,028 years and the economic effect for the consumer when using this development.

Key words: startup, mathematical formula, expert system, authentication, backpropagation of error.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Ефективне управляти стартапом надзвичайно актуальна тема сьогодення. Сьогодні у світі налічуються мільйони стартапів, але лише одиниці стають по-справжньому успішними. Більшість стартапів жахливо керуються і звичайно в середовищі де завжди щось ламається, де у вас ніколи не вистачає ресурсів і де менеджери не мають часу для швидкого вдосконалення – система управління стартапом є надзвичайно необхідною. Часто стартапом керують люди, які чудові інженери, але не володіють менеджерськими якостями, що накладає відбитки на управлінні стартапом. Адже від організації роботи залежить те чи буде успішним даний стартап та чи залучить інвестиції.

Налагодити роботу у стартапові доволі складне завдання, потрібно контролювати чимало процесів роботи і швидко адаптуватись до зміни середовища, тому важливо користуватись допоміжними сервісами.

Управління стартапом охоплює програми для планування завдань, спільної роботи, спілкування, швидкого керування, набором працівників. Використовуючи автоматизоване управління процесами та керуванням, команди можуть покращити прибуток та зменшити витрати, тому система для управління стартапом актуальна тема.

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

Мета та завдання досліджень. Мета магістерської кваліфакаційної роботи є підвищення ефективності управління стартапом, за допомогою розробки інформаційної технології надання рекомендацій.

Для досягнення мети слід розв’язати такі **задачі**:

– здійснити аналітичний огляд предметної області та існуючих програмних рішень для управління стартапом;

– обґрунтувати вибір доцільних технологій для реалізації додатку управління стартапом;

– розробити загальну структуру та структуру бази знань для інформаційної технології управління стартапом та заповнити її;

– розробити математичну модель інформаційної технології управління стартапом;

– розробити алгоритм роботи веб-додатку управління стартапом та виконати відповідну програмну реалізацію;

– провести тестування розробленого програмного засобу та проаналізувати отримані результати.

Об’єкт дослідження є процес управління проектами, новоствореними, невеликими техно-компаніями.

Предметом дослідження є інформаційна технологія для управління стартапом.

Методи дослідження. У роботі використані такі методи наукових досліджень: методи оброблення цифрової інформації, методи математичної статистики для обрахунків результатів отриманих за допомогою програмного засобу.

Наукова новизна одержаних результатів полягає в наступному: розширено функціональні можливості інформаційної технології для автоматизованого управління стартапом, яка навідміну від існуючих містить удосконалену математичну модель рекомендаційної системи, що дозволяє мінімізувати трудовитрати менеджменту. Вперше запропонована реалізація вищенаведеного функціоналу на базі новітньої технології .NET Core, що дозволяє гнучко розширяти та впроваджувати технологію.

Практичне значення одержаних результатів:

1. Розроблено архітектуру технології управління стартапом та інтегровано базу знань у технологію.
2. Розроблено алгоритм роботи серверної частини та клієнтської частини.
3. Здійснено програмну реалізацію технології управління стартапом.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується строгістю постановки задач, коректним застосуванням математичних методів під час доведення наукових положень, строгим виведенням аналітичних співвідношень, порівнянням результатів з відомими, та збіжністю результатів математичного моделювання з результатами, що отримані під час впровадження розроблених програмних засобів.

Особистий внесок здобувача. Усі результати, наведені у магістерській кваліфікаційній роботі, отримані самостійно.

Апробація результатів роботи. Результати досліджень було апробовано на IV міжнародній науково-практичній конференція “SCIENTIFIC PROGRESS: INNOVATIONS, ACHIEVEMENTS AND PROSPECTS”.

Публікації. За основними результатами досліджень опубліковано одну публікацію [1] та отримано свідоцтво про реєстрацію авторського права на твір.

1 АНАЛІЗ ПРОБЛЕМ СТВОРЕННЯ ВЕБ-ДОДАТКІВ ДЛЯ АВТОМАТИЗОВАНОГО МЕНЕДЖМЕНТУ СТАРТАПІВ

1.1 Аналіз предметної області менеджменту стартапів

Термін «стартап» визначає новостворену компанію на перших етапах свого існування. Стартапи пов`язані з високим ризиком, оскільки невдача можлива, але вони також можуть бути унікальними з акцентом на інноваційні технології або засновані на новій, унікальній ідеї, яку інші не помітили. Доволі часто стартапи зосереджуються на одному продукті або послузі, ці компанії, як правило, не мають бізнес-моделі і що більш важливо, не мають достатньо капіталу для інтегрування наступного етапу бізнесу. Більшість із них фінансується їхніми засновниками. У сьогоденні багато стартапів засновують інженери, які мають прикладні інженерські навички, але не мають менеджерської практики управління новоствореним проектом. Також варто зазначити, що стартапи можуть використовувати початковий капітал для інвестування в дослідження ринку, що допомагає визначити попит на продукт або послугу, тоді як комплексний бізнес-план визначає місію компанії, бачення та цілі, а також стратегії управління та маркетингу [1].

Головною характеристикою стартапа є безупинна недостача ресурсів та безупинна боротьба у конкурентному середовищі. Також варто зазначити , що більшість стартапів запроваджують нові технології на відміну від традиційного бізнесу, у якому компанії доволі повільно запроваджують нові технології.

Стартапи часто збирають кошти, звертаючись до родини та друзів або за допомогою венчурних капіталістів. Це група професійних інвесторів, які спеціалізуються на фінансуванні стартапів. Краудфандинг став життєздатним способом для багатьох людей отримати доступ до грошей, необхідних для

просування в бізнес-процесі. Підприємець створює краудфандингову сторінку в Інтернеті, що дозволяє людям, які вірять у компанію, жертвувати гроші. Стартапи можуть використовувати кредит для початку своєї діяльності. Ідеальна кредитна історія може дозволити стартапу використовувати кредитну лінію як фінансування. Цей варіант несе в собі найбільший ризик, особливо якщо запуск невдалий. Інші компанії обирають позики для малого бізнесу, щоб сприяти зростанню, саме тому зазвичай стартапи мають постійну недостачу фінансів.

Класично науковці, досліджуючи стартапи, описують розвиток стартапа у 5 скорочених стадія :

- Seed – створення ідеї стартапу;
- Startup – старт проекту;
- Growth – етап зросту стартапа;
- Expansion – розширення стартапу;
- Exit вихідний етап;

Інколи науковці вводять і розширену систему класифікацій розвитку стіртапів [2]:

- Створення ідеї, визначення цільової аудиторії, пошук конкурентів:
- Розробка бізнес плану - pre-seed;
- Залучення фінансів
- Створення прототипу;
- Залучення інвестицій на повноцінний додаток
- Альфа версія продукту або додатку (alpha);
- Бета-версія проекту або продукту для тестування (private beta);
- Готова бета-версія проекту. Доопрацювання ще продовжується (public beta).

Управління стартапом у конкурентному ринку доволі не просте, за короткий проміжок часу, змінюється багато факторів, які впливають на розвиток. Саме тому у наш час ми можемо знайти чимало різних продуктових

додатків, однак одиниці з них лише стають успішними, одним з вирішальних факторів успіху є управління. Так як стартап новостворений проект, який лише пробує залучити інвестиції, додаток для управління має бути максимально дешевий та доступний. На рисунку 1.1 наведено види користувачів системи управління стартапом системи.

Рисунок 1.1 –види користувачів системи управління стартапом

Управління компанією, якою б доброю вона не була, є величезним викликом. Однак керувати стартапом може бути навіть важче, тому що вам доведеться подбати про всі потреби вашої компанії, працюючи з обмеженим бюджетом. Управління стартапом означає робити більше з меншими витратами – і це також означає навчитися не робити все самостійно. Щоб побудувати та розвивати свій стартап, потрібні певні інвестиції в інструменти, які допоможуть покращити кожну робочу частину компанії, автоматизуючи та оптимізуючи все, від підтримки клієнтів до розробки програмного забезпечення.

Управління стартапом можна полегшити, якщо використовувати автоматизовані рішення для управління командою. Менеджеру стартапа потрібно контролювати всі аспекти проекту: ресурси, які є в проекту, результати праці команд, вивчати зміни у ринку, проводити зустрічі з

інвесторами та зацікавленими сторонами. За для успішної роботи керівнику потрібно помічники та також масивний стаж праці. Інструмент управління проектом допоможе керівнику організувати, спланувати, відстежувати та аналізувати дані, процеси, команду та інші аспекти на різних етапах проекту. Крім того, управління проектами добре підходить для перевірки ідеї. Система стає у нагоді для дотримання термінів, сприяння співпраці та навіть скорочення витрат. Комунікація та організація - це два поняття, які відіграють активну роль у будь-якій фірмі. Тож важливо стежити за двома або тримати їх у полі зору.

Система для моніторингу дозволяє менеджеру збирати інформацію про ринок та прогнозувати подальші кроки. Також управлінець може позбутися не запланованих витрат, адже має повний контроль та моніторинг. Вдале управління проектами може допомогти команді бути мотивованою, залишатися на шляху та не відволікатися. Це особливо важливо для стартапів з меншими командами, де кожен член команди має більше обов'язків і жонглює кількома проектами.

Більшості стартапів потрібна кристально чітка дорожня карта того, куди вони рухаються, команди повинні запитувати керівників, якщо завдання чи проект потребують подальших дій або корекції курсу. У стартапів також є проекти, де пов'язані завдання. Ці завдання не можна виконувати окремо. Тому для керівників проектів дуже важливо мати інструмент для визначення всіх етапів розробки проекту разом із залежностями завдань.

Основні фактори управління можна згрупувати:

- Неправильний поділ часу на завдання та не коректний підпис;
- Не правильний розподіл ресурсами;
- Невиконання поставлених завдань у робітників;
- Тривалі простої, які спричиняють затримку розвитку стартапа.

Вилучивши не потрібні витрати проект може зменшити витрати і збільшити прибуток не змінюючи цінової політики стартапу. Після

встановлення додатку можна оцінити ефективність економічного управління та помітити ефективні зміни. Додаток управління містить модулі з рисунку 1.2.

Рисунок 1.2 – модулі системи управління стартапом

1.2 Аналіз підходів управління стартапом

Методології управління стартапом впливають на всі етапи розвитку проекту. Існує чимала кількість методологій для управління бізнесом, однак розглянем найбільш відомі такі як Agile, CPM, Waterfall [3].

Якщо ваш проект клієнтський, то Agile чудовий метод, який розбиває проект на різні контрольні точки. Підчас зустрічей розробники можуть поділитися досягненнями, а клієнти мають можливість зробити пропозицію або поділитися враженнями та вхідними даними відносно проробленої роботи або майбутніх завдань. У Agile є недоліки, цілком природньо, що вам доведеться витрачати чимало часу на спілкування зі замовниками, що призведе до затримки, однак часто відгуки від клієнтів заощаджують багато часу та клопоту, уникаючи скарг від клієнтів ще до їх появ. Головною перевагою Agile це адаптація до можливих змін. Отож, якщо ваш проект зазнає

значущих змін у майбутньому, то ця система підходить як найкраще. Critical Path Method навідміну від Agile направлений на ресурси, цей метод допомагає командам комбінувати ресурси. Методологія визначає сортування завдань у стартапі на основі часу та посилань на інші завдання. CRM визначає критичні завдання, які обов'язково потрібно як найшвидше зробити. Головною перевагою цього методі перш за все, це дозвіл керівникам доцільніше планувати завдання відштовхуючись від тривалості завдання і не від інших факторів, завдяки цьому керівник стартапу може швидше і точніше спланувати роботу. Головною перепоною цього метода полягає, що неможливо його використовувати без обізнаності у предметній області. Якщо менеджер не може визначити пріоритетність завдань, то у нього не вийде все точно спланувати, приклад роботи CRM наведено на рисунку 1.3.

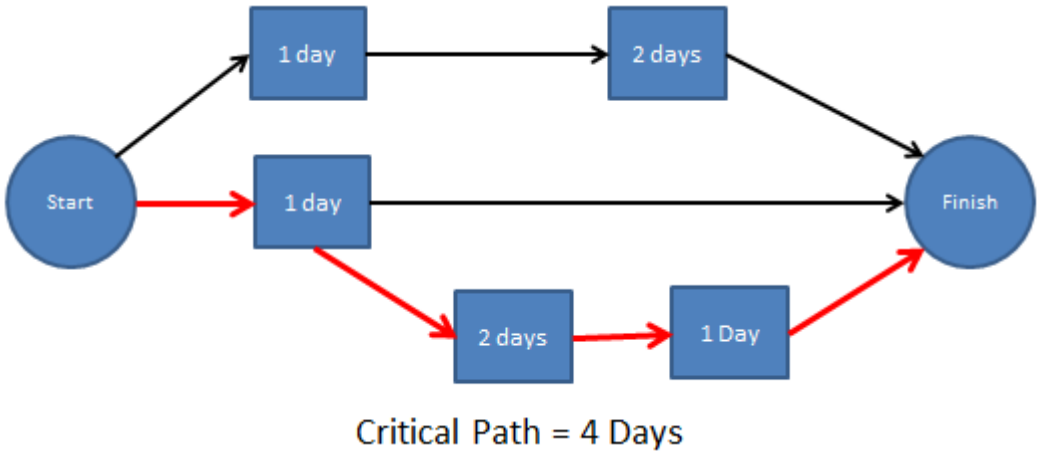


Рисунок 1.3 – Приклад роботи CRM

Waterfall подібний до Agile, предбачає розбиття проекту на періоди праці, однак якщо Agile є гнучким(не важливо у якій послідовності працювати з етапами), то цей метод передбачає послідовність виконання. Метод передбачає не рухатись вперед до того як команда не виконає поставленні завдання від попереднього етапу. Цей метод чудово підходить до маленьких проектів, де початкові цілі та завдання чудово визначенно і не будуть змінюватись.

Підсумовуючи аналіз методологій управління стартапом слід зазначити, що кожна з них має свої переваги, однак розробляючи інформаційну систему варто розширити до гнучкого застосування цих методів, адже на стартапі можуть бути різні періоди і в залежності від цього застосовувати той чи інший метод управління, отож у даній системі варто комбінувати підходи.

1.3 Аналіз відомих програмних рішень

Розглянемо найпопулярніші реалізації зі великого переліку методологій управління стартапом.

Популярним додатком для управління стартапом є ActiveCollab [4], на рисунку 1.4, наведено приклад роботи даної системи.

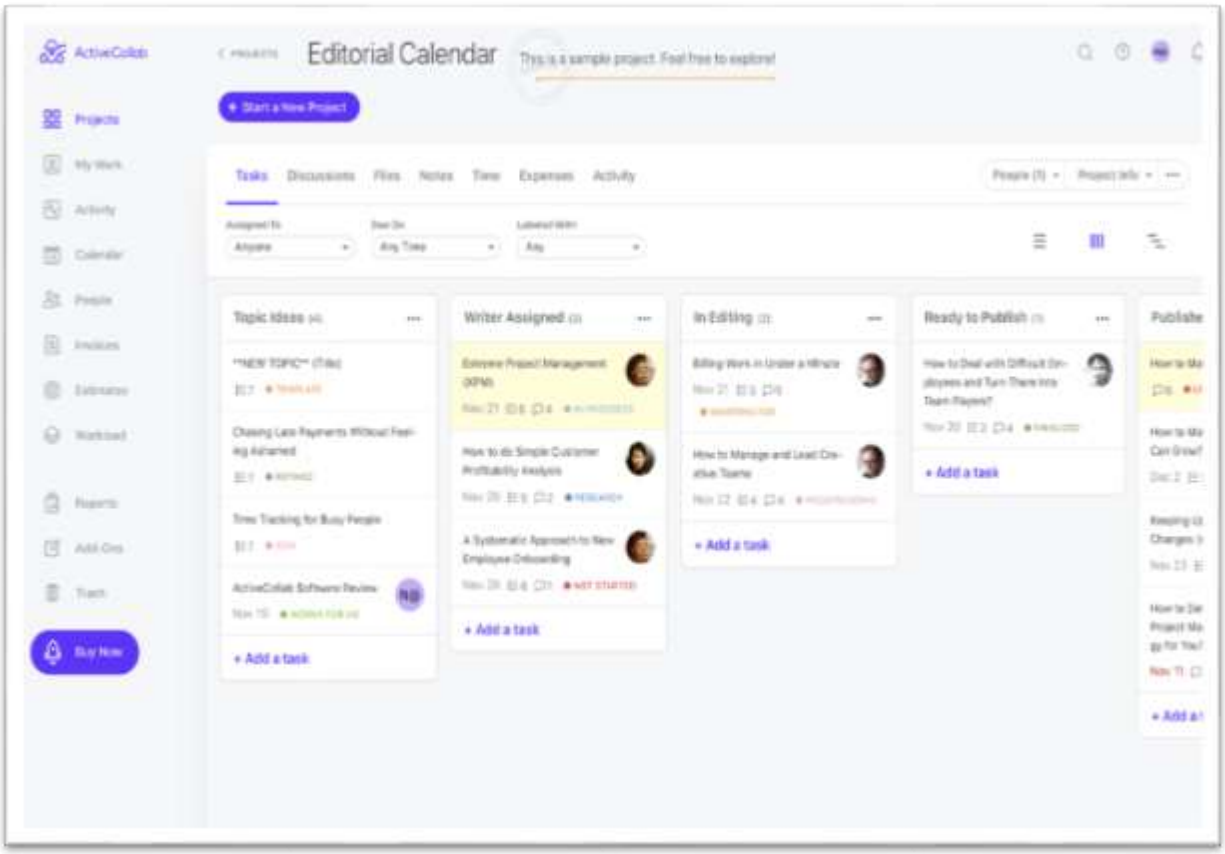


Рисунок 1.4 – Інтерфейс ActiveCollab

Застосунок чудово виконує поставленні вимоги, надіючи менеджеру можливості контролювати виконання завдань у команді. У додатку менеджер стартапу може контролювати командну роботу, добавляти чи редагувати працівників, також є можливість різнорівневої комунікації з командами або розсилання повідомлення. Важливим функцією є створення завдань, розділів а є можливість призначати певне завдання на конкретного працівника або на команду. Варто зазначити що всі зміни, які відбуваються зі завданням зберігаються в історії завдання. Додаток доволі добре взаємодіє з існуючими командними сервісами, що дозволяє легше контролювати робочий процес.

Також в ActiveCollab менеджер може створювати звіт в реальному часі з максимально актуальними оновленими даними своїм клієнтам, які можуть аналізувати за розвиток проекту не належачи до системи, інтерфейс застосунка наведено на рисунку 1.5.

