

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ І ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
МОТИВАЦІЇ І УПРАВЛІННЯ КОМАНДОЮ РОЗРОБКИ ІТ-СТАРТАПУ.
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS FOR
MOTIVATING AND MANAGING AN IT STARTUP TEAM

Виконав: студент(ка) 2 курсу, групи КНТ-220м
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)

Освітня програма (спеціалізація) Системи
штучного інтелекту

Іхсанов Д.М.

(прізвище та ініціали)

Керівник Дубровін В.І.

(прізвище та ініціали)

Рецензент Касьян М.М.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти магістр
 Спеціальність 122 Комп'ютерні науки
 (код і найменування)
 Освітня програма (спеціалізація) Системи штучного інтелекту
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” _____ 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Іхсанова Дмитра Михайловича
 (прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження і програмна реалізація методів мотивації і управління командою розробки ІТ-стартапу. Research and Software Implementation of Methods for Motivating and Managing an IT Startup Team
 керівник проєкту (роботи) Дубровін Валерій Іванович, канд.техн.наук, проф. ,
 (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
 затверджені наказом закладу вищої освіти від “5” листопада 2021 року № 435
2. Строк подання студентом проєкту (роботи) 20 грудня 2021 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановки завдань дослідження. 2. Матеріали і методи. 3. Розробка архітектури програми. 4. Розробка програмного додатку. 5. Експлуатація, тестування та експериментальне дослідження програми.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Дубровін В.І., професор		
Нормоконтроль	Липовець М.В., асистент		

7. Дата видачі завдання 6 вересня 2021 р.**КАЛЕНДАРНИЙ ПЛАН**

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз проблеми та постановки завдань дослідження.	2–3 тижні	Розділ 1
3	Матеріали і методи.	4-5 тижні	Розділ 2
4	Розробка архітектури програми.	5–6 тижні	Розділ 3
5	Розробка програмного додатку.	7-9 тижні	Розділ 4
6	Експлуатація, тестування та експериментальне дослідження програми.	10-12 тижні	Розділ 5
7	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	13-15 тижні	Додатки
8	Захист роботи.	16 тиждень	

Студент (ка) _____ Іхсанов Д.М.
(підпис) (прізвище та ініціали)Керівник проєкту (роботи) _____ Дубровін В.І.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи магістра:
78 с., 23 рис., 1 табл., 2 дод., 23 джерел.

УПРАВЛІННЯ КОМАНДОЮ, ОЦІНЮВАННЯ РОБОТИ КОМАНДИ, ІТ-СТАРТАП.

Об'єкт дослідження – програмні засоби методів мотивації і управління командою.

Предмет дослідження – методи управління командою розробників.

Мета роботи – створення програмної системи для автоматизації методів мотивації і управління командою розробки ІТ-стартапу.

Матеріали, методи та технічні засоби: мова програмування JavaScript, програмна платформа Node.js та JavaScript фреймворк для створення інтерфейсу користувача Vue.js, реляційна база даних PostgreSQL, персональний комп'ютер з процесором Intel Core i5 під управлінням операційної системи Microsoft Windows 10.

Результати. Проаналізовано особливості стартап-проектів та керування командою розробників. Розглянуто методи керування співробітників у стартап-проекті. Створено програмний веб-застосунок для автоматизації методів мотивації і управління командою розробки ІТ-стартапу.

Висновки. Розроблено веб-застосунок для автоматизації методів мотивації і управління командою розробки ІТ-стартапу, що дозволяє розрахувати ефективність кожного співробітника.

Галузь використання – ІТ-компанії та установи, що мають у своєму складі відділи ІТ.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 78 pages, 23 figures, 1 table, 2 appendixes, 23 sources.

TEAM MANAGEMENT, PERFORMANCE TEAM EVALUATION, IT STARTUP.

The object of study is software tools for motivation methods and team management.

The subject of study is managing methods of a development team.

The purpose of the work is to create a software system for the automation of motivation methods and management of an IT-startup development team.

Materials, methods, and tools: programming languages JavaScript, development platform Node.js with JavaScript framework Vue.js, database PostgreSQL, IBM-compatible personal computer with an Intel Core i5 processor under the control of the Microsoft Windows 10 operating system.

Results. Features of startup projects and management of the development team are analyzed. Methods of employee management in a startup project are considered. Created a software web application to automate methods of motivation and management of an IT-startup development team.