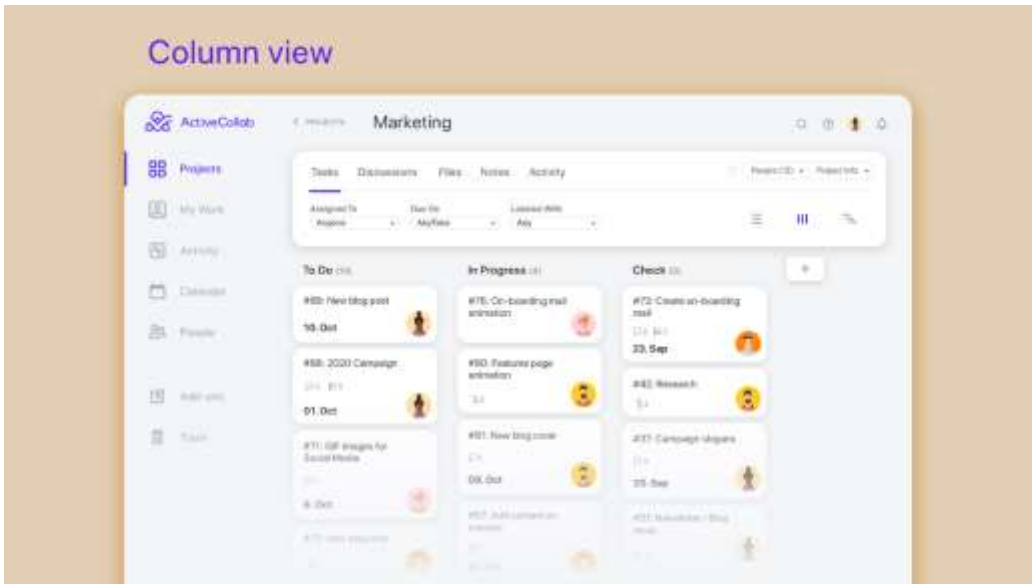


Рисунок 1.5 – Інтерфейс ActiveCollab

Інший приклад, система Monday.com [5], додаток один з найпопулярніших програмних рішень для управління проектами, яким сьогодні користуються гравці у ринковій системі управління стартапом. Monday є одним із найкорисніших інструментів, який команди

стартапів повинні розглянути про впровадження на ранній стадії, щоб згодом приєднатися до великих гравців. Додаток дозволяє вам спілкуватися різними способами, не залишаючи платформи: коментування, вкладення файлів, теги, «лайки» дописів і призначення завдань за допомогою приміток. Ви також можете створювати інформаційні панелі та дошки проектів для кожної команди, щоб кожен міг легко бачити, над чим працює.

Monday.com інтегрується з інструментами, що робить перехід команди стартапу без проблем. Інтеграція включає програми для керування проектами, як-от Slack, Google Drive, Gmail, Google Calendar, Jira, GitHub, Trello, Dropbox, Typeform та багато інших, доступних через платний план із Zapier. Додаток коштує від 6 доларів США за користувача на місяць і має безкоштовну 14-денну пробну версію.

Monday.com базується на концепції створення або оголошення на дошці. Дошка— це місце для організації ваших елементів, таких як проекти, списки справ, CRM(Customer relationship management) і розклади. Маючи декілька типів плат, материнські плати створюють прозорість у компанії, оскільки їх можуть бачити та використовувати всі користувачі облікових записів. Ви можете використовувати попередньо визначені шаблони або створювати налаштовані відповідно до ваших потреб. Після цього можна створювати групи. Дошка складається з груп. Якщо визначаєте групи, це стає кольоровим розділом на дошці, який містить усі відповідні завдання. Групою може бути, наприклад, тиждень, місяць або конкретний крок проекту. Після завершення board і груп ви оголошуєте нові елементи. Вони можуть бути окремим рядком у групі, що представляє будь-що, від завдань до проектів. Тоді настав час почати додавати стовпці. Стовпці – це спосіб опису даних, що стосуються оголошених/створених елементів. Можна призначати завдання, обговорювати їх і змінювати їхній статус відповідно до прогресу.

Створивши проект менеджер буде мати можливість добавляти працівників у проекті, а співробітники досліджувати проект на інтерактивній дошці, що наведено на рисунку 1.6.

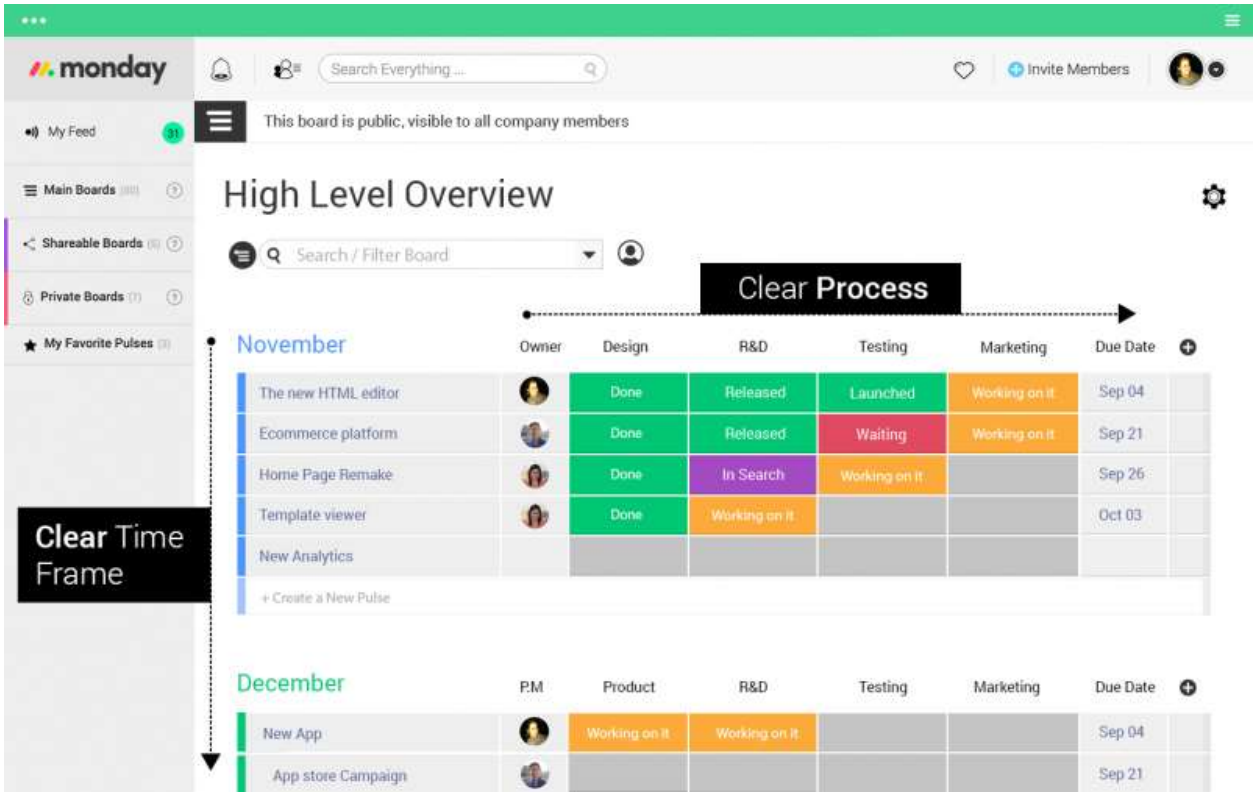


Рисунок 1.6 – Інтерфейс Monday

На початковій сторінці користувачі можуть бачити всі новини сайту, а саме: новостворені проекти, нових користувачів, що мають певні ресурси, також можна побачити як розвивається той, чи інший проект.

Також варто розглянути додаток Basecamp [6]. Basecamp — це онлайн- додаток для спільної роботи, який дозволяє людям спільно керувати своєю роботою та спілкуватися один з одним. Менеджери використовуєте його, щоб відстежувати всі завдання, терміни, файли, обговорення та оголошення, які відбуваються на роботі. Хоча дехто називає його програмою для керування проектами — і, звичайно, ви можете використовувати її для керування проектами, — ми класифікуємо її як

програму для співпраці через її більш гнучку структуру та відсутність діаграм Ганта. Це більше схоже на віртуальний центр для команд і організацій. Basecamp чудовий у тому, що він робить, і дуже доступний для великих команд. Обліковий запис Basecamp коштує 99 доларів на місяць або 999 доларів на рік для необмеженої кількості членів команди та необмеженої кількості проектів. Існує також безкоштовна версія Basecamp під назвою Basecamp Personal, яка дозволяє керувати трьома проектами до 20 осіб, але обліковий запис не включає всі функції.

Програми для керування проектами відрізняються тим, як вони дозволяють керівникам відстежувати кілька проектів одночасно з дрібними деталями, звертаючи увагу, зокрема, на кінцеві терміни, хід виконання завдань і навантаження. Будь-яка хороша програма для керування проектами дає змогу побачити, коли дедлайн загрожує зривом, перш ніж це станеться. Якщо хтось пропустив дедлайн, програма для керування проектами дозволить вам відкоригувати всі наступні дедлайни завдань, які залежать від того, яке пройшло. Наприклад, якщо B і C не можна виконати, доки не буде зроблено A, а A запізнюється на два дні, тоді кінцеві терміни для B і C, можливо, доведеться також перенести на два дні, і так далі.

Програми для керування проектами також дозволяють переглядати докладні звіти про те, скільки роботи кожна особа доручила їм, і легко перепризначати завдання, коли хтось стає перевантаженим або неочікувано недоступним. Деякі також містять інструменти для встановлення бюджету для проектів і автоматичного розрахунку вартості зусиль кожного члена команди на основі погодинної ставки, яку ви вводите для них.

Коротше кажучи, програми для керування проектами дозволяють вам детально бачити та керувати тим, як усі частини багатьох проектів поєднуються разом. Basecamp цього не робить.

Однак також варто відмітити, що до у додатку багато приділено уваги до комунікацій, та додаток дає гнучкі можливості для роботи з файлами. Для

завантаження файлу доступна можливість попереднього перегляду, на рисунку 1.7 показано сторінку управління в даній системі.

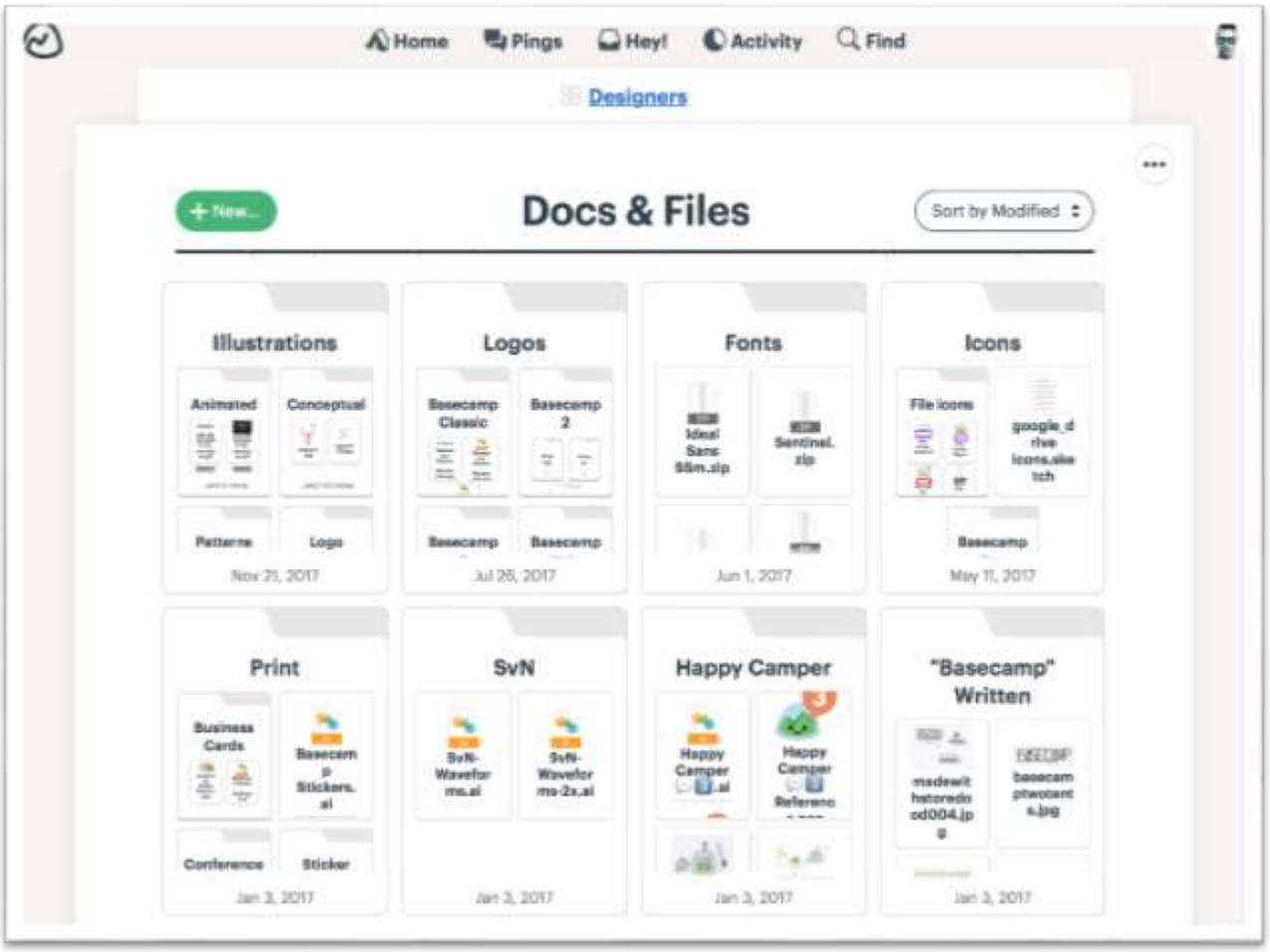


Рисунок 1.7 – Інтерфейс Basecamp

Basecamp має перевагу у роботі з файлами та трансферах файлів серед команди або між командами. Головним мінусом є його ціна та відсутність повного функціоналу для управління командою.

Проаналізувавши вищенаведені системи , можна дійти висновка, що інформаційна система повинна включати:

- «functionality for command control» – а саме функціональність для редагування команд, створення нових тощо;

- warnings - функціонал для підказки, якщо проект не виконується за чітким графіком;

- reporting – варто добавити звітність у проект, що має включати різні графіки та взаємодіяти з блоком warnings;

1.4 Постановка задач дослідження

Підставою для розробки технічного завдання є завдання з дипломного проектування.

Проаналізувавши вищенаведені приклади реалізації та шляхи їх вирішення, було вирішено розробити інформаційну система для управління стартапом, яка полегшує управління стартапом

У ході дослідження ми зосередилися на системі, яка у собі має містити управлінські можливості та пошук проекту. Особливу увагу варто приділити методологіям управління стартапом, адже в залежності від методологій управління буде вибудовуватись робота у команді, якщо не вірно налаштувати роботу у команді то стартап погрузне у не виконаних завданнях та втратить свою привабливість.

Упровадження системи буде доволі просте, через реєстрацію команди у системі, кожен стартап буде мати свої налаштування для інтерфейсу за для полегшення роботи. Також варто добавити експертні рішення у інформаційну систему за для полегшення управління системою.

Рішення має бути Multitenancy, коли кілька незалежних екземплярів однієї або кількох програм працюють у спільному середовищі. Користувачам може бути надана можливість налаштувати деякі частини програми, наприклад колір інтерфейсу користувача (UI) або бізнес-правила , але вони не можуть налаштувати код програми.

Основними вимогами інформаційної системи управління стартапом будуть:

- multitenancy – для кожного орендаря створюється окремий екзепляр реалізації;
- stability – додаток має бути стійким до вірусних додатків та збоїв;

- cross-platform – має мати можливість розгортатись на різних операційних системах;
- 5 Гб пам’яті на кожнен зареєстрований проект
- кожен стартап має мати змогу редагувати технологію під свої задачі

Розроблений додаток дозволить:

- контролювати реальну витрату ресурсів;
- контролювати час роботи ;
- запобігати махінаціям з ресурсами;
- робити оптимальний витрати часу на під проект;
- поліпшити трудову дисципліну;
- підвищити ефективність управління компанією в цілому.

1.5 Висновок до розділу 1

Здійснено аналітичний огляд найбільш поширених методологій управління проектами. Огляд показав що дані методології мають ряд недоліків, зокрема scrum не доцільно використовувати на проекті з не чіткими вимогами, отож запропоновано поєднувати методології у даній системі. Обгрунтовано доцільність розробки інформаційної технології для менеджменту стартапів, що передбачає додаткові функціональні можливості.

Здійснено огляд та аналіз існуючих інформаційних систем, що дозволяють управляти стартапом. Преважна більшість систем не можуть забезпечити достатню високу гнучкість під час управління та не задовільнять вимогам сьогодення, жодна система не аналізує дій менеджера та не надає відповідних підказок.

Поставлено задачі до магістерської кваліфікаційної роботи та розроблено вимоги до технології управління стартапом.

2 РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМУ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ УПРАВЛІННЯ СТАРТАПОМ

2.1 Розробка математичної моделі для інформаційної технології управління стартапом.

Первиною метою інформаційної технології менеджменту стартапів є покращення управління, тобто завдання зводиться до дослідження причин збитків та активна діяльність щодо покращення управління. Запорукою успішного бізнесу є підготовка. Перш ніж стартап буде успішним, йому доведеться заплатити за рахунками. Розуміння витрат допоможе вам успішно розпочати роботу.

Модель витрати стартапа можна подати у наступній формулі [7]:

$$Q(X) = \sum_{i=1}^{n}(Q_i d_i + w_i)x_i \rightarrow \min. \quad (2.1)$$

Де Q(X) – цільова функція, що описує сумарні витрати стартапу на реалізацію продукції за оденицю часу i=1,…, n; Формула аналізу використовується для визначення одиниць, необхідних для отримання прибутку. Вартість відноситься до постійних і змінних витрат, понесених компанією. Обсяг означає кількість проданих продуктів. Прибуток означає, скільки грошей компанія заробляє, враховуючи ціну проданого продукту, обсяг проданої продукції та постійні та змінні витрати компанії.

$$NPV = \sum_{t=1}^{n}(\frac{B_t - C_t}{(1+i)})/ \quad (2.2)$$

NPV – дисконтова вартість стартапу, що визначає вартість прибутків відносно вкладених інвестицій.

Якщо аналізувати довгостроковий проект із кількома грошовими потоками, то формула для NPV проекту виглядає наступним чином [8]:

$$\text{TO} = (t_0 - 1) + \frac{\sum_{t=1}^{n}(\frac{P_t}{(1+i)})}{(1+i)^t}, \tag{2.3}$$

$$\text{NPV} = \sum_{t=1}^{n}(\frac{P_t}{(1+i)}), \tag{2.4}$$

$$\text{NPV} = \sum_{t=1}^{n}\left(\frac{P_t}{(1+i)}\right) - (t_0 - 1) + \frac{\sum_{t=1}^{n}(\frac{P_t}{(1+i)})}{(1+i)^t}. \tag{2.5}$$

NPV враховує часову вартість і може використовуватися для порівняння прибутку різних проектів або для порівняння прогнозованої норми прибутку з перешкодою, необхідною для схвалення інвестицій. Вартість грошей у часі представлена у формулі NPV ставкою дисконту, яка може бути перешкодою для проекту на основі вартості капіталу компанії. Незалежно від того, як визначається ставка дисконтування, від'ємний NPV показує, що очікувана норма прибутку буде нижчою від неї, тобто проект не створить цінності. Ставка дисконту є центральною у формулі, ставка дисконту – це просто базова норма прибутку, яку проект має перевищити, щоб він був корисним [9-12].

Аналізуючи ці формули, реалізація даної математичної моделі дозволить позбутись витрат та покращити NPV. Вищенаведені фактори покривають поставленні задачі.

2.2 Розробка структури роботи інформаційної системи управління стартапом

Аналізуючи вхідні дані потрібного функціоналу інформаційної системи для менеджменту стартапу, та досліджуючи дані, які потрібно зберігати та досліджуючи предметну область, розроблено структуру інформаційної системи для управління стартапом (рис. 2.1).

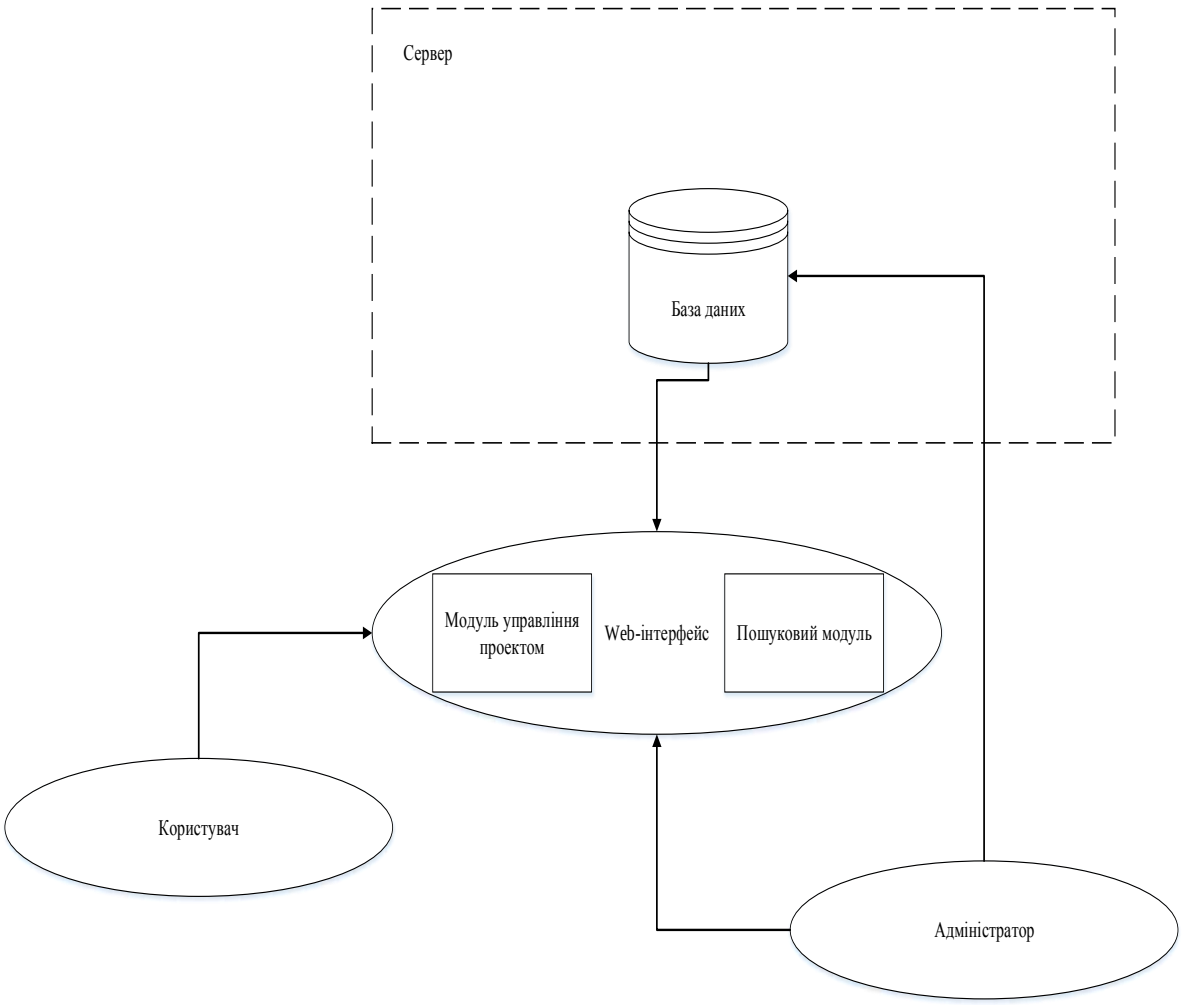


Рисунок 2.1− Структурна схема веб-системи управління стартапом

Структурна схема для інформаційної системи буде включає такі елементи:

− База даних, для зберігання усієї інформації стосовно стартапу;

- Серверна частина, яка відповідає за працю з базою даних;
- Клієнтську частину, яка відповідає за відображення;

У системі будуть різні типи користувачів з різними ролями. По замовчуванню у додатку буде адміністратор, який буде мати всі доступи та зможе контролювати всю систему.

Визначимо кроки взаємодії користувачі з системою [13-15]:

1. При першому потраплянні користувача у системо він потрапляє на сторінку авторизації, де користувач має можливість увійти у систему або зареєструватись, якщо профіля не існує. Відправивши форму клієнтська частина отримує токен доступа, який важливо використовувати при послідуючих запитах або транзакціях.

2. Ввівши пароль або увійшовши через інший сайт, на сервері створюється унікальний зашифрований токен доступу.

3. На серверній частині відбувається валідація та якщо користувач існує, генерується токен доступу інакше повертається помилка.

4. Отримавши токен, користувач має можливість взаємодіяти з системою відправляючи токен на сервер.

5. Сервер перевіряє токен доступу та опрацьовує потрібний запит, вертаючи відповідь на запит або помилку.

6. На сторонні клієнта важливо опрацьовувати помилку або відображати потрібні дані.

На рисунку 2.2 наведено діаграму діяльності роботи системи веб-служб, враховуючи взаємодію описаних інстанцій.

Рисунок 2.2 – Діаграма діяльності алгоритму роботи модуля

Якщо запит не пройшов верифікацію креденшелів, користувача буде направлено на сторінку логіну де потрібно буде увести пароль.

Для взаємодії між клієнтською і серверною частиною використовується HTTPs. HTTPS використовує протокол шифрування для шифрування зв’язку. Протокол називається Transport Layer Security (TLS), хоча раніше він був відомий як Secure Sockets Layer (SSL). Цей протокол забезпечує захист зв’язку за допомогою так званої інфраструктури асиметричного відкритого ключа . Цей тип системи безпеки використовує два різні ключі для шифрування зв’язку між двома сторонами

HTTPS запобігає трансляції інформації веб-сайтів у спосіб, який легко переглядати будь-хто, хто стежить за мережею. Коли інформація надсилається через звичайний HTTP, інформація розбивається на пакети даних, які можна легко отримати за допомогою безкоштовного програмного забезпечення. Це робить зв’язок через незахищене середовище, наприклад загальнодоступну Wi-Fi, дуже вразливим до перехоплення. На рисунку 2.3 наведено UML-діаграму класів, на ній зображено зв’язки між класами та структурами поекту.

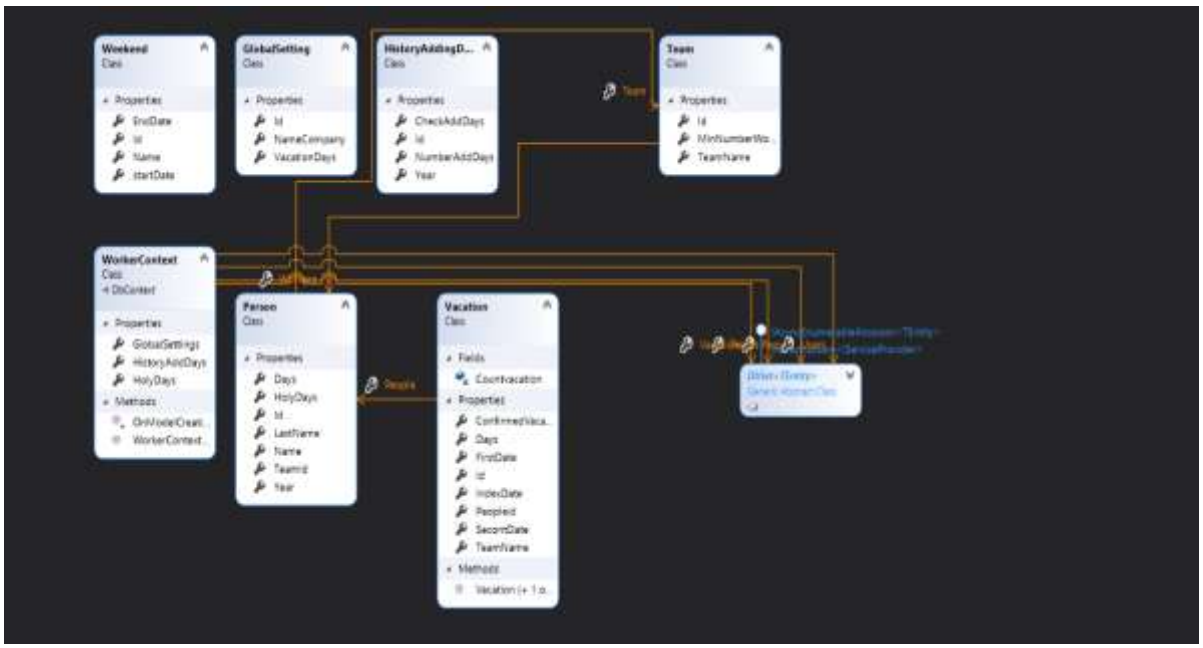


Рисунок 2.3 – UML-діаграма класів

UML-діаграма зображає структуру проекту. Аналізуючи діаграму ми можемо побачити різні зв’язки у даній інформаційній системі, а саме клас Vacation має зв’язок з Person багато до одного, аналогічна ситуація з Person і Team має також зв’язок один до багатьох.

2.3 Проектування структури бази даних для інформаційної технології управління стартапів

Для продуктивної роботи інформаційної системи, варто забезпечити швидку працю з великими об’ємами даних, їх варто зберігати на різних серверах та потрібно забезпечити безпеку даних, а саме доступ до серверів має бути лише у обмеженого кола. Вирішити поставлену задачу можна за допомогою (СУБД) MsSQL. Microsoft SQL Server є однією з основних систем керування реляційними базами даних на ринку, яка обслуговує широкий спектр програмних додатків для бізнес-аналітики та аналізу в корпоративних середовищах. Microsoft SQL Server ідеально підходить для зберігання всієї потрібної інформації в реляційних базах даних, а також для безпроблемного керування такими даними завдяки своєму візуальному інтерфейсу та наявним у нього параметрам і інструментам. Це життєво важливо, особливо для веб-сайтів, які мають можливість реєструвати користувачів для входу. Заснований на мові Transact-SQL, він містить набір стандартних мовних розширень програмування, а його додаток доступний для використання як локально, так і в хмарі.

Основні переваги даної СУБД є [16]:

1. Підвищує безпеку даних. Головною метою Microsoft SQL Server є підтримка безпеки бази даних. Дане програмне середовище забезпечує працю з таблицями обмежуючи можливість редагування.

2. Простота налаштування. Дана СУБД доволі популярна і доволі проста у налаштуванні.

3. Швидкість обробки даних. Microsoft SQL Server дозволяє індексувати дані. Індекс — це структура на диску, пов’язана з таблицею або поданням, яка прискорює пошук рядків із таблиці чи подання. Індекс містить ключі, створені з одного або кількох стовпців у таблиці чи поданні. Ці ключі зберігаються в структурі (B-дереві), яка дозволяє SQL Server швидко й ефективно знаходити рядок або рядки, пов’язані зі значеннями ключів. На рисунку 2.4 наведено інтерфейс профайлера бази даних.

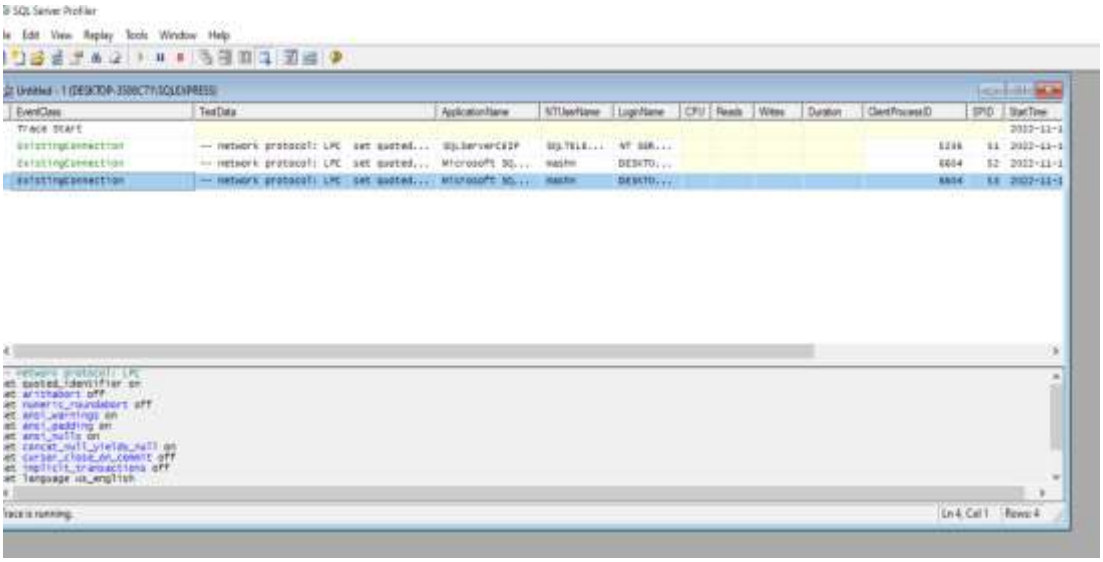


Рисунок 2.4 – Інтерфейс профайлера бази даних

Перевагою даної СУБД є студія для управління, що дозволяють інтуїтивно працювати з базою даних, також підтримується звітність роботи бази даних. Розроблено базу даних на рисунку 2.5 наведено зв’язки між сутностями, базу даних розділено на базу даних з проектом управління проектом, командою. Також створено ще одну базу даних з користувачами.

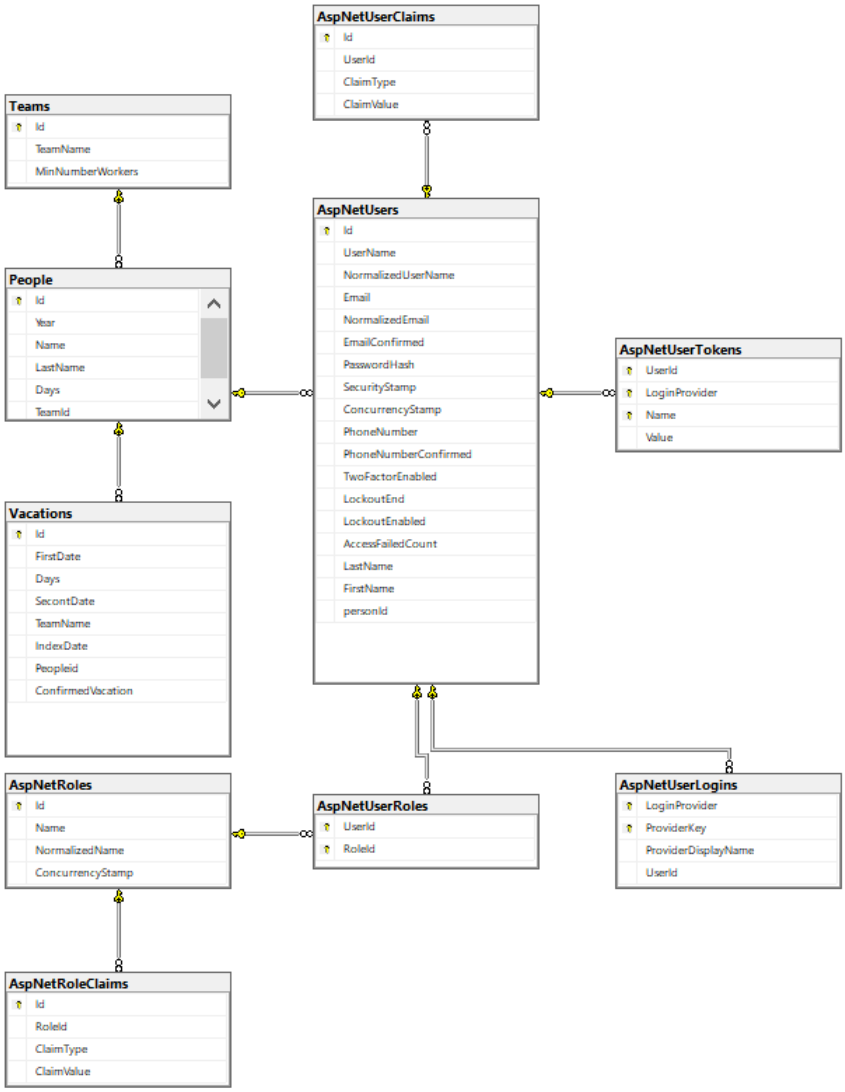


Рисунок 2.5 – Загальна схема бази даних роботи модуля

Усі зв’язки індексуються автоматично при створенні таблиці по первинному ключі, що значно пришвидшує пошук у таблицях.

Також було застосовано нормалізацію бази даних до третьої форми. Перша нормальна форма була досягнута розділенням таблиць на скалярні дані, що дозволило не повторяти кортежі. Друга нормальна форма була досягнута адже кожний не ключовий атрибут залежить від первинного ключа. Третя нормальна форма дозволила уникнути транзитивної залежності атрибутів у таблиці.

2.4 Розробка експертної системи для управління стартапом

Експертна система — це класичний символічний підхід до штучного інтелекту, який є одним із найкращих прикладів представлення штучних когнітивних систем (ACS) [17]. Оболонка експертної системи — це, по суті, інструмент спеціального призначення, створений відповідно до вимог і стандартів конкретної області або додатків експертної області знань. Його можна визначити як пакет програмного забезпечення експертних систем шляхом надання схеми представлення знань і механізму логічного висновку.

Shell відноситься до програмного модуля, що містить інтерфейс, механізм логічного висновку та структурований вибір бази знань із відповідними засобами представлення знань. З точки зору неспеціаліста, це можна розглядати як порожню чашу, яку ще потрібно наповнити елементами експертних знань, які разом із механізмом інтерфейсу можуть використовуватися для обробки запитів користувачів для створення рішень для проблем користувачів. По суті, будь-яка комп'ютерна програма, яка, будучи забезпечена певною базою знань, створює експертну систему, називається оболонкою експертної системи. Сховище фактичних і евристичних знань. Інструмент експертної системи надає одну або кілька схем представлення знань для вираження знань про область застосування. Деякі інструменти використовують як фрейми (об'єкти), так і правила IF-THEN. У PROLOG знання представлені у вигляді логічних тверджень.

Механізми висновку для маніпулювання символічною інформацією та знаннями в базі знань формують лінію міркувань у вирішенні проблеми. Механізм висновку може варіюватися від простого modus ponens зворотного ланцюжка правил ЯКЩО-ТОДІ до аргументації на основі випадків (рис. 2.6).