Conclusions. The web application for automation of motivation method and management of an IT-startup development team are developed. It allows to calculate the effectiveness of each employee.

The field of use is IT company and organizations with IT departments.

ЗМІСТ

	С.
Перелік скорочень та умовних познач 8	8
Вступ 9	9
1 Аналіз проблеми та постановки завдань дослідження 10	10
1.1 Аналіз особливостей стартапів 10	10
1.2 Аналіз управління командою розробки 14	14
1.3 Підтримка прийняття рішень в управлінні командою розробки ІТ-стартапу 20	20
1.4 Висновки до розділу 24	24
2 Матеріали і методи 25	25
2.1 Мова програмування JavaScript 25	25
2.2 Каскадна таблиця стилів CSS 26	26
2.3 JavaScript-фрейворк Vue.js 28	28
2.4 Програмна платформа Node.js 30	30
2.5 База даних PostgreSQL 31	31
2.6 Редактор коду Visual Studio Code 33	33
2.7 Висновки до розділу 34	34
3 Розробка архітектури програми 35	35
3.1 Загальний опис та архітектура програмного додатку 35	35
3.2 Функціональні елементи додатку 37	37
3.3 Опис функціональності модулів додатку 38	38
3.4 Висновки до розділу 40	40
4 Розробка програмного додатку 41	41
4.1 Автоматизація управління командою розробки ІТ-стартапу 41	41
4.2 Розробка інтерфейсу додатку 41	41
4.3 Розрахунок ефективності роботи співробітника 43	43
5 Експлуатація, тестування та експериментальне дослідження програми 47	47
5.1 Тестування інтерфейсу додатку 47	47

5.2 Тестування скрипту для пошуку активних і неактивних IP 48

5.3 Тестування сервера.....49

5.4 Симуляція DoS і DDoS атак 49

5.5 Висновок за розділом 50

Висновки..... 51

Перелік джерел посилань 52

Додаток А Текст програми 54

Додаток Б Слайди презентації 73

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
DAE	– Data Envelopment Analysis;
IDE	– Integrated Development Environment;
HR	– Human Resources;
MVC	– Model-View-Controller;
SFA	– Stochastic Frontier Approach;
TFA	– Thick Frontier Approach;
UI	– User Interface;
ПЗ	– програмне забезпечення.

ВСТУП

Слово «стартап» в останні десятиліття стало синонімом прориву, нерозривно пов'язане з новими технологіями [1]. Стартап або стартап-компанія (від англ. Start-up – запускати) – це тип бізнесу, спрямований на отримання доходу шляхом реалізації принципово нової ідеї. На жаль, 90% стартапів терплять поразку ще на перших етапах своєї діяльності [2]. В останні роки все більше і більше компаній розуміють важливість стратегічного управління, і нерідко перед керівником команди з інформаційних технологій (ІТ) ставиться завдання розробити ІТ-стратегію, бо без цього керівник команди буде стикатися з дуже великою кількістю проблем. Як правило, основною причиною проблем стає саме відсутність чітко сформульованої ІТ-стратегії і її ув'язки з цілями бізнесу і суміжними галузями знань, такими як бізнес-аналіз і бізнес-моделювання [3]. Вони - сполучні ланки між бізнесом і ІТ. На цьому факті необхідно акцентувати увагу, так як грамотна ІТ-стратегія впливає з чіткого уявлення практичних завдань управління. Керівник повинен активно аналізувати роботу своєї команди та знати про ефективність роботи кожного члена команди. Завдяки цьому керівник зможе побудувати подальшу стратегію.

За статистичними даними, лише 10% стартапів стають успішними [4]. В умовах нестабільності та конкуренції на міжнародному ринку ІТ послуг процес управління компанією залежить від впливу великої кількості внутрішніх та зовнішніх чинників. Компанії - початківці стикаються з цілою низкою проблем, наприклад [1]-[2]: неправильний підбір персоналу, відсутність бізнес-моделі, незатребуваний на ринку товар, якісне відстеження внутрішніх та зовнішніх ризиків, неправильне використання бюджету.