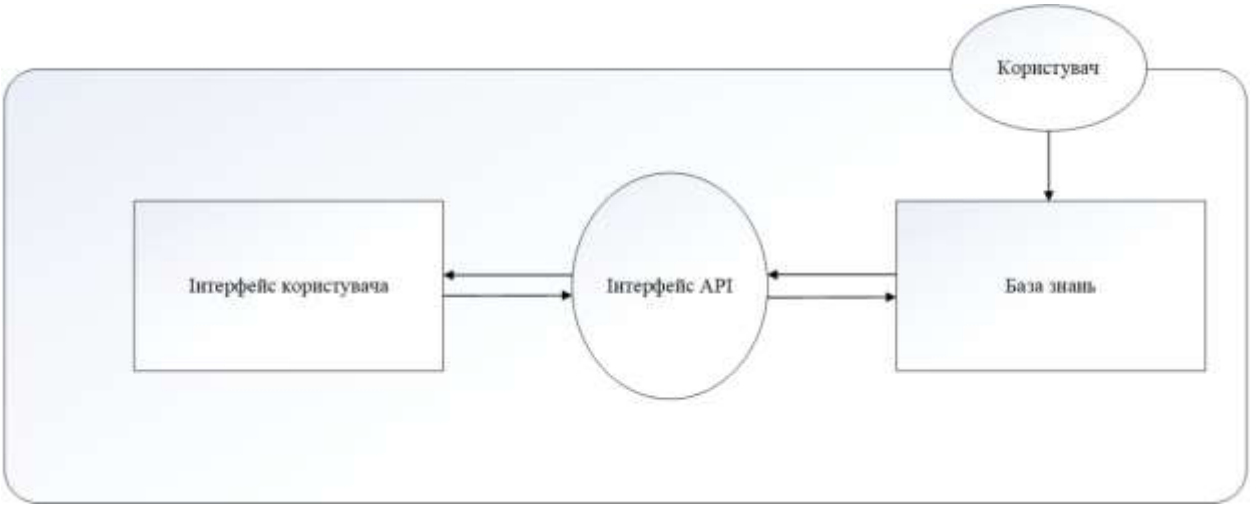


Рисунок 2.6 – Загальна схема роботи експертної системи

2.4 Висновок до розділу 2

Запропоновано математичну модель прогнозування чистої поточної вартості інформаційної технології для менеджменту стартапами.

Розроблено загальну структурну схему інформаційної технології для управління стартапом. Система складається з модулів: клієнтська частина (відповідає за відображення даних та взаємодію з користувачем), експертна система (аналізує проект) а також серверна частина (взаємодіє з базою даних та експертною системою).

Наведено діаграму діяльності інформаційної технології для управління стартапом, визначено структуру системи та наведено роль кожного рівня системи. Створено алгоритм роботи системи, який задовільняє поставленим вимогам, зокрема користувач, який не авторизований не має доступу до управління стартапом.

Обґрунтовано доцільність вибору системи управління для бази даних, а саме MsSql, яка дозволяє швидко взаємодіяти з великими об’ємами даних та містить переваги реляційної СУБД. Здійснено нормалізацію бази даних до третьої нормальної форми.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ДЛЯ МЕНЕДЖМЕНТУ СТАРТАПІВ

3.1 Обґрунтування вибору мови та середовища програмування

Інформаційна система менедменту стартапами вимагає високої швидкості опрацювання запитів та потребує об’єктно-орієнтовану парадигму за для швидкого масштабування.

Головними серверними мовами , які задовільнять ці вимоги є:

– Java;
– C#;
– Node.js.

З переліку було обрано C# [18], дана мова задовольняє критеріям, вирішено використовувати кросплатформений фреймворк Asp .net Core.

Мова C# має такі основні критерії:

– В основі об’єктно-орієнтована парадигма;
– Компонентно-орієнтована мова;
– Строго типізована мова;
– Існує автоматичний контроль використання пам’яті;
– Мова є компільованою;
– Існує велика кількість бібліотек , що дозволяють опрацьовувати веб-запити або міститься різні алгоритми шифрування, авторизації та автентифікаці, тощо.

Варто також відмітити велику кількість інструментів для опрацювання найрізноманітнітніших типів даних, з базовими такими як числа, символи, стрічки, графічні елементи, вектори, математичні величини також є можливість будувати власні типи, дана мова програмування підтримує силочні типи та значимі:

– Array;

– List;
– LinkedList;
– IEnumerable<T> структури, що мають визначений базовий тип;
– Кортежі – Tuple;
– Структури даних, що спеціалізуються на безпечній багатопоточній обробці – Thread-Safe Collections;

Якщо розглядати продуктивність, то мова являється однією з найкращих, адже C# компілює коди в кросплатформені двійкові файли.

Також у мові добре реалізовано асинхронне програмування. За допомогою асинхронного програмування ви можете розділити свою логіку на очікувані завдання, де ви можете виконувати деякі тривалі операції без необхідності блокувати виконання вашої програми. «.Net Framework» надає прості та легкі у використанні ключові слова, які є модифікаторами Async і await для перетворення коду з синхронного на асинхронний, на рисунку 3.1 наведено приклад асинхроного підходу.

Таким чином, можна викликати асинхронний метод окремо, отримавши для нього об’єкт завдання, потім виконати деяку непов’язану роботу, а після цього очікувати це завдання, або воно вже завершилося, що призведе до повернення значення із завдання, або якщо воно не завершилося завершено, але програма виконуватиметься нормально без блокування.

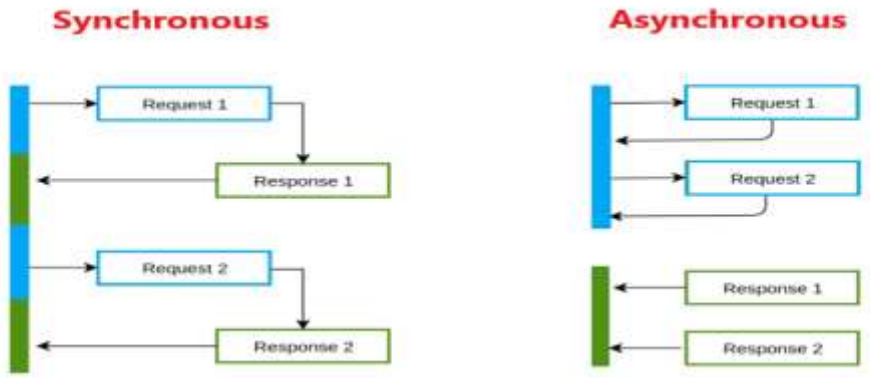


Рисунок 3.1 – Основних підходів асинхронного програмування у системах Async і Await

Коли ми працюємо з C# .Net framework, варто використовувати Garbage Collector, щоб переконатися, що ми не використовуємо непотрібні об’єкти коду в стеку пам’яті, а також ми можемо звільнити небажаний простір для процесу. Збирання сміття є критично важливим аспектом, якому приділяє особливу увагу будь-яка компанія-розробник .NET під час розробки проекту. Він вирішує проблеми з пам’яттю та легко керує різними об’єктами в проекті. Коли ми працюємо з класом, який буде представляти об’єкт у середовищі виконання, який виділить простір пам’яті. Коли дії, пов’язані з об’єктом, будуть завершені, вони будуть використані, оскільки в пам’яті залишиться невикористаний простір. Це подолає існуюче явне звільнення пам’яті.

Дана мова стрічко розвивається та постійно оновлюється, саме тому серед рейтингу мов, вона посідає провідні місця. Також варто відмітити чітку документацію, яка постійно оновлюється разом з виходом нової версії мови.

Створено чимало патернів та підходів реалізацій об’єктно орієнтованої взаємодії компонентів у проекту, також варто зазначити, що у мові чимало інтегровано патернів та їх реалізацій, що полегшує роботу інженерів з цією мовою.

Активного розвитку набув фреймворк Asp.NET Core, він дозволяє створювати крос-платформні рішення і налаштовувати додатки не лише на windows а й на linux, було полегшено працю з cloud-технології також варто зазначити що було розділено CLR від FX.

Платформа Asp.NetCore надає велику кількість реалізованих модулів для створення веб-застосунків, наприклад Microsoft.AspNetCore.Http, IHttpClientFactory, що дозволяє менше часу витрачати на розробку.

Також дана платформа містить можливість використовувати identityserver, що автоматично генерує авторизації та автентифікації на серверах. Дана бібліотека містить чимало корисних методів, які полегшують розробку. Для захисту було створено свою реалізацію

AspNetCore.DataProtection.Abstractions та AspNetCore.Cryptography, що дозволить у відповідь повертати зашифровані дані.

AspNetCore дозволяє надавши контролеру атрибут, змінювати властивістьдля роботи з класом або моделлю, що використовуються для роботи з Api.

Основними атрибутами є [19-22]:

ApiController – зазначає, що клас є контролером;

HttpPost – визначає що запит потрібно виконувати як post

HttpGet – визначає що запит потрібно виконувати як get;

Authorize – визначає хто має доступ до контролера;

Route – даний атрибут визначає налаштування маршутизації у проекті, за допомогою нього знаходиться потрібний метод.

Головним середовищем розробки для вибраної серверної Rider та WebStorm.

Rider це середовище розробки, яке дозволяє розробнику швидко імплементувати код, дозволяє працювати з усіма можливими реалізаціями Asp.Net.Core. Дана ID має можливість генерувати застостосунки з базовими налаштуваннями, що пришвидшує роботу. На рисунку 3.1 зображено роботу rider.

Дане середовище включає текстовий редактор, що дозволяє швидко опрацьовувати великі об’єми коду та має зручний інтерфейс. Intellisence надає можливість швидко виявляти помилки, маркуючи їх червоного кольору, також надає варіанти виправлення їх. Дана система містить можливість перегляду коду на застарілість і радить замітити на нові імплементації.

Даний редактор містить вбудовану можливість підключати базу даних, що надає можливість змінювати базу даних створивши міграцію. Rider містить можливість встановлення розширень, що дозволяє добавляти необхідне розширення у проект.

Система чудово взаємодіє з git, надаючи можливість створювати коміти або гілки не виходячи з додатку. Існує консоль, тому можна працювати з неї.

Rider від JetBrains має лише платну версію, а не безкоштовну. Rider походить від інших JetBrains, таких як ReSharper і WebStorm, але тепер перетворився на IDE. Він є кросплатформним, тобто може працювати як на Windows, Mac, так і на кількох версіях Linux, пропонуючи однаковий набір функціональних можливостей і однакову поведінку на всіх них. Це велика перевага для Rider: він просто виглядає та поводиться однаково всюди.

Rider є адаптивним і настроюваним, ви можете вибрати свою колірну схему, прив'язку клавіатури тощо. Він також швидкий і чуйний, ви можете мати кілька вікон, які відображаються так, як ви хочете, навіть згорнуті, а потім зберегти налаштування.

Можна створювати проекти за допомогою мов C#, F# або VB, але не всі ці мови доступні для всіх типів проектів. Ви можете націлити будь-яку з встановлених версій .NET Framework, але лише останні .NET Core або .NET Standards. Для проектів ASP.NET Core ви можете вибрати шаблон .NET, який використовує Angular, React або React і Redux.

Rider — це ReSharper, тому ви можете очікувати, що все, що було доступно в ReSharper, також буде тут. Практично будь-який рядок коду можна відредагувати, навіть якщо це просто для порізання довгих рядків або введення змінних, параметрів або полів замість жорстко закодованих («магічних») значень. Rider може інвертувати логіку умовного блоку, витягувати код до нового методу, створювати похідний тип, переміщувати методи до іншого файлу (часткові класи), перетворювати властивість у метод, перетворювати член екземпляра на статичний, видаляти Оголошення «this» тощо. Перейменування простору імен або типу подбає про всі його посилання (за допомогою операторів), як і слід було очікувати. Корисний рефакторинг використовує базовий тип замість похідних типів, коли це можливо (відповідно до принципу заміни Ліскова), інший генерує базовий тип для

існуючого, необов’язково переміщуючи до нього деякі члени. Інший перевіряє учасників на їх видимість і пропонує обмежити її, якщо це можна зробити, нічого не порушуючи.

Зручний інтерфейс дозволяє швидко інженеру виконувати потрібні маніпуляції, затративши мінімальний час на написання команд або пошук кнопок на рисунку 3.2 зображено дане середовище. Під час виконання додатку виводиться інформація про використану пам’ять та напруженість системи.

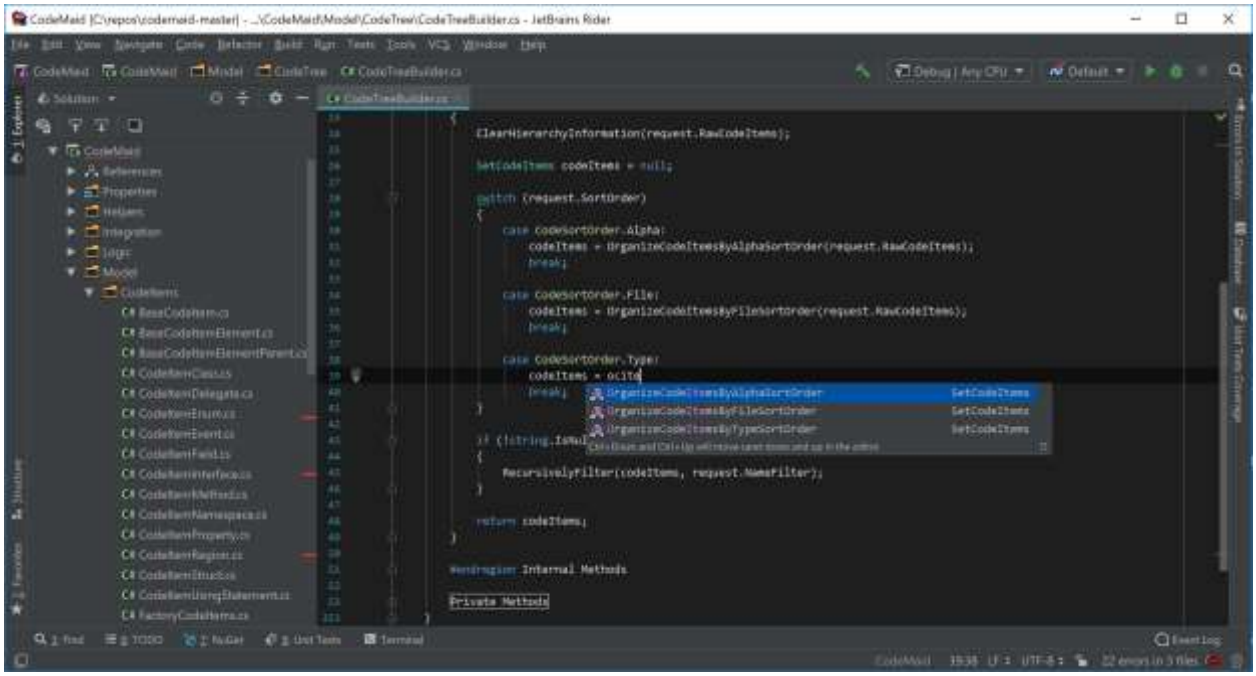


Рисунок 3.2 – Середовище розробки Rider

WebStorm – це текстовий редактор для програмного коду, виготовлений JetBrains. WebStorm забезпечує надійний, швидкий і гнучкий статичний аналіз коду. Цей аналіз виявляє помилки мови та виконання та пропонує виправлення та покращення. Він також індексує весь ваш проект і може, наприклад, виявити всі невикористані методи, змінні тощо, на рисунку 3.3 зображено роботу rider.

WebStorm надає всі функції для складного робочого процесу git з коробки. Ви можете фіксувати файли, переглядати зміни та вирішувати

конфлікти за допомогою інструменту візуального розрізнення/злиття в IDE. Система дозволяє налаштовувати кольорову гаму, як бажає користувач, також дозволяється налаштувати комбінацію клавіш, що надає можливість за допомогою комбінацій створювати наприклад контролери або конструктор у класа. Також дозволяється встановлювати розширення, для взаємодії з різними фрейм ворками і містити допоміжні методи.

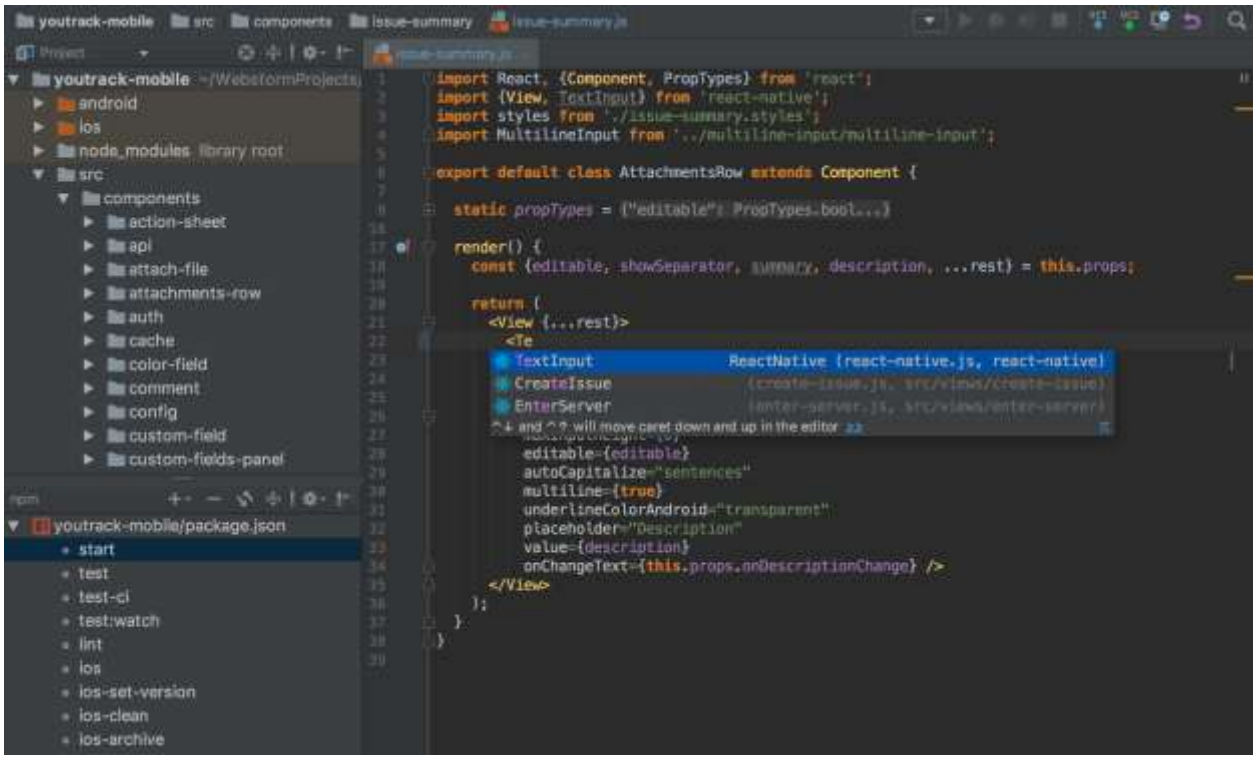


Рисунок 3.3 – Середовище розробки WebStorm

3.2 Програмна реалізація системи менеджменту стартапами

Клієнтську частину інформаційної системи як було проаналізовано, доцільно розробити на cshtml з бібліотеки ASP.NET Core та react. Для відпрацювання https запитів варто реалізувати API, використовуючи Net Core WebApi, що наддасть гнучкість системі.

Було створено загальний репозиторій для роботи з сутностями бази даних та реалізовано для конкретних сутностей. Робота контролерів буде працювати з використанням паттерну MVC (Model-View-Controller), у даному

патерні буде реалізовано зв'язок представлення та колнтролера. За для полегшення розробки буде використано swager, що дозволить графічно відобразити точки виклику Api, модель роботи зображено проекта зображено на рисунку 3.4.

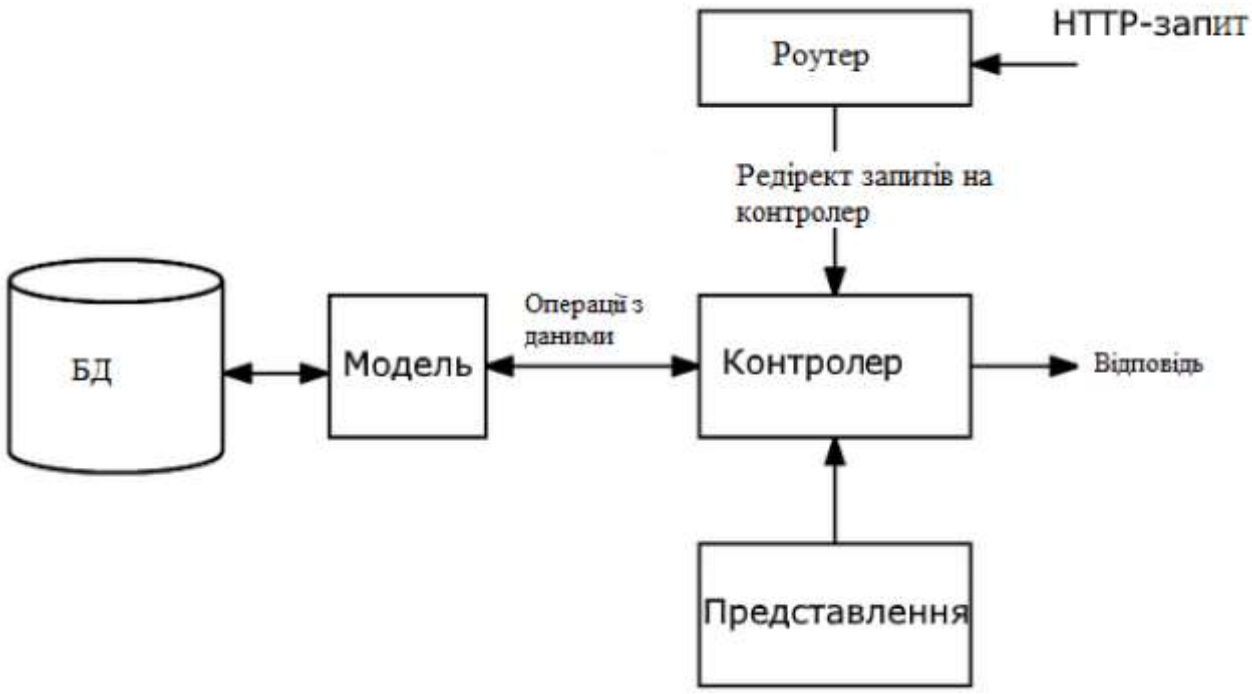


Рисунок 3.4 – Схема MVC-моделі роботи програмного модуля

У клієнтській частині додатку буде створюватись інша модель ніж у контролерах, що наддасть гнучкості системі та полегшить майбутнє розширення системи. У react проекті буде використовуватись typescript, що дозволить добавити типізації до клієнтської частини. На клієнті буде зберігатись вся інформація у store з використання redux, для цього встановлено rtk query, на рисунку 3.5 зображено state додатку. Даний підхід дозволить кешувати дані та дозволить запобігти не потрібним запитам до серверної частини, дана бібліотека працює на основі redux та розроблялась саме для react.

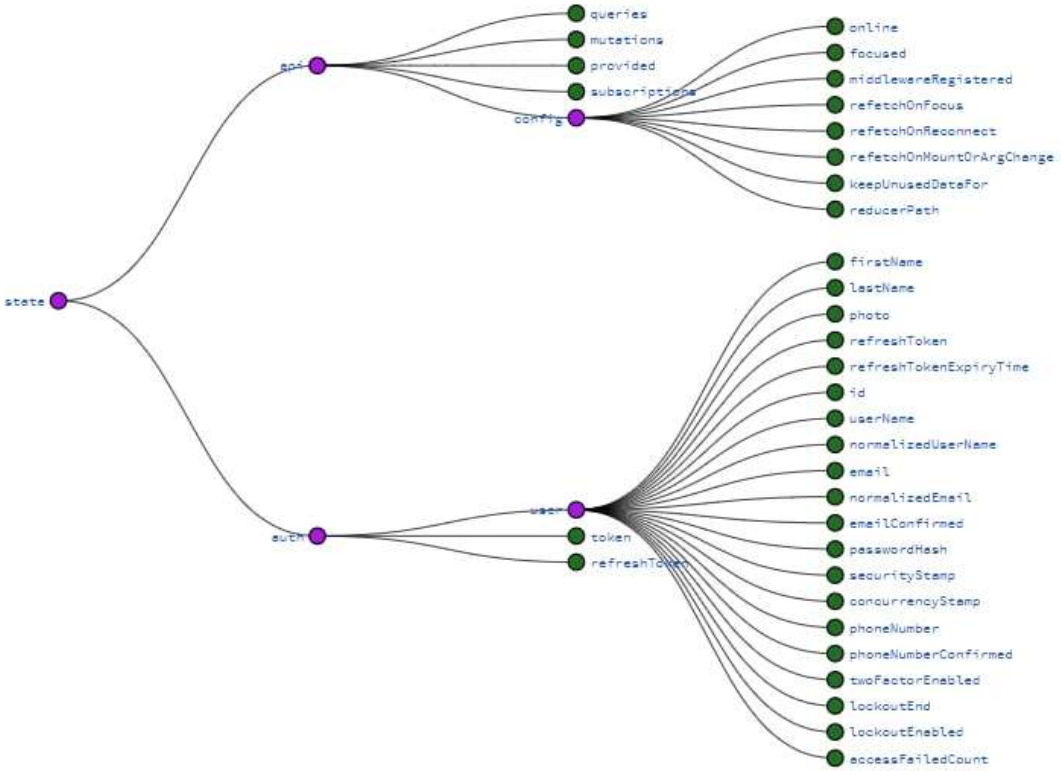


Рисунок 3.5 – Схема функціонування програмного модуля Rtk-query

Серверну частину доцільно будувати на основі підходу REST (Representational State Transfer або передача репрезентативного стану), що регламентує обмеження для Api.

Даний принцип має свої головні принципи:

1. Уніфікований інтерфейс. Кожен інтерфейс має унікально ідентифікувати кожен ресурс, що залучений до взаємодії. Ресурси мають бути однозначно ідентифіковані за допомогою однієї URL-адреси, і тільки за допомогою основних методів мережевого протоколу, таких як DELETE, PUT і GET з HTTP, можна маніпулювати ресурсом.

2. Розділення клієнт-сервер. За архітектури REST клієнт і сервер можуть взаємодіяти лише одним способом: клієнт надсилає запит серверу, а потім сервер надсилає відповідь клієнту. Сервери не можуть робити запити, а клієнти не можуть відповідати — усі взаємодії ініціюються клієнтом.

3. Stateless. Усі виклики з REST API мають бути без стану. Це означає, що кожна взаємодія є незалежною, і кожен запит і відповідь надають всю інформацію, необхідну для завершення взаємодії. Кожен запит клієнта інтерпретується сервером як абсолютно новий запит — сервер нічого не пам’ятає про попередні запити.

RESTful архітектуру поділяють на певні системи, або структури функціювання [23–25]:

1. Система даних. Системи між собою спілкуються на основі описаних інтерфейсів, що дозволяє відправляти та взаємодіяти з різними клієнтами на основі однакових моделей. Формати даних передачі інформації визначаються динамічно, відповідно до потреб клієнтського застосунка.

2. Система обробки даних. Дані передаються на сервери де з json формату за допомогою Asp WEB API формується модель, та автоматично підставляються дані. Систему обробки інформації потрібно робити незалежною від клієнтської частини, що наддасть змогу розширювати додаток. REST дозволяє керувати ресурсами та працювати наприклад з динамічними даними, на рисунку 3.6 зображено приклад RESTful api.

Рисунок 3.6 –RESTful api для авторизації у технології управління

Також слід зазначити, що хороший додаток розпочинається з архітектури та чистого коду. Система буде будуватись на основі принципів SOLID. Дані

принципи визначають як об’єднати функції і структури даних в класи та як повинні взаємодіяти класи між собою. Головна ціль даного патерна зробити додаток можливим до розширення, простим і зрозумілим для подальших робіт.

Вищенаведений патерн засновується на п`яти принципах:

1. SRP Single Responsibility Principle – стверджує що кожен програмний модуль має відповідати лише за одну задачу та вирішувати лише одну проблему та скривати реалізацію.

2. OCP Open –Closed Principle – твердження сформовано Бетраном Мейєром і визначає що модуль має бути закритим до змін та відкритим до розширення.

3. LSP Liskov Substitution Principle – стверджує що об’єкти підкласів повинні мати можливість підміняти об’єкти суперкласу. Що дозволяє зробити код більш чистим та зрозумілим.

4. ISP Interface Segregation Principle – Роберт Мартін стверджує у даному принципі, що клієнт не повинен мати залежності від методів, що вони використовують, отож якщо сервіс використовує абстракцію, що має сукупність не використовую чого функціоналу, варто її розділити.

5. DIP Dependency Inversion Principle – даний принцип стверджує, що сервіси високого рівня не мають залежати від реалізації сервісів низького рівня, а від їх абстракції.

3.3 Тестування програмного забезпечення та аналіз отриманих результатів

Розроблена інформаційну систему було перевірено на відповідність до поставлених вимог та характеристик, що дозволяють менеджеру створювати стартап та керувати ним підчас роботи.

Під час завантаження додатку, користувач потрапляє на вікно авторизації рисунок 3.7.

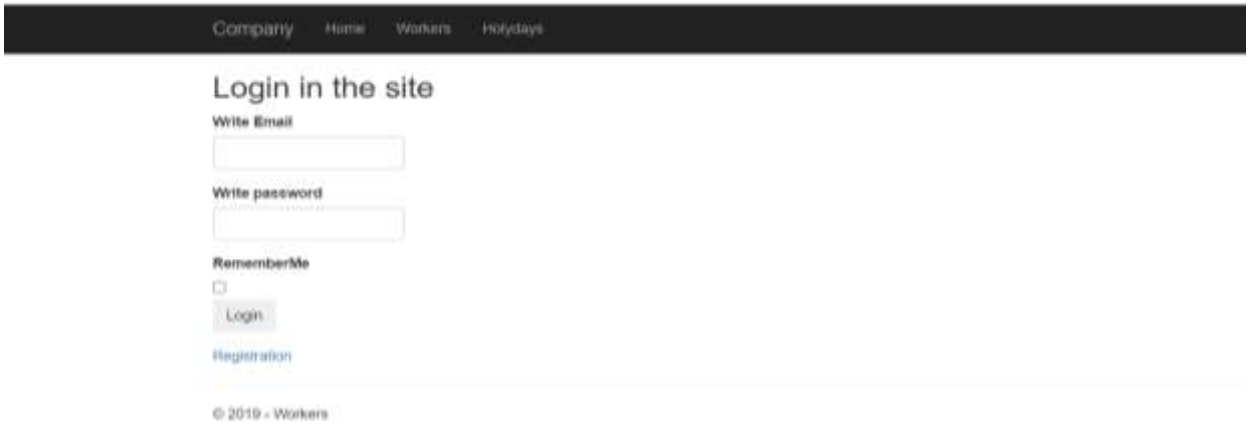


Рисунок 3.7 – Інтерфейс авторизації

Якщо додаток перевіривши, визначає що користувача не існує, система радить користувачу зареєструватись рисунок 3.8.

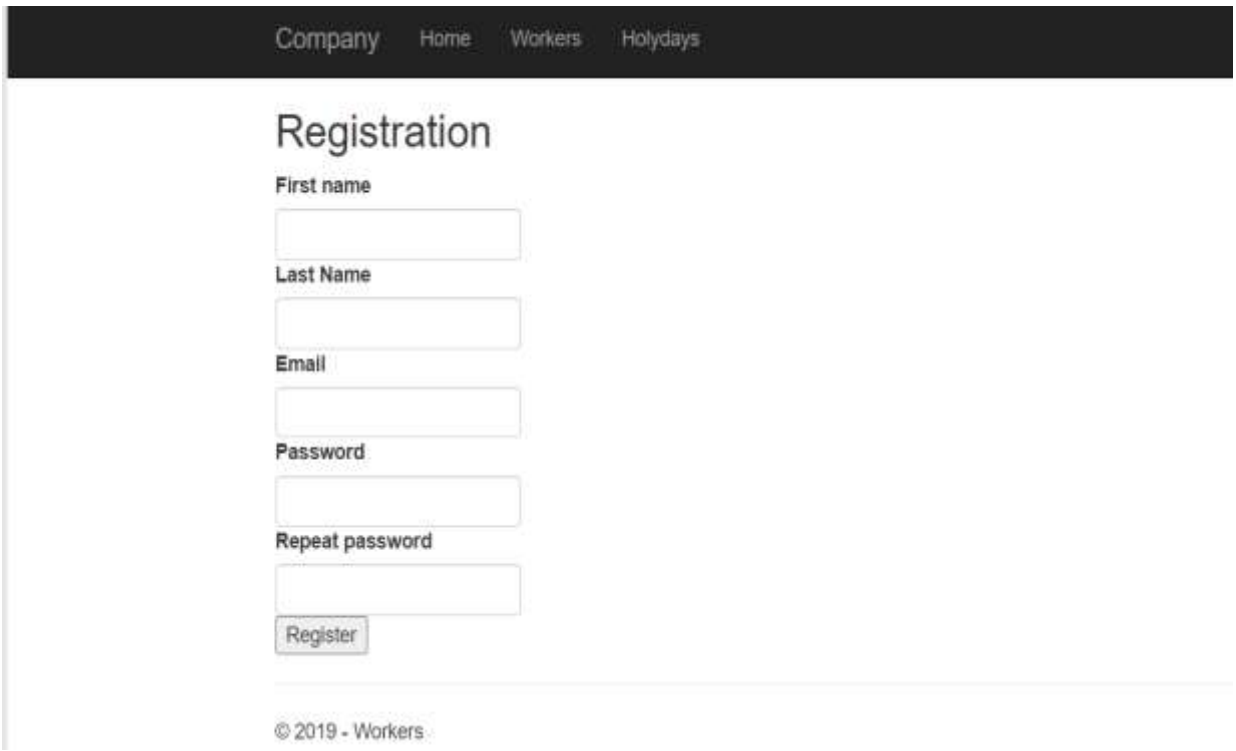


Рисунок 3.8 – Форма реєстрації користувачів

Після авторизації, користувач бачить різні графіки виконих робіт та відпусток, що наведено на рисунку 3.9. Користувач може відфільтрувати список по команді або вибраним користувачам, також він має змогу передвинути відбустку, змінивши її. Створивши відпустку у адміністратора з’являється повідомлення про неї, також адмінстратор має змогу видаляти працівників або змінювати інформацію про них, наведено на рисунку 3.10.

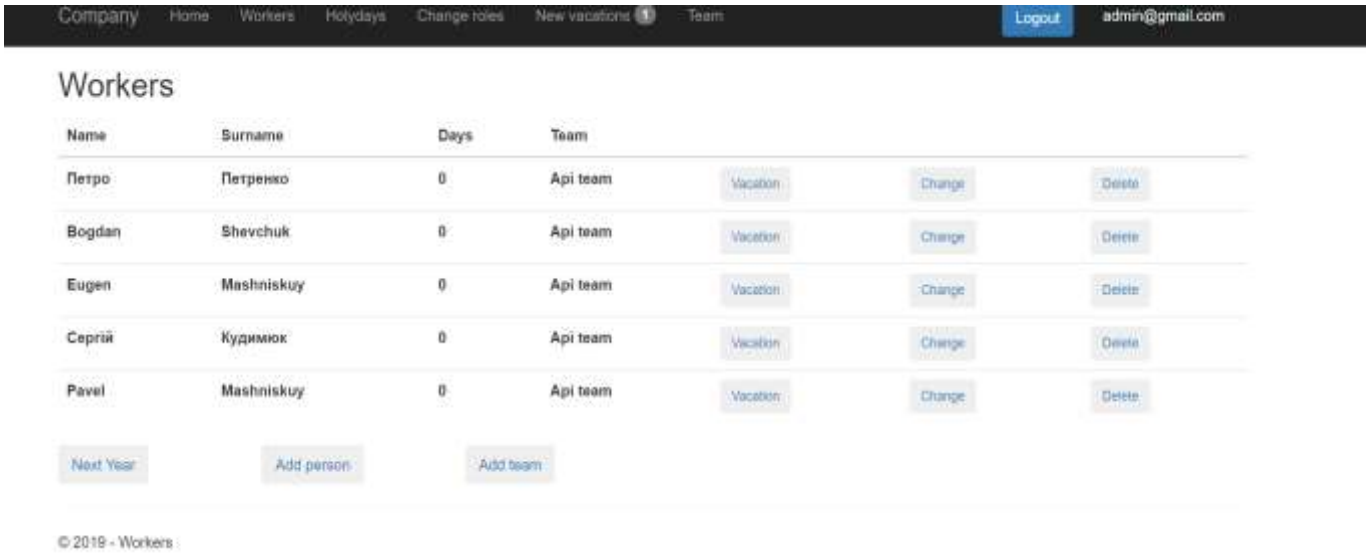


Рисунок 3.9 – Список працівників та інформація

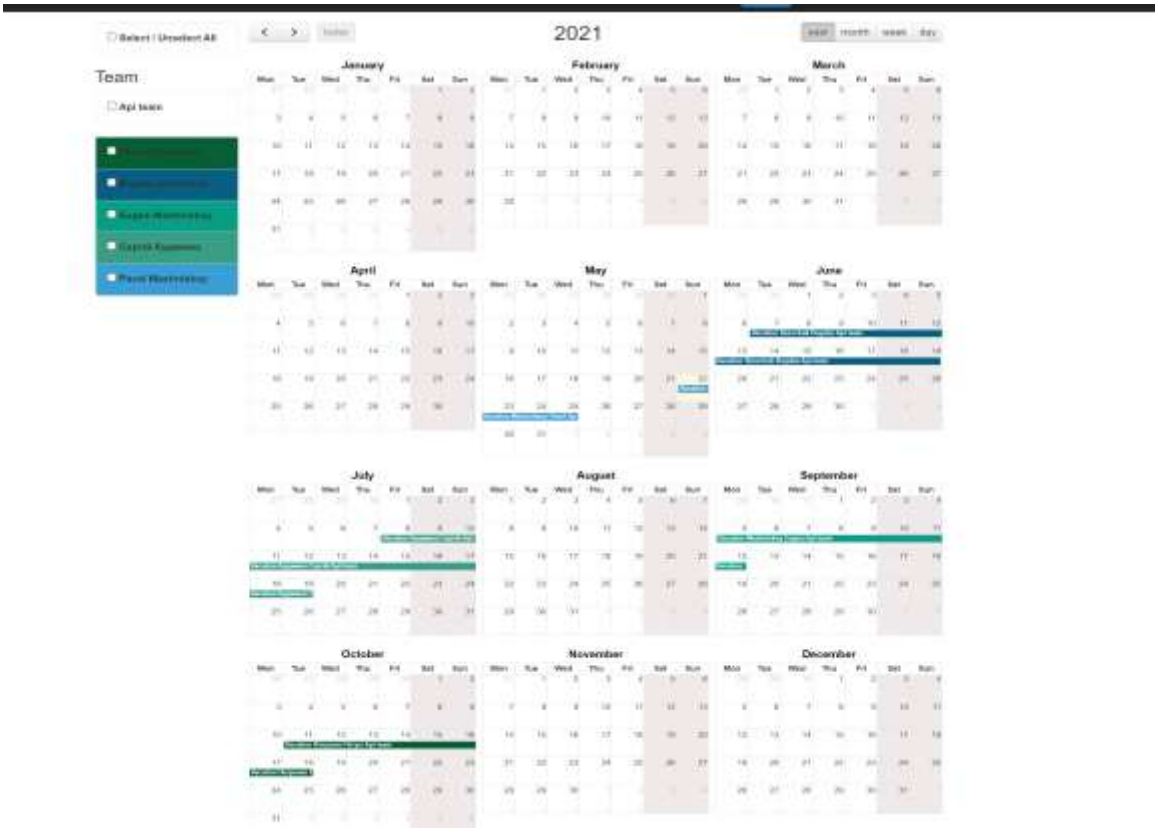


Рисунок 3.10 – Графік робіт та відпусток

Кожен адміністратор має змогу підтвердити або відмовити у відпусткі робітнику. Додаток перевіряє чи може працівник взяти відпустку, якщо не може, то начальник відділу отримає інформацію про це і не дозволить забронювати її. Запропонована система містить функціонал длч управління завданням, трекінг виконаних завдань наведено на рисунку 3.11.