Таким чином, розробка системи для аналізу ефективності праці членів команди є актуальною проблемою, вирішення якої допоможе командам розробки оптимізувати свою роботу.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКИ ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Аналіз особливостей стартапів

1.1.1 Поняття стартапу

В ІТ стартапами називають нові компанії, що тільки з'являються або плановані до створення. ІТ-стартапи не прив'язані до агресивних маркетингових і піар-методів, спрямованих на швидку розкрутку фірми. ІТ-стартапи можуть створюватися на тривалий термін і з великим терміном виходу на ринок (наприклад, створення продукту протягом двох років і тільки після цього вихід на ринок і просування) [2]. Всесвітньо відома компанія Apple колись теж була стартапом, а сьогодні вона сама щорічно купує стартапи на суму більш ніж 1 мільярд доларів. Google, YouTube, Facebook, Вконтакте, і багато інших всім відомих сервісів теж були стартапами [1]-[3].

Є класифікація стадій розвитку стартапів, згідно з якою стартап проходить в своєму розвитку 5 стадій: посівну стадію (seed stage), стадію запуску (startup stage), стадію зростання (growth stage), стадію розширення (expansion stage) і стадію «виходу» (exit stage) [4].

Стартапи мають свою відмінності від просто нових компаній. Будь-який бізнес - це повторення існуючих основ, нехай навіть з унікальним торговим пропозицією. Це ще одна кав'ярня, магазин, виробництво труби або ще одна послуга по установці вікон. На відміну від звичайного бізнесу, який людина тільки відкриває, стартап має інноваційну основу, тобто відкривається бізнес, який раніше не існував взагалі [4].

Ідея будь-якого стартапу полягає в новизні, новий продукт, нової технології, в тому, чого раніше не було в світі. Відмінність стартапу від бізнесу в тому, що це принципово нова послуга або товар, якої раніше не існувало [5].

Ще стартап відрізняється від звичайного бізнесу тим, що практично відразу має інвесторів, тобто інвестори вкладають гроші в компанію не в той момент, коли вона вже успішно працює, а тільки на самому початку її зародження. Стартап ґрунтується на ідеях, і саме в ці ідеї і інвестори вкладають

гроші. Можна сказати, що стартап – це завжди монополія, так як другого такого бізнесу бути не повинно [6].

1.1.2 Особливості управління командою в сфері ІТ

Менеджмент, як техпроцес, є основним і невід'ємним фактором розвитку проєктів. Менеджмент – це управління. У нашому випадку, управління проєктом. Зрозуміло, що управління проєктом – це робота над його складовими.

Управління проєктами – це мистецтво організації, планування, керівництва, координації трудових, фінансових і матеріально-технічних ресурсів протягом проєктного циклу, спрямоване на ефективне досягнення його цілей шляхом застосування сучасних методів, технік і технологій управління для досягнення визначених у проєкті результатів за складом і обсягом робіт, вартості, часу, якості й задоволення учасників проєкту [7].

Менеджер несе пряму відповідальність за досягнення цілей і за можливі помилки перед замовником. Під керівництвом менеджера проєкту відбувається формування команди проєкту в залежності від складності організації проєкту і покладених на команду зобов'язань. При наборі людей в команду менеджер проєкту повинен враховувати їх професійну кваліфікацію і досвід роботи, повинен контролювати прогрес та успіхи членів команди.

Управління ІТ-проєктами складніше, ніж управління проєктами інших типів. Термін «ІТ-проєкт» можна описати у вигляді загального терміна «проєкт» та з деякими важливими доповненнями, пов'язаними, перш за все, з тим, що ІТ-проєкт має на увазі собою діяльність, спрямовану на створення або використання інформаційних технологій. По-перше, дисонанс між замовником і виконавцем [7]. Найчастіше ІТ-проєкти замовляються бізнес-відділом організації, а впроваджуються - власним ІТ відділом або самостійної ІТ-компанією. При такому підході очевидно, що можливі труднощі у визначенні бізнес-вимог і загальних очікувань від розроблюваного і впроваджуваного