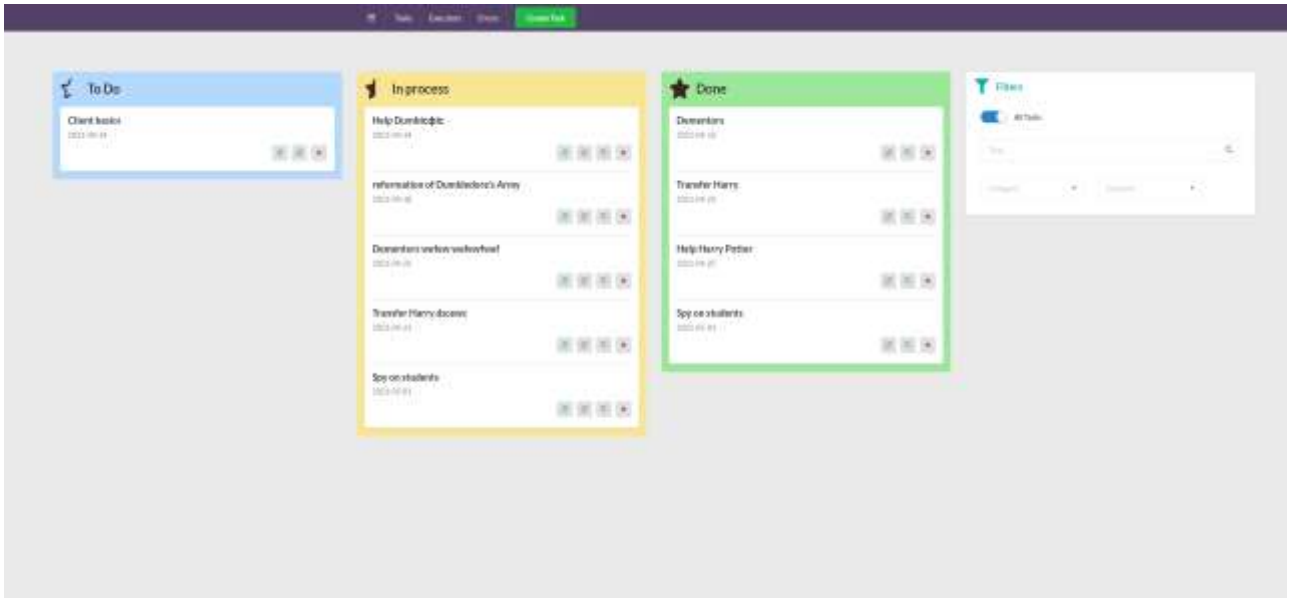


Рисунок 3.11 – Графік виконаних робіт

Додаток має можливість зробити звіт по виконаних роботах, а саме навести графіки залежності виконаних робіт від годин рисунок 3.12.

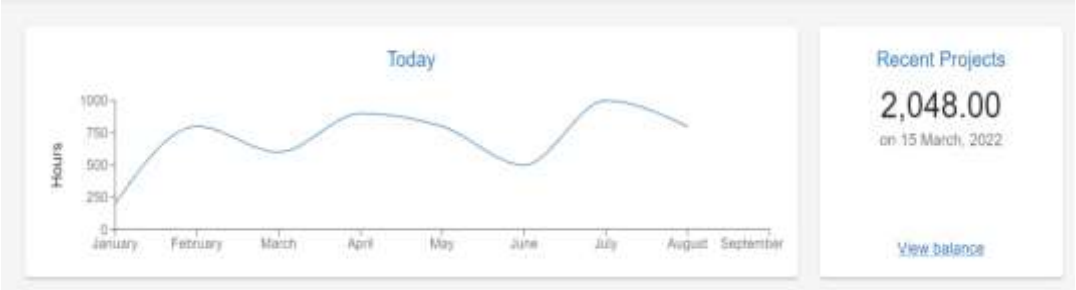


Рисунок 3.12 – Графік залежності виконаних робіт від години

Також є можливість згенерувати звіт потрачених годин по кожному працівнику у проекті, рисунок 3.13.

Рисунок 3.13 – Графік залежності виконаних робіт для години

3.4 Висновок до розділу 3

Обґрунтовано вибір мови програмування C# для розробки інформаційної технології для управління стартапів, адже дана мова містить важливі інструментарії для створення даної техгології, а саме ASP .NET CORE, фреймворк, що надає можливість швидко розробляти веб-застосунки.

Здійснено вибір середовища розробки Rider для роботи з серверною частиною, система містить інтерактивну можливість роботи з базою даних та має можливість розширятись. WebStorm було обрано для розробки клієнтської частини. Дані технології надають можливість створювати гнучкі та ефективні застосунки.

Розроблено інформаційну технологію для управління стаптапом із застосуванням новітніх технологій. Відокремлено клієнтську частину та серверну, що дозволило застосовувати серверну частину в інших системах, використовуючи API запити.

Проілюстровано приклади роботи управління стартапом у різних життєвих ситуаціях.

Здійснено тестування розробленого програмного забеспечення, яке показало правильність виконання поставлених задач та відповідність поставленим технічним вимогам. Доцільно зауважи, що було використано unit-тестування, яке допомогло уникнути неочікуваних помилок на етапі розробки.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1Комерційний та технологічний аудит науково-технічної розробки

Метою даного розділу є проведення технологічного аудиту, в даному випадку нового програмного продукту Інформаційна система для управління стартапом. Дана система має можливості управління проектом і допомагає менеджеру спостерігати за прогресом реалізації стартапу. Особливістю програми є те, що розроблена інформаційна система надає можливість більш ефективно керувати стартапом за допомогою використання експертної системи.

Аналогом може бути Basic Monday – оплата 3500 на рік, Pro Monday – оплата 7000 на рік.

Для проведення комерційного та технологічного аудиту залучають не менше 3-х незалежних експертів. Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, у відповідності із табл. 4.1.

Таблиця 4.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздат-ність продук-ту в реаль-них умовах

Продовження табл. 4.1

Ринкові переваги					
2	Багато аналогів на малому ринку	Ринкові п Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практик на здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

Продовження табл. 4.1

11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Усі дані по кожному параметру занесено в таблиці 4.2

Таблиця 4.2 – Результати оцінювання комерційного потенціалу розробки

Критерії оцінювання	ПІБ експертів		
	Експерт 1	Експерт 2	Експерт 3
	Бали		
Технічна здійсненність концепції	4	3	3
Наявність аналогів на ринку	4	4	4
Цінова політика	4	4	4
Технічні та споживчі властивості виробу	4	3	4
Експлуатаційні витрати	3	4	3
Ринок збуту	4	3	4
Конкурентоспроможність	3	4	3
Фахівці з технічної і комерційної реалізації	4	3	4
Фінансування	4	4	3
Матеріально-технічна база	3	3	3
Термін реалізації ідеї	3	3	3
Супровідна документація	3	3	4
Сума	43	41	42
Середньоарифметична сума балів	(43+41+42) / 3 = 42		

За даними таблиці 4.2 можна зробити висновок щодо рівня комерційного потенціалу даної розробки. Для цього доцільно скористатись рекомендаціями, наведеними в таблиці 4.3.

Таблиця 4.3 - Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

Як видно з таблиці, рівень комерційного потенціалу розроблюваного нового програмного продукту є високим, так як велика кількість чудових стартапів закривається, через погане управління, отож потрібно розробити інформаційну систему, яка буде допомагати управлінню проектом. Розроблена інформаційна система надає можливість більш ефективно керувати стартапом за допомогою експертної системи.

4.2 Прогнозування витрат на виконання науково-дослідної (дослідно-конструкторської) роботи

4.2.1 Основна заробітна плата розробників, яка розраховується за формулою:

$$З_о = \frac{M}{T_р} \cdot t \text{ ,} \qquad (4.1)$$

де М – місячний посадовий оклад конкретного розробника (дослідника), грн.;

$T_р$ – число робочих днів в місяці, 22 днів;

t – число днів роботи розробника (дослідника).

Результати розрахунків зведемо до таблиці 4.1.

Таблиця 4.1 – Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту	34000	1545,45	34	52545,455
Інженер	26000	1181,82	34	40181,818
Всього				92727,27

Так як в даному випадку розробляється програмний продукт, то розробник виступає одночасно і основним робітником, і тестувальником розроблюваного програмного продукту.

4.2.2 Додаткова заробітна плата розробників, які приймали участь в розробці обладнання.

Додаткова заробітна плата прийнято розраховувати як 11,5 % від основної заробітної плати розробників та робітників:

$$З_д = З_о \cdot 11,5\ \% / 100\ \% \qquad (4.2)$$

$З_д = (92727,27 \cdot 11,5\ \% / 100\ \%) = 10663,64$ (грн.)

4.2.3 Нарахування на заробітну плату розробників.

Згідно діючого законодавства нарахування на заробітну плату складають 22 % від суми основної та додаткової заробітної плати.

$$Н_з = (З_о + З_д) \cdot 22\ \% / 100\% \qquad (4.3)$$

$Н_з = (92727,27 + 10663,64) \cdot 22\ \% / 100\ \% = 22746,00$ (грн.)

4.2.4. Оскільки для розроблювального пристрою не потрібно витрачати матеріали та комплектуючи, то витрати на матеріали і комплектуючі дорівнюють нулю.

4.2.5 Амортизація обладнання, яке використовувалось для проведення розробки.

Амортизація обладнання, що використовувалось для розробки в спрощеному вигляді амортизація обладнання, що використовувалась для розробки розраховується за формулою:

$$A = \frac{Ц}{Тв} \cdot \frac{t_{вик}}{12} \text{ [грн.].} \quad (4.4)$$

де Ц – балансова вартість обладнання, грн.;

Т – термін корисного використання обладнання згідно податкового законодавства, років

$t_{вик}$ – термін використання під час розробки, місяців

Розрахуємо, для прикладу, амортизаційні витрати на комп'ютер балансова вартість якого становить 50400 грн., термін його корисного використання згідно податкового законодавства – 2 роки, а термін його фактичного використання – 1,55 міс.

$$A_{обл} = \frac{50400}{2} \times \frac{1{,}55}{12} = 3245{,}455 \; грн.$$

Аналогічно визначаємо амортизаційні витрати на інше обладнання та приміщення. Розрахунки заносимо до таблиці 4.2. Для розрахунку амортизації нематеріальних ресурсів використовується формула:

$$A_{н.р.} = Ц_{н.р.} * Н_a * \frac{t_{вик.}}{12} \quad (4.5)$$

Але, так як вартість ліцензійної ОС та спеціалізованих ліцензійних нематеріальних ресурсів менше 20000 грн, то даний нематеріальний актив не амортизується, а його вартість включається у вартість розробки повністю, $B_{нем.ак.}$ = 16500 грн.

Таблиця 4.2 – Амортизаційні відрахування матеріальних і нематеріальних ресурсів для розробників

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн.
Комп'ютер та комп'ютерна переферія	50400	2	1,55	3245,455
Офісне обладнення	23700	4	1,55	763,068
Приміщення	800000	20	1,55	5151,515
Всього				9160,04

4.2.6 Тарифи на електроенергію для непобутових споживачів (промислових підприємств) відрізняються від тарифів на електроенергію для населення. При цьому тарифи на розподіл електроенергії у різних постачальників (енергорозподільних компаній), будуть різними. Крім того, розмір тарифу залежить від класу напруги (1-й або 2-й клас). Тарифи на розподіл електроенергії для всіх енергорозподільних компаній встановлює Національна комісія з регулювання енергетики і комунальних послуг (НКРЕКП). Витрати на силову електроенергію розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п, \quad (4.6)$$

де В – вартість 1 кВт-години електроенергії для 1 класу підприємства, В = 6,2 грн./кВт;

П – встановлена потужність обладнання, кВт. П = 0,5 кВт;

Ф – фактична кількість годин роботи обладнання, годин.

$К_п$ – коефіцієнт використання потужності, $К_п$ = 0,9.

$В_е = 0,9 \cdot 0,5 \cdot 8 \cdot 34 \cdot 6,2 = 758,88$ (грн.)

4.2.7 Інші витрати та загальновиробничі витрати.

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками. Витрати за статтею «Інші витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.7)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

$I_в = 92727,27 * 100\% / 100\% = 92727,27$ (грн.)

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін. Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників:

$$Н_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.8)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$Н_{нзв}$ = 92727,27 * 100 % / 100 % = 92727 (грн.)

4.2.8 Витрати на проведення науково-дослідної роботи.

Сума всіх попередніх статей витрат дає загальні витрати на проведення науково-дослідної роботи:

$В_{заг}$ = 92727,27+10663,64+22746,00+9160,04+16500+758,88+92727,27+
+92727 = 338010,37 грн.

4.2.9 Розрахунок загальних витрат на науково-дослідну (науково-технічну) роботу та оформлення її результатів.

Загальні витрати на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються *ЗВ*, визначається за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} \text{ (грн)}, \quad (4.9)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи.

Так, якщо науково-технічна розробка знаходиться на стадії: науково-дослідних робіт, то η=0,1; технічного проектування, то η=0,2; розробки конструкторської документації, то η=0,3; розробки технологій, то η=0,4; розробки дослідного зразка, то η=0,5; розробки промислового зразка, то η=0,7;

впровадження, то η=0,9. Оберемо η = 0,5, так як розробка, на даний момент, знаходиться на стадії дослідного зразка:

$ЗВ$ = 338010,37 / 0,5 = 676021 грн.

4.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнювальним позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку. Саме зростання чистого прибутку забезпечить потенційному інвестору надходження додаткових коштів, дозволить покращити фінансові результати його діяльності, підвищить конкурентоспроможність та може позитивно вплинути на ухвалення рішення щодо комерціалізації цієї розробки.

Для того, щоб розрахувати можливе зростання чистого прибутку у потенційного інвестора від можливого впровадження науково-технічної розробки необхідно:

а) вказати, з якого часу можуть бути впроваджені результати науково-технічної розробки;

б) зазначити, протягом скількох років після впровадження цієї науково-технічної розробки очікуються основні позитивні результати для потенційного інвестора (наприклад, протягом 3-х років після її впровадження);

в) кількісно оцінити величину існуючого та майбутнього попиту на цю або аналогічні чи подібні науково-технічні розробки та назвати основних суб’єктів (зацікавлених осіб) цього попиту;

г) визначити ціну реалізації на ринку науково-технічних розробок з аналогічними чи подібними функціями.

При розрахунку економічної ефективності потрібно обов’язково враховувати зміну вартості грошей у часі, оскільки від вкладення інвестицій до отримання прибутку минає чимало часу. При оцінюванні ефективності інноваційних проектів передбачається розрахунок таких важливих показників:

- абсолютного економічного ефекту (чистого дисконтованого доходу);
- внутрішньої економічної дохідності (внутрішньої норми дохідності);
- терміну окупності (дисконтованого терміну окупності).

Аналізуючи напрямки проведення науково-технічних розробок, розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором можна об’єднати, враховуючи визначені ситуації з відповідними умовами.

4.3.1 Розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання масовим споживачем.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

$$\Delta П_i = (\pm \Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (4.10)$$

де $\pm \Delta Ц_o$ – зміна вартості програмного продукту (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу;

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки;

$Ц_о$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки, $Ц_о = Ц_б \pm \Delta Ц_о$;

Цб – вартість програмного продукту у році до впровадження результатів розробки;

ΔN – збільшення кількості споживачів продукту, в аналізовані періоди часу, від покращення його певних характеристик;

λ – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт $\lambda = 0{,}8333$.

p – коефіцієнт, який враховує рентабельність продукту;

ϑ – ставка податку на прибуток, у 2022 році ϑ =18%.

Припустимо, що при прогнозованій ціні 2250 грн. за одиницю виробу, термін збільшення прибутку складе 3 роки. Після завершення розробки і її вдосконалення, можна буде підняти її ціну на *100* грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року – на 5000 шт., протягом другого року – на 3000 шт., протягом третього року на 2000 шт. Домоменту впровадження результатів наукової розробки реалізації продукту не було:

$\Delta П_1 = (0*100 + (2250 + 100)*5000)* 0{,}8333* 0{,}35) * (1 - 0{,}18) = 2690624{,}892$ грн.

$\Delta П_2 = (0*100 + (2250 + 100)*(5000+3000)* 0{,}8333* 0{,}35) * (1 - 0{,}18) = 4496333{,}153$ грн.

$\Delta П_3 = (0*100 + (2250 + 100)*(5000+3000+2000)* 0{,}8333* 0{,}35) * (1 - 0{,}18) = 5620416{,}442$ грн.

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 12807374,49 грн.

4.3.2 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Розраховуємо приведену вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (4.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої науково-дослідної (науково-технічної) роботи, грн;

Т – період часу, протягом якою виявляються результати впровадженої науково-дослідної (науково-технічної) роботи, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0,05...0,15$;

t – період часу (в роках).

Збільшення прибутку ми отримаємо починаючи з першого року:

ПП= (2690624,892/(1+0,1)[1])+(4496333,153/(1+0,1)[2])+(5620416,442/(1+0,1)[3]) = 2446022,63 + 3715977,813 + 4222702,06 = 10384702,5 грн.

Далі розраховують величину початкових інвестицій *PV*, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{інв} * ЗВ, \qquad (4.12)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{інв}=2...5$, але може бути і більшим;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$PV = 2 * 676021 = 1352041{,}49$ *грн.*

Тоді абсолютний економічний ефект $E_{абс}$ або чистий приведений дохід *(NPV, Net Present Value)* для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV, \tag{4.13}$$

$E_{абс} = 10384702{,}5 - 1352041{,}49 = 9032661{,}01$ грн.

Оскільки $E_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів даної науково-дослідної (науково-технічної) роботи може бути доцільним.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність або показник внутрішньої но-рми дохідності *(IRR, Internal Rate of Return)* вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього використаємо формулу:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{4.14}$$

$T_ж$ – життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[3]{(1 + 9032661{,}01/1352041{,}49} - 1 = 0{,}973$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f, \tag{4.15}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2022 році в Україні d = (0,09…0,14);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,5).

$$\tau_{\min} = 0{,}14 + 0{,}05 = 0{,}19.$$

Так як $E_в > \tau_{\min}$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в}, \tag{4.16}$$

$T_{ок}$ = 1 / 0,973 = 1,028 р.