програмного забезпечення, так як область ІТ є досить специфічною. Це означає, що обговорення, які проводяться на перших етапах між замовником і виконавцем, можуть нести в собі досить розмиті ідеї, які на ранніх стадіях фіксуються наспіх в технічному завданні, а потім змінюються. При первинному читанні технічного завдання у замовника може виникнути ряд питань, і це нормально, тому що розробник технічного завдання міг зрозуміти якісь аспекти ІТ-проєкту дуже суб'єктивно. Іншою особливістю ІТ-проєкту є рівномірний розподіл провини на учасниках проєкту при його неблагополучної реалізації. Таким чином, відповідальність за результат лягає не тільки на виконавця (за можливу неправильну реалізацію) або не тільки на замовника (за можливу помилку в формуванні вимог), а на всіх учасників разом, тому що, так чи інакше, вони повинні створювати між собою ефективну комунікацію, мінімізуючи можливий суб'єктивізм в подальшому. Третьою особливостями ІТ-проєкту можна назвати його високу вартість. Якщо при побудові будівлі можливі лише мінімальні відхилення від кінцевих вимог і очікувань, то в ІТ-проєктах це досить імовірно. Відповідно, є ризик внесення істотних змін, а так як дуже часто відносини між замовником і виконавцем регулюються згідно з раніше підписаним контрактом, то будь-яка зміна тягне за собою так званий «запит на зміну», який коштує певну суму грошей. Велике число змін має на увазі під собою і велику вартість, саме тому ІТ-проєкти вважаються одними з найдорожчих видів проєктів [5].

Раніше життєвий цикл ІТ-проєктів розглядався з точки зору традиційної Водоспадної моделі (тобто у вигляді моделі, де кожний наступний етап процесу починається тільки тоді, коли завершиться попередній), застосування якої базується, скоріше, на звичці менеджерів проєктів, ніж на реальній ефективності моделі [3]. А останнім часом великої популярності набувають «гнучкі» методології управління ІТ-проєктами, основна суть яких базується на перевагах ітеративної та спіральної моделей життєвого циклу. Крім цього, «гнучкі» методології пропагуються як «готові до змін», що підходить для ІТ-проєктів, керованих в швидко мінливих середовищах. Scrum і Kanban -

представники методологій Agile-сімейства. Обидві вважаються гнучкими і ітеративними.

Основні принципи Scrum [5]:

- за встановлені невеликі відрізки часу (спринти) замовник отримує готовий продукт з можливостями, які мають максимальний пріоритет;
- в процесі роботи команда збирається для обговорень. На них здійснюється детальна перевірка виконаних завдань, встановлюються чергові цілі, виконується коригування всього процесу;
- проєкт можна допрацьовувати, попередньо визначаючи нові цілі, які перетворюються в завдання, і встановлюючи тривалість спринту.

Такий підхід дозволяє максимально контролювати процес розробки, за відносно невеликий термін отримувати намічений результат і при необхідності його коригувати.

Сильні сторони Scrum: швидкий запуск проєкту, чіткі терміни виконання завдань, складання планів і розбір підсумків, мінімізація бюджету за рахунок розстановки пріоритетів, працездатність продукту на виході, постійний контроль над ходом проєкту.

Слабкі сторони Scrum: зниження командного духу через недостатньо хорошої роботи когось із фахівців, імовірність виконання зайвих операцій, швидкий і жорсткий графік, велика кількість часу, що виділяється на обговорення на шкоду реальній роботі.

Основа підходу Kanban – зниження кількості виконуваних в даний момент задач.

Основні принципи Kanban:

- визначення етапів роботи. Їх зображають у вигляді стовпців на дошці. Завдання видаються картками, які переміщаються по етапах. Після проходження кожного з них ми отримуємо готовий до постачання замовнику елемент продукту або цілий продукт;
- відсутність поділу процесу виконання завдань за часом (немає спринтів). Тобто кожен учасник команди просто виконує завдання із загального

пулу, працює над нею з самого початку і до завершення. Процес вважається виконаним, завдання готова [8];

- кількість завдань зменшується за рахунок збільшення числа пунктів у кожній з них;
- обмеження на числа завдань на конкретному етапі. Ви самі можете змінювати кількість завдань. Це дозволить швидко виявити і вирішити будь-яке «затор» або недолік роботи;
- безперервний потік. Завдання потрапляють в чергу в порядку пріоритету, тому робота ніколи не припиняється.

Головне завдання Kanban – це зменшення часу проходження завдання від початку до стадії готовності.

Сильні сторони Kanban: підходить для згуртованих і мотивованих команд, немає фіксованих дедлайнів, ідеальний розрахунок навантаження між фахівцями, чітка розстановка обмежень і акцент на постійне поліпшення, економія трудових ресурсів, максимальна гнучкість [8].