Оскільки $T_{ок}$ < 3-х років, а саме термін окупності рівний 1,028 роки, то фінансування даної наукової розробки є доцільним.

4.4 Висновки до розділу 4

Висновки до розділу: економічна частина даної роботи містить розрахунок витрат на розробку нового програмного продукту, сума яких складає 676021 гривень. Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розраховано період окупності витрат для інвестора та економічний ефект при використанні даної розробки. В результаті аналізу розрахунків можна зробити висновок, що розроблений програмний продукт за ціною дешевший за аналог і є висококонкурентоспроможним. Період окупності складе близько 1,028 роки.

ВИСНОВКИ

У магістерській кваліфікаційній роботі розроблено інформаційну технологію менеджменту стартапами з розширеними функціональними можливостями.

Проаналізовано та обгрунтовано актуальність розробки інформаційної технології управління стартапами. Здійснено аналіз існуючих рішень та детально проаналізовано їх недоліки та сформовано теоретичну базу для покращеного управління стартапами з додатковими функціональними можливостями.

Зроблено огляд та аналіз основних існуючих рішень для менеджменту стартапами. Аналіз зокрема показав, що ActiveCollab система, яка чудово пристосована до управління невелкими проектами, однак не пристосована до великих проектів та проектів, що швидко зростають. Система Monday є одним із найкорисніших інструментів основним недоліком системи є відсутність керування документообігом.

Розроблено структурну схему інформаційної технології управління стартапами. Схема складається з двох модулів: клієнтської частини(відповідає за взаємодію з користувачем та відображення інформації) та веб-сервера (виконує роль обробника інформації та відповідає за зберігання даних), експертної системи.

Створено діаграму діяльності роботи інформаційної технології менеджменту стартапів. Здійснено аналіз архітектурних рівнів та наведено роль кожного рівня взаємодії, а саме приклад взаємодії серверної частини з клієнським застосунком.

Обрано та обгрунтовано систему для керування базами даних MsSql, що надає можливість швидко обробляти великі масиви даних та містить усі переваги реляційної СУБД.

Створено базу даних, зі зв’язками між таблицями. Здійснено нормалізацію бази даних до третьої форми. Перша нормальна форма надала можливість позбутися повторів кортежів у сутностях бази даних інформаційної технології та створено лише прості скалярні атрибути. Друга нормальна форма була досягнута наявністю залежності кожного не ключового атрибуту від первинного ключа. Третя нормальна форма здійснена за рахунок відокремлення всіх не ключових полів, які відносяться до декількох кортежів, у інші таблиці.

Проаналізовано та обґрунтовано вибір мови C#, адже у ній міститься чимала кількість необхідних інструментів для розробки інформаційної технології, а саме ASP .NET CORE, що надає можливість швидко розробляти та впроваджувати додаток. Обґрунтовано вибір середовища розробки Rider для серверної частини та WebStorm для клієнтської частини.

Розроблено інформаційну технологію для управління стартапом із застосуванням новітніх технологій та MVC підходу. Це надає можливість використовувати API без клієнтської частини, шляхом обробки https запитів.

Проілюстровано приклади роботи інформаційної технології управління стартапами у різних життєвих ситуаціях.

Виконано тестування розробленого програмного додатку, що показало правильність роботи та відповідність зазначеним технічним вимогам, доведено підвищення успішності управління стартапом. Протестовано нові функціональні можливості, зокрема діаграм звіти та підсказки недоцільного використання часу.

Доведено економічну доцільність розробки даної технології. В результаті аналізу розрахунків, розроблений програмний продукт за ціною дешевший за аналог і є висококонкурентоспроможним. Витрати на розробку нового програмного продукту складає 676021 гривень. Період окупності складе близько 1,028 роки.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Машницький Є.П., Арсенюк І. Р. Обгрунтування доцільності розробки веб-додатку для менеджменту стартапів [Електронний ресурс] – Режим доступу до ресурсу: https://sci-conf.com.ua/wp-content/uploads/ 2022/12/ EURASIAN-SCIENTIFIC-DISCUSSIONS-18-20.12.22.pdf

2. Хенрік, К. М. Scrum і XP / К.М. Хенрік // Освіта і управління. – 2007. – № 2. – С. 152–158.

3. Pic Є.С, Метод Lean Startup для швидкого тестування ідеї і вибір бізнес-моделі / К.М. Хенрік // Освіта і управління. – 2017. – С. 101–105.

4. Офіційни сайт системи ActiveCollab.[Електронний ресурс]. – Режим доступу: https://activecollab.com/.

5. Офіційни сайт системи Monday[Електронний ресурс]. – Режим доступу: https://monday.com/

6. Офіційни сайт системи BaseCamp.[Електронний ресурс]. – Режим доступу: https://basecamp.com/

7. Збірник «Наука і метрика» Відповідальний за випуск Т. О. Журкович.[Електронний ресурс]. – Режим доступу: https://nim.media/articles/realizatsiyi-doslidnitskikh-innovatsiy-startap-rukh-v-ukrayini.

8. Тренди інтелектуальної економіки [Електронний ресурс]. – Режим доступу: https://kpi-fict-ip32.github.io/Blog/s09/startup_management.html#id1.

9. Основні тенденції розвитку стартапів в Укріїні[Електронний ресурс]. – Режим доступу:http://www.transport–ukraine.eu/page/transportna–strategiya–ukrayini–na–period–do–2020–roku.

10. Поняття стартап. [Електронний ресурс]. – Режим доступу: https://lafounder.com/article/startup.

11. Система моніторингу завдань. [Електронний ресурс]. – Режим доступу: https://www.benishgps.com/ge/products/sistema_ monitoringa/.

12. Управління стартапом. [Електронний ресурс]. – Режим доступу: https://techsisindia.com/solutions/fleet-management/.

13. Роберт С.М. Чиста архітектура / Роберт С.М. / 2019. /С. 56-88.

14. ASP.NET Core WebApi. [Електронний ресурс]. – https://docs.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-3.1

15. Best Practices for Developing Secure Web Applications –[Електронний ресурс]. – https://docs.microsoft.com

16. MySQL по максимуму. – Електронний навчальний посібник / Шварц Б., Зайцев П., Ткаченко В.,– Санкт-Петербург, 2018. – 864с.

17. OpenLayers. [Електронний ресурс]. – Режим доступу: https://leafletjs.com/.

18. C# documentation [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/uk-ua/dotnet/csharp/.

19. React: Up & Running: створення веб-додатків, посібник / Стоян Стефанов. – O'Reilly Media, Inc., 2016. – 222 с.

20. HTML & CSS: розробка та створення веб-сайтів, посібник / Джон Дюкетт. – John Wiley & Sons, 2011. – 512 с.

21. MVC для веб [Електронний ресурс]. – Режим доступу: https://habr.com/ru/post/181772/

22. Паттерни проектування [Електронний ресурс]. – Режим доступу: https://metanit.com/sharp/patterns

23. Azure DevOps [Електронний ресурс]. – Режим доступу: https://azure.microsoft.com/ru-ru/services/devops/

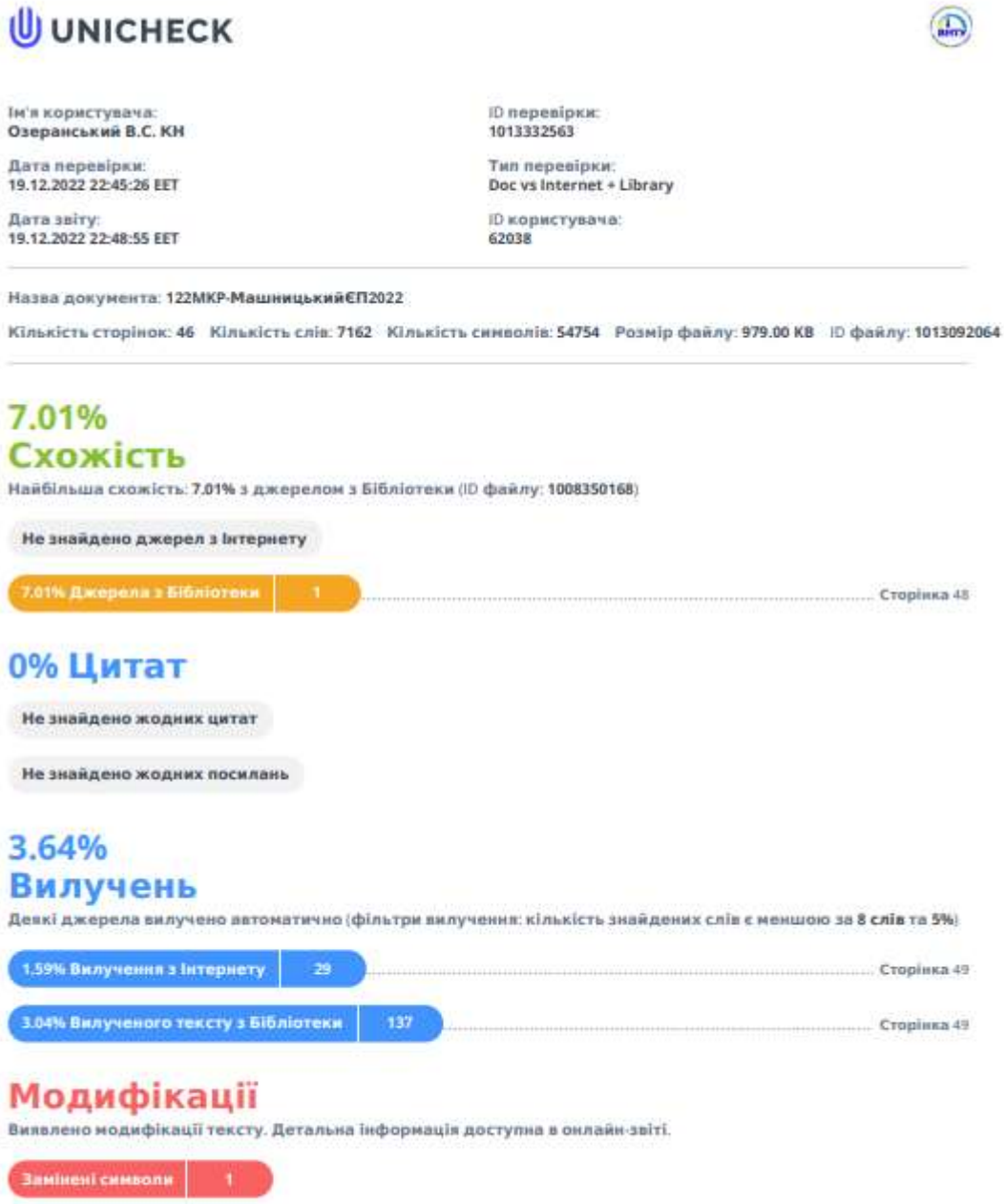
24. Git documentation [Електронний ресурс]. – Режим доступу: https://git-scm.com/

25. Javascript documentation [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/docs/Web/Jascript

ДОДАТКИ

Додаток А (обов’язковий)

Результат перевірки на плагіат в онлайн-системі UNICHECK



Додаток Б (обов'язковий)

Лістинг програми

```
using Domain.Models.Identity;
using Microsoft.AspNetCore.Identity;

namespace Server.Authorization.Service;

public class UserAndRoleDataInitializer
{
public static Task SeedData(IServiceScope scope) =>  SeedData(

scope.ServiceProvider.GetRequiredService<UserManager<ApplicationUser>>(),

scope.ServiceProvider.GetRequiredService<RoleManager<IdentityRole>>());


public static async Task SeedData(UserManager<ApplicationUser>
userManager, RoleManager<IdentityRole> roleManager)
{
await SeedRolesSync(roleManager);
await SeedUsersSync(userManager);
}

private static async Task SeedUsersSync(UserManager<ApplicationUser>
userManager)
{
if (userManager.FindByEmailAsync("test@test").Result == null)
{
var user = new ApplicationUser();
user.UserName = "test@test";
user.Email = "test@test.com";
user.FirstName = "Test";
user.LastName = "Test";

var result = userManager.CreateAsync(user, "P@ssw0rd1!").Result;

if (result.Succeeded)
{
await userManager.AddToRoleAsync(user, Roles.Admin);
}
}
}
```

```
private static async Task SeedRolesSync(RoleManager<IdentityRole>
roleManager)
{
if (!await roleManager.RoleExistsAsync(Roles.Admin))
await roleManager.CreateAsync(new IdentityRole(Roles.Admin));
if (!await roleManager.RoleExistsAsync(Roles.Customer))
await roleManager.CreateAsync(new IdentityRole(Roles.Customer));
if (!await roleManager.RoleExistsAsync(Roles.Developer))
await roleManager.CreateAsync(new IdentityRole(Roles.Developer));
if (!await roleManager.RoleExistsAsync(Roles.ProjectManager))
await roleManager.CreateAsync(new
IdentityRole(Roles.ProjectManager));
}
private static async Task SeedRolesClaimsSync(RoleManager<IdentityRole>
roleManager)
{
if (!await roleManager.RoleExistsAsync(Roles.Admin))
await roleManager.CreateAsync(new IdentityRole(Roles.Admin));
if (!await roleManager.RoleExistsAsync(Roles.Customer))
await roleManager.CreateAsync(new IdentityRole(Roles.Customer));
if (!await roleManager.RoleExistsAsync(Roles.Developer))
await roleManager.CreateAsync(new IdentityRole(Roles.Developer));
if (!await roleManager.RoleExistsAsync(Roles.ProjectManager))
await roleManager.CreateAsync(new
IdentityRole(Roles.ProjectManager));
}
}

using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Security.Cryptography;
using System.Text;
using Domain.Models.Identity;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.IdentityModel.Tokens;
using Server.Authorization;
using Server.Authorization.Models;

namespace Server.Controllers;

[Route("api/[controller]")]
[ApiController]
public class AuthenticateController : ControllerBase
```

```
{
private readonly UserManager<ApplicationUser> _userManager;
private readonly RoleManager<IdentityRole> _roleManager;
private readonly IConfiguration _configuration;

public AuthenticateController(
UserManager<ApplicationUser> userManager,
RoleManager<IdentityRole> roleManager,
IConfiguration configuration)
{
_userManager = userManager;
_roleManager = roleManager;
_configuration = configuration;
}

[HttpPost]
[Route("login")]
public async Task<IActionResult> Login([FromBody] LoginModel model)
{
var user = await _userManager.FindByNameAsync(model.Username);
if (user != null && await _userManager.CheckPasswordAsync(user,
model.Password))
{
var userRoles = await _userManager.GetRolesAsync(user);

var authClaims = new List<Claim>
{
new Claim(ClaimTypes.Name, user.UserName),
new Claim(JwtRegisteredClaimNames.Jti,
Guid.NewGuid().ToString()),
};

foreach (var userRole in userRoles)
{
authClaims.Add(new Claim(ClaimTypes.Role, userRole));
}

var token = CreateToken(authClaims);
var refreshToken = GenerateRefreshToken();

_ = int.TryParse(_configuration["JWT:RefreshTokenValidityInDays"],
out int refreshTokenValidityInDays);

user.RefreshToken = refreshToken;
```

```
user.RefreshTokenExpiryTime =
DateTime.Now.AddDays(refreshTokenValidityInDays);

await _userManager.UpdateAsync(user);

return Ok(new
{
Token = new JwtSecurityTokenHandler().WriteToken(token),
RefreshToken = refreshToken,
User = user,
Expiration = token.ValidTo
});
}

return Unauthorized();
}

[HttpPost]
[Route("register")]
public async Task<IActionResult> Register([FromBody] RegisterModel
model)
{
var userExists = await _userManager.FindByNameAsync(model.Username);

if (userExists != null)
return StatusCode(StatusCodes.Status500InternalServerError,
new ResponseAuth {Status = "Error", Message = "User already
exists!"});

ApplicationUser user = new()
{
Email = model.Email,
SecurityStamp = Guid.NewGuid().ToString(),
FirstName = model.FirstName,
LastName = model.LastName,
UserName = model.Username,
PhoneNumber = model.PhoneNumber,
Photo = model.Photo
};
var result = await _userManager.CreateAsync(user, model.Password);
if (!result.Succeeded)
return BadRequest(result.Errors);
await _userManager.AddToRoleAsync(user, Roles.Developer); // default
roles
```

```
return Ok(new ResponseAuth {Status = "Success", Message = "User created
successfully!"});
}

[HttpPost]
[Route("register-admin")]
public async Task<IActionResult> RegisterAdmin([FromBody] RegisterModel
model)
{
var userExists = await _userManager.FindByNameAsync(model.Username);

if (userExists != null)
return StatusCode(StatusCodes.Status500InternalServerError,
new ResponseAuth {Status = "Error", Message = "User already
exists!"});


ApplicationUser user = new()
{
Email = model.Email,
SecurityStamp = Guid.NewGuid().ToString(),
UserName = model.Username
};
var result = await _userManager.CreateAsync(user, model.Password);

if (!result.Succeeded)
return StatusCode(StatusCodes.Status500InternalServerError,
new ResponseAuth
{Status = "Error", Message = "User creation failed! Please check user
details and try again."});
if (await _roleManager.RoleExistsAsync(Roles.Admin))
{
await _userManager.AddToRoleAsync(user, Roles.Admin);
}
if (await _roleManager.RoleExistsAsync(Roles.Customer))
{
await _userManager.AddToRoleAsync(user, Roles.Customer);
}
if (await _roleManager.RoleExistsAsync(Roles.Developer))
{
await _userManager.AddToRoleAsync(user, Roles.Developer);
}
if (await _roleManager.RoleExistsAsync(Roles.ProjectManager))
{
```

```
        await _userManager.AddToRoleAsync(user, Roles.ProjectManager);
                }

        return Ok(new ResponseAuth {Status = "Success", Message = "User created
                successfully!"});
                }

                [HttpPost]
                [Route("refresh-token")]
public async Task<IActionResult> RefreshToken(TokenModel tokenModel)
                {
                if (tokenModel is null)
                {
                return BadRequest("Invalid client request");
                }

                string? accessToken = tokenModel.AccessToken;
                string? refreshToken = tokenModel.RefreshToken;

        var principal = GetPrincipalFromExpiredToken(accessToken);
                if (principal == null)
                {
                return BadRequest("Invalid access token or refresh token");
                }
                string username = principal.Identity.Name;
        var user = await _userManager.FindByNameAsync(username);

                if (user == null  user.RefreshToken != refreshToken
                user.RefreshTokenExpiryTime <= DateTime.Now)
                {
                return BadRequest("Invalid access token or refresh token");
                }

        var newAccessToken = CreateToken(principal.Claims.ToList());
                var newRefreshToken = GenerateRefreshToken();

                user.RefreshToken = newRefreshToken;
                await _userManager.UpdateAsync(user);

                return new ObjectResult(new
                {
                accessToken = new
JwtSecurityTokenHandler().WriteToken(newAccessToken),
                refreshToken = newRefreshToken
                });
```

```
}

[Authorize]
[HttpPost]
[Route("revoke/{username}")]
public async Task<IActionResult> Revoke(string username)
{
var user = await _userManager.FindByNameAsync(username);

if (user == null) return BadRequest("Invalid user name");

user.RefreshToken = null;
await _userManager.UpdateAsync(user);

return NoContent();
}

[Authorize]
[HttpPost]
[Route("revoke-all")]
public async Task<IActionResult> RevokeAll()
{
var users = _userManager.Users.ToList();
foreach (var user in users)
{
user.RefreshToken = null;
await _userManager.UpdateAsync(user);
}

return NoContent();
}

private JwtSecurityToken CreateToken(List<Claim> authClaims)
{
var authSigningKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(_configuration["JWT:Secret"]
));
_ = int.TryParse(_configuration["JWT:TokenValidityInMinutes"], out int
tokenValidityInMinutes);

var token = new JwtSecurityToken(
issuer: _configuration["JWT:ValidIssuer"],
audience: _configuration["JWT:ValidAudience"],
expires: DateTime.Now.AddMinutes(tokenValidityInMinutes),
claims: authClaims,
```

```
            signingCredentials: new SigningCredentials(authSigningKey,
                SecurityAlgorithms.HmacSha256)
            );

            return token;
        }

        private static string GenerateRefreshToken()
        {
            var randomNumber = new byte[64];
            using var rng = RandomNumberGenerator.Create();
            rng.GetBytes(randomNumber);

            return Convert.ToBase64String(randomNumber);
        }

        private ClaimsPrincipal? GetPrincipalFromExpiredToken(string? token)
        {
            var tokenValidationParameters = new TokenValidationParameters
            {
                ValidateAudience = false,
                ValidateIssuer = false,
                ValidateIssuerSigningKey = true,
                IssuerSigningKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(_configuration["JWT:Secret"]
                )),
                ValidateLifetime = false
            };

            var tokenHandler = new JwtSecurityTokenHandler();
            var principal = tokenHandler.ValidateToken(token,
tokenValidationParameters, out SecurityToken securityToken);

            if (securityToken is not JwtSecurityToken jwtSecurityToken ||
!jwtSecurityToken.Header.Alg.Equals(SecurityAlgorithms.HmacSha256,
                    StringComparison.InvariantCultureIgnoreCase))
                throw new SecurityTokenException("Invalid token");

            return principal;
        }
    }
```

```
import { configureStore, ConfigureStoreOptions} from '@reduxjs/toolkit';
import {TypedUseSelectorHook, useDispatch, useSelector} from 'react-redux';
import {apiMainQuery} from './api/createApi';
```