Слабкі сторони Kanban: для максимального ефекту від роботи навички фахівців з команди повинні перетинатися (для спільного вирішення складних завдань), відсутність чітких термінів розхолоджує [8].

1.2 Аналіз управління командою розробки

1.2.1 Управління командою розробки

В наш час складно уявити керівника, який б не усвідомлював ролі і значення співробітників своєї організації у вирішенні поставлених задач. Не обладнання, технічна оснащеність, будівлі та споруди створюють найважливіші передумови діяльності організації, а його співробітники.

Методи управління персоналом в умовах посилення конкурентної боротьби стають головним чинником формування стійкої ефективної роботи організації та її конкурентоспроможності.

Ефективне керівництво персоналом вимагає знання в самих різних областях: менеджмент, психологія, стратегічне планування та інші. Важливо побудувати на підприємстві систему управління персоналом, яка допоможе вирішувати завдання фірми.

Управління трудовими ресурсами (HR – менеджмент, від англ. human resources – людські ресурси) є одним з найважливіших напрямків в діяльності багатьох організацій і вважається основним критерієм досягнення ними економічного успіху. Якщо раніше головну увагу було приділено розвитку та вдосконаленню технічного прогресу, впровадження прогресивних технологій, модифікації управлінських структур, то зараз видно значний поворот в сторону людського фактора, іншими словами, сутність і ефект бізнесу визначають люди.

Реалізація як стратегічних завдань і цілей, так і короткострокових планів будь-якої компанії передбачає виконання її персоналом певних дій, сукупність яких можна назвати виробничим поведінкою. Основний сенс і мету управління персоналом підприємства (організації) полягає в забезпеченні необхідного для досягнення організаційних цілей поведінки кожного співробітника. Ефективність використання людських ресурсів в організації багато в чому залежить від діяльності HR-менеджера або навіть групи співробітників служби кадрів підприємства [6].

1.2.2 Аналіз мотивації розробників

Мотивація є однією з провідних функцій управління, оскільки досягнення основної мети залежить від злагодженості в роботі людей [9]. Кожен кваліфікований керівник прагне підштовхнути співробітників працювати краще, створити для них зовнішні і внутрішні стимули для активної і творчої діяльності, підтримувати інтерес до роботи, показувати задоволення від отриманих результатів [9]. Цінно також, щоб працівники прагнули до досягнення мети організації добровільно і творчо. Система мотивації і

стимулювання персоналу замикає на собі багато інших складових цієї стратегії. Приклад, ефективно побудована система кадрового моніторингу, так само і навчання і розвитку персоналу, сприяє вирішенню завдання його мотивації [9].

Мотивація є динамічним процесом, що ініціює і підтримує певному рівні поведінкову, активність індивіда [10]. Вона включає систему стимулюючих факторів (потреби, мотиви, цілі, наміри і т.д.), здатних впливати на поведінку людей, сукупність причин психологічних чинників, які стимулюють, підтримують і пояснюють поведінку і активність людини [10].

Філософія управління персоналом – це формування поведінки окремих працівників по відношенню до цілей розвитку проєкту [11]. В таких умовах мотивація трудової діяльності співробітників набуває особливо важливого значення. Для того, щоб людина виконував доручену йому роботу сумлінно і якісно, він повинен бути в цьому зацікавлений або, інакше кажучи, мотивований [11]. В управлінні персоналом мотивація розглядається як процес активізації мотивів працівників (внутрішня мотивація) і створення стимулів (зовнішня мотивація) для їх спонукання до ефективної праці [11].

В основі сучасних теоретичних підходів мотивація визначається як рушійна сила людської поведінки, в основі якої знаходиться взаємозв'язок потреб, мотивів і цілей людини [11].

У теперішній час організація ефективної системи стимулювання персоналу є однією з найбільш складних практичних проблем менеджменту [11]. Типовими проблемами в організаціях, пов'язаними з низькою мотивацією персоналу є: висока плинність кадрів, висока конфліктність, низький рівень виконавської дисципліни, неякісна праця, нераціональність мотивів поведінки виконавців, слабкий зв'язок результатів праці виконавців і заохочення [11]. Також недбале відносини до праці, умови для самореалізації співробітників, низька ефективність впливу керівників на підлеглих, малорослий рівень міжособистісних комунікацій, збої у виробничому процесі, проблеми при заснуванні злагодженої команди, слабка перспектива кар'єрного росту, що відбивається на робочому тонусі співробітників, суперечність у відносинах між

підприємцем і працівником, низька ефективність методів нормативного опису праці, незадоволеність роботою співробітників, малий фаховий рівень персоналу, безініціативність співробітників, діяльність керівництва негативно оцінюється персоналом [11]. Досить важливо, незадовільний морально психологічний клімат, недостатнє оснащення робочих місць, організаційна плутанина, недостатня увага до навчання та стажування резерву, небажання співробітників підвищувати свою кваліфікацію, неналагодженість системи стимулювання праці, невідповідність між реальною поведінкою виконавця і очікуваннями від нього начальником, низький моральний дух в колективі, проблеми в управлінні персоналом, кар'єрного росту і багато інших [11].

Виділення потреб, які значимі для співробітників, і організація заходів щодо їх мотивації – необхідна зовнішня умова активізації та підтримка механізму самомотивації [11].

Для проведення стимулювання, доведеться гарно розумітися в мотивації кожного співробітника, так як, ефективність застосування стимулів безпосередньо залежить від ступеня індивідуалізації підходу. Підстави, вимоги і цінності індивідуальні. Вони не бувають повністю ідентичними для окремої соціальної групи або для всіх співробітників організації, тому важливо зуміти визначати і застосовувати особистісні підстави і вимоги співробітника [11].

При дослідженні мотивації необхідно враховувати вік і стать співробітників. Можна з упевненістю прогнозувати, що для різних вікових груп переважаючими будуть різні мотиватори. Так у віковому періоді від 22 до 36 років молоді люди мотивуються умовами роботи, які сприятимуть його професійному самоствердженню і зростання [12]. Співробітник даної вікової групи буде зацікавлений у підвищенні кваліфікації та подальшому професійному навчанні. Молоді люди добре розуміють, що чим більше він знає і вміє, тим більше затребуваний на ринку праці і тим він більше коштує [12].

Крім того, для нього серйозним мотиватором виступить робота в інноваційній команді, яка розробляє і впроваджує в компанії новий проєкт. Креативна змістовна діяльність захопить його і підвищить продуктивність [12].

Нарешті, молоді працівники даної вікової групи зацікавлені в заробітку, оскільки саме в цей період свого життя вони одружуються або виходять заміж і потребують досягненні матеріальної незалежності від батьків [8].

Таким чином, для даної вікової групи основними мотиваторами можуть бути: побудова вертикальної або горизонтальної кар'єри, підвищення професійної компетентності, заробітня плата, інтерес до інновацій, нових методів роботи, робота в команді з розробки та впровадження нового, амбіційного проєкту [11].

Персонал у віці від 36 до 60 років зацікавлений в отриманні посадового статусу, відповідного його кваліфікації, а також позитивне ставлення до нього з боку керівництва компанії. Він безумовно оцінить серйозний соціальний пакет, пропонується компанією, особливо, якщо в нього будуть входити заходи з підтримки здоров'я (оплата обідів, медична страховка, оплата занять спортом і фітнесом) [12].

Його приверне стабільна компанія з хорошою атмосферою і відсутністю важких конфліктів. У певний віковий період його обрадує перспектива передачі накопиченого професійного досвіду.

Таким чином, мотиватори для цієї вікової групи такі: службовий статус, позитивне відношення з боку керівництва, соціальний пакет, стабільна робота.

Звісно, система мотивації не є чимось застиглим, раз і назавжди створеним, вона повинна бути динамічною, здатною адаптуватися до мінливих зовнішніх умов, вимогам організації і людей [10]-[12].

1.2.3 Методи оцінки команди розробки

Оцінювання персоналу являє собою технологічний напрям кадрового менеджменту, яке вимагає певного професійного рівня. Оцінки співробітників виставляються виходячи з корпоративної культури, а також вимог до персоналу. Системи корпоративних політик і кадрових процедур на підприємстві, що використовуються для обчислення відповідності працівників

вимогам ринку, називають оцінкою персоналу. Це порівняння реального співробітника з його ідеальною моделлю, яка існує в голові керівника або створеної менеджером з персоналу. У системну оцінку колективу включені: внутрішні документи, що містять завдання і методи оцінки персоналу, процедуру оцінювання, обробку отриманої інформації, застосування результатів; проведення менеджерами з персоналу або керівниками процедур оцінювання ефективності праці персоналу, а також їх інтерпретація; збори результативних методів оцінки працівників компанії [7].