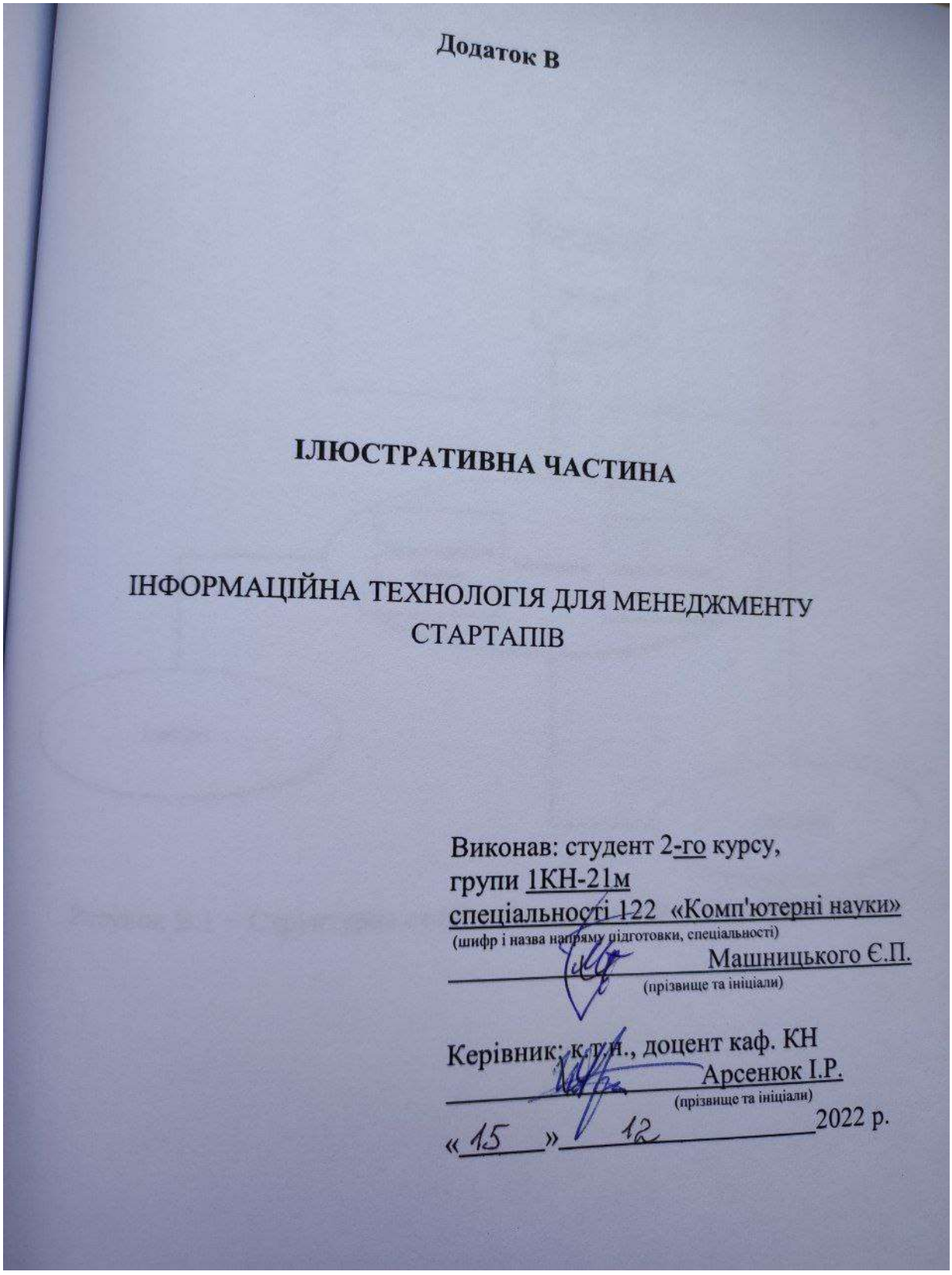
```
import auth from './slice/authSlice';


export const createStore = (
  options?: ConfigureStoreOptions["preloadedState"] | undefined
) =>
  configureStore({
    reducer: {
      [apiMainQuery.reducerPath]: apiMainQuery.reducer,
      auth
    },
    middleware: (getDefaultMiddleware) =>
      getDefaultMiddleware().concat(apiMainQuery.middleware),
    ...options
  });

export const store = createStore();

export type AppDispatch = typeof store.dispatch;
export const useAppDispatch = () => useDispatch<AppDispatch>();
export type RootState = ReturnType<typeof store.getState>;
export const useTypedSelector: TypedUseSelectorHook<RootState> =
  useSelector;
```

Додаток В (обов’язковий)



Додаток В

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ДЛЯ МЕНЕДЖМЕНТУ СТАРТАПІВ

Виконав: студент 2-го курсу,
групи 1КН-21м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
_____ Машницького Є.П.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
_____ Арсенюк І.Р.
(прізвище та ініціали)
«15» 12 2022 р.

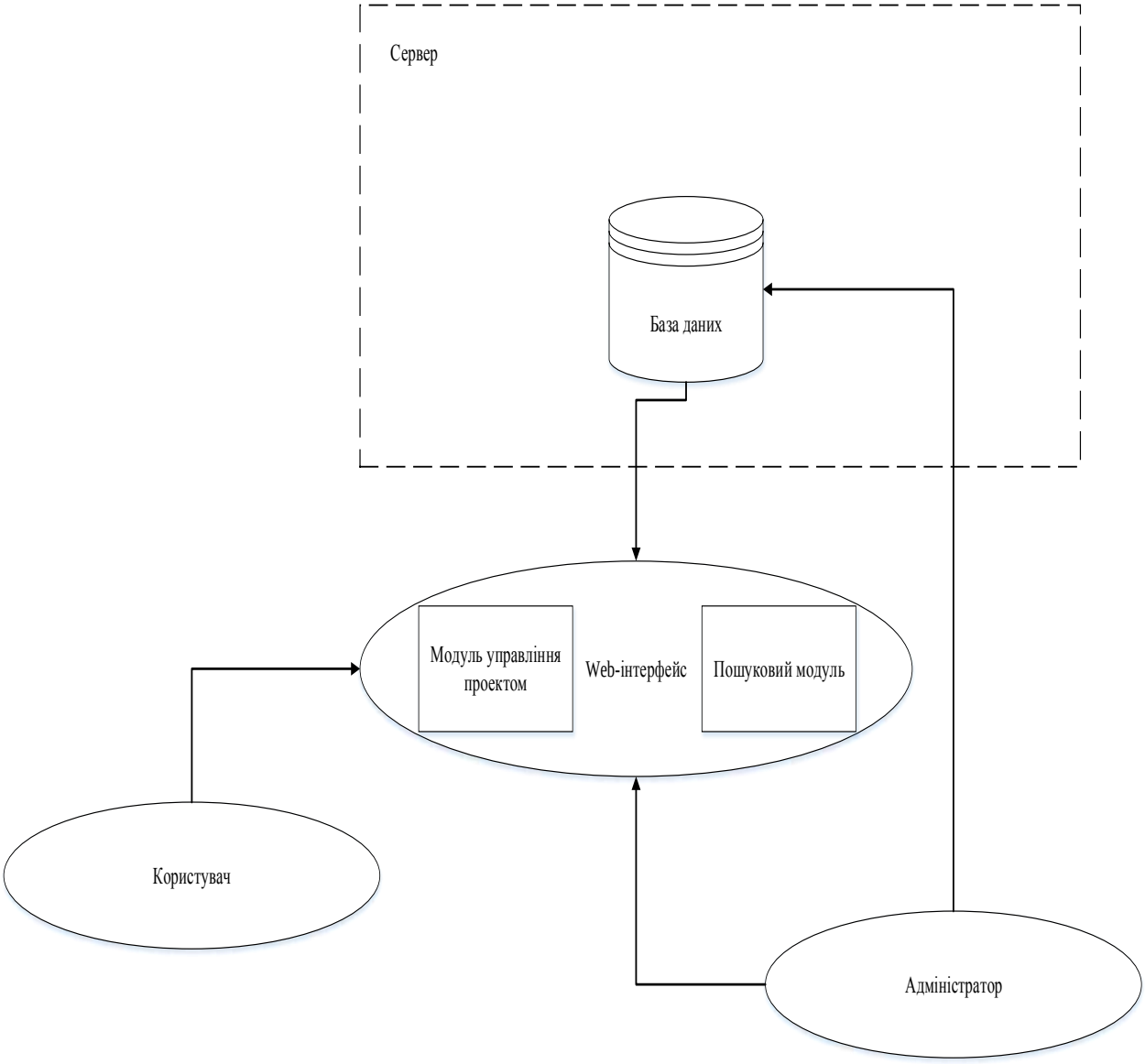


Рисунок В.1 − Структурна схема веб-системи управління стартапом

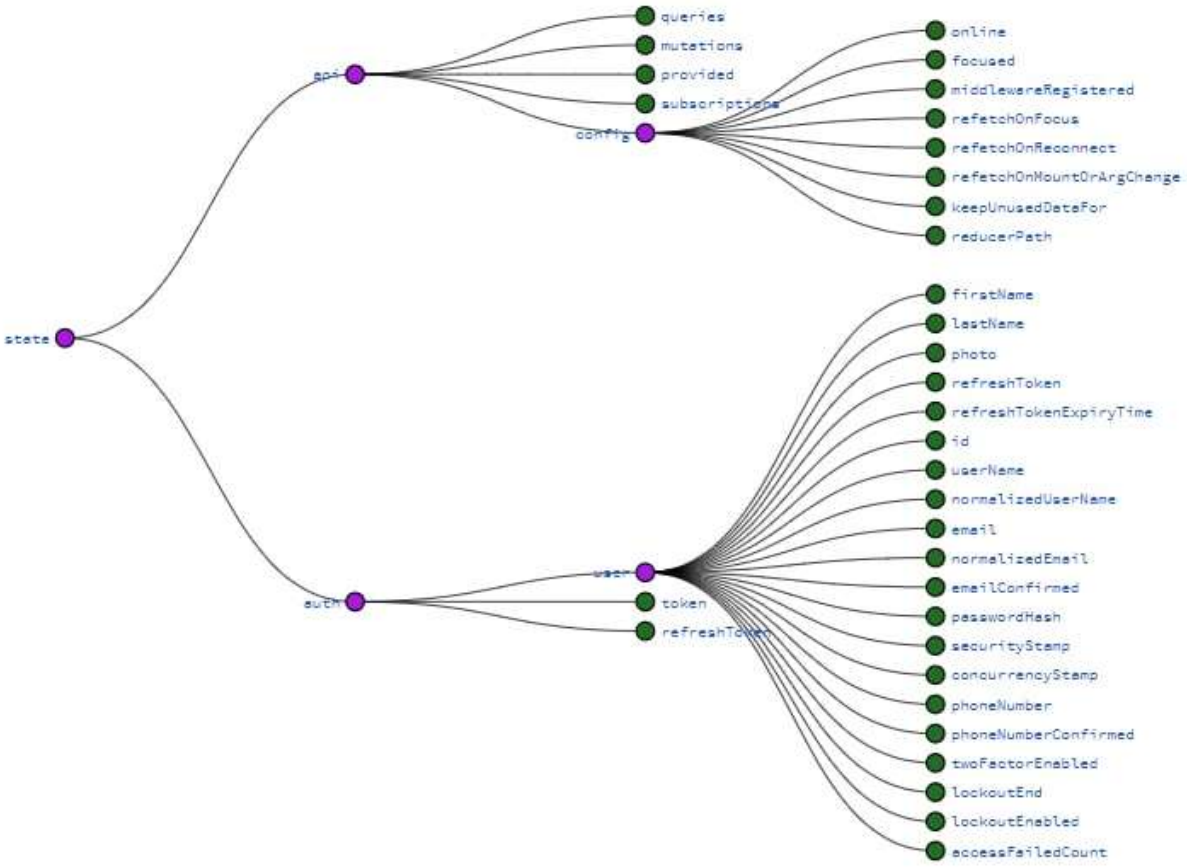


Рисунок В.2 – Схема функціонування програмного модуля Rtk-query

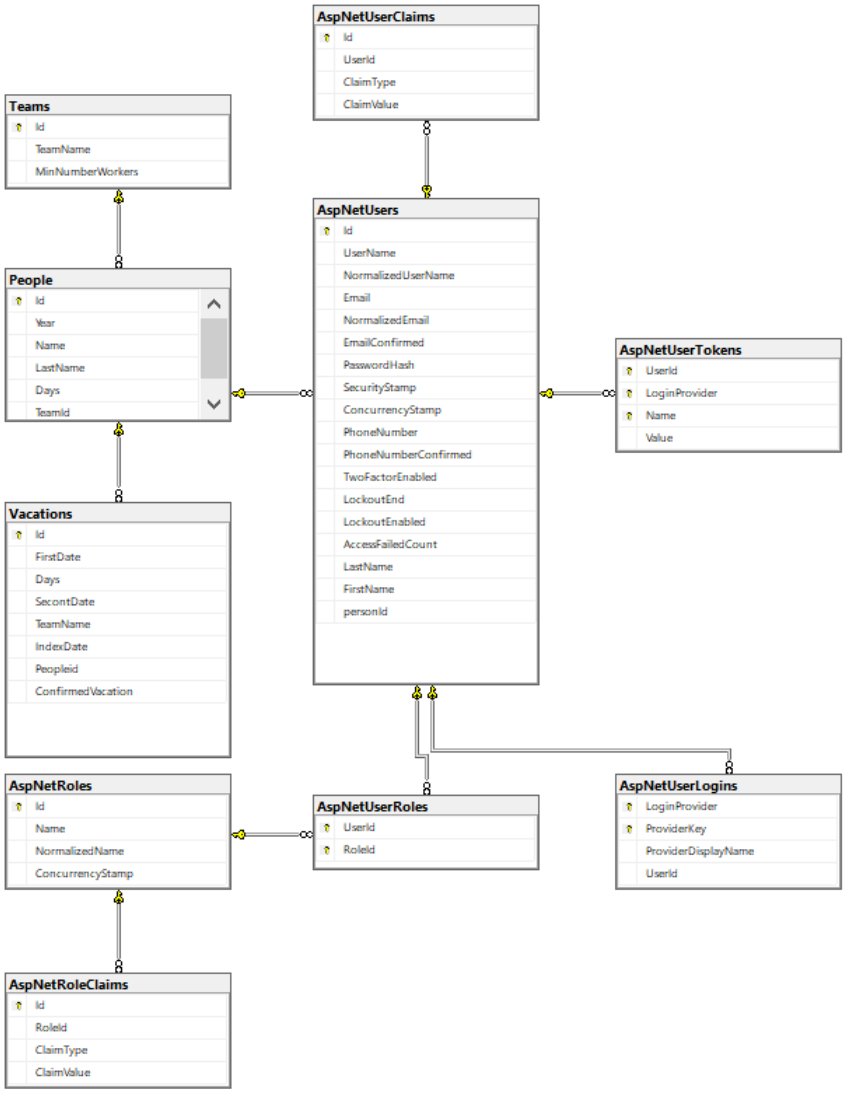


Рисунок В.3 – Загальна схема бази даних роботи модуля

Рисунок В.4 – Діаграма діяльності алгоритму роботи модуля

Додаток Г (довідниковий)

Інструкція користувача

Для запуска розробленої програми управління стартапами потрібно вперейти за посиланням. Стартове вікно програми показано на рис. Г.1. За замовчанням там відображається вікно авторизації.

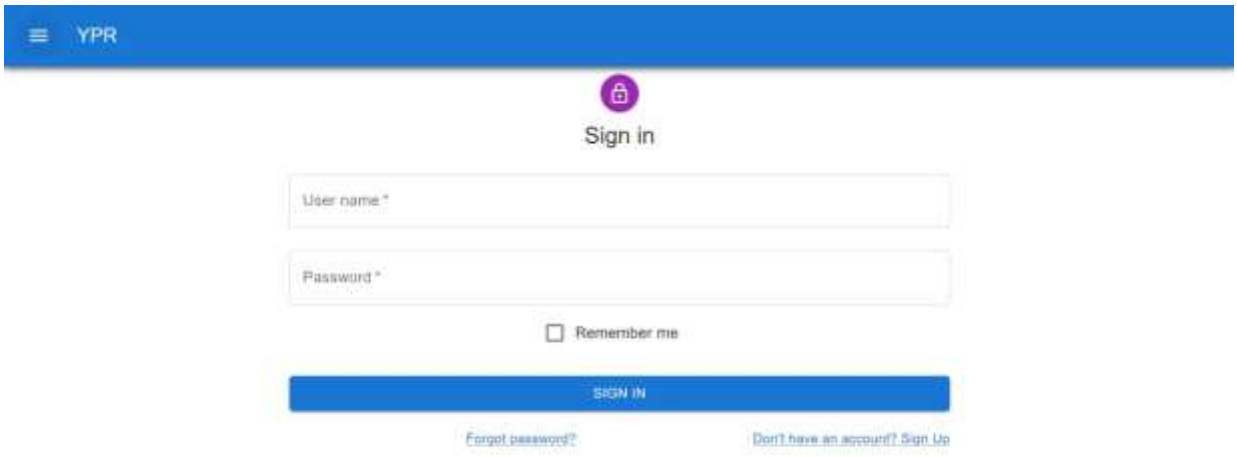


Рисунок Г.1 - Вид стартового вікно програми менеджменту стартапу