Головне завдання управління персоналом, яке вирішується за допомогою проведення оцінювання, - це спосіб отримати відомості про психологічні характеристики, результативності праці персоналу [1]-[2]. Результати оцінки допомагають начальнику або кадровику прийняти аргументоване рішення щодо підвищення (зниження) або переміщення окремих співробітників, підвищення їх кваліфікації або відкриття вакансії. Обумовлено проведення оцінювання співробітників організації збільшенням доходів компанії, підвищенням числа задоволених клієнтів, просуванням компанії на спільному ринку.

Оцінювання дозволяє відкоригувати і підвищити мотивацію персоналу. Ефективність оцінки персоналу залежить від її відповідності корпоративної етики, наявності зворотного зв'язку, спрямованості на підвищення кваліфікації кадрів. Оцінювання є способом вплинути на працівників. Отримавши відомості про свої сильні і слабкі сторони, співробітник прагне збільшити ефективність своєї праці. Та й «чудодійний стусан» іноді не завадить, і отримання оцінки своєї роботи виступає в ролі стимулу підвищити показники.

Сучасними методами проведення оцінки роботи є:

- інтерв'ю;
- тести;
- спостереження;
- зворотний зв'язок або оцінки 180 °-360 °;

Для оцінки діяльності членів команди менеджер може використовувати більш серйозні методи для отримання більш точного результату: параметричний та непараметричний підходи. Більш поширеним є використання методу DEA, де показники ефективності визначаються методами лінійного програмування. Отже, для оптимізації управління командою розробників можна використати метод DEA [9].

1.3 Підтримка прийняття рішень в управлінні командою розробки ІТ-стартапу

Ефективність – одне з найважливіших якостей, що характеризують системи будь-якої природи. Тому проблемі дослідження ефективності функціонування систем у всіх сферах діяльності людини в останні роки приділяється підвищена увага.

Для проведення таких досліджень використовуються різні підходи, в залежності від предметної області. Існує ряд методів, які використовуються для оцінки ефективності об'єктів в досить широких класах систем. Для оцінки діяльності членів команди менеджер використовує різні методи, наприклад [8]:

- параметричний підхід - апіорне завдання функціональної форми, яка описує кордон, що дозволяє застосовувати для її оцінювання економетричних методів (метод стохастичною кордону - stochastic frontier approach, SFA, метод товстої кордону - thick frontier approach, TFA і метод вільного розподілу - distribution-free approach, DFA);

- непараметричний підхід - показники ефективності визначаються методами лінійного програмування (різні варіації моделей методу огинаючих - data envelopment analysis, DEA).

Переваги непараметричного підходу:

- одночасний облік всіх витрат і випусків;
- відсутність апіорних обмежень на функціональну форму кордону.

Недоліки непараметричного підходу – необхідність великої кількості об'єктів для стійкості результатів.

Було обрано саме метод DEA [9], він підходить для оцінки членів команди стартапу, бо така команда складається лише із декількох людей, і саме непараметричний підхід легко використовувати для невеликої групи людей. Наприклад, метод SFA потребує більше часу, більше аналізу (а при керуванні невеликим стартапом це є зайвим), та результат буде таким, як і при використанні непараметричного підходу. Отже, раціональніше використовувати метод DEA.

Метод Data Envelopment Analysis (DEA) був розроблений вченим М. Фареллом та вдосконалений А. Чарнесом і В. Купером. В останні 25 років він широко застосовується для оцінки ефективності функціонування складних об'єктів в різних сферах. В якості української назви даного методу запропоновано варіант - «аналіз середовища функціонування (АСФ)» [9].

Основна ідея – це оцінка відносної ефективності функціонування об'єктів і побудова межі ефективності, яка має форму опуклої оболонки і використовується як еталон для проведення оцінки ефективності об'єктів у досліджуваній сукупності.

Традиційною основою DEA є метод лінійного програмування, що дозволяє знайти оптимальне серед безлічі допустимих рішень. При цьому вирішення завдання зведене до побудови так званої межі ефективності у багатовимірному просторі змінних, що характеризують аналізований об'єкт, та визначення відносної ефективності цих об'єктів.

Сучасні дослідження теоретично-методологічних та практичних засад методу DEA представлені у працях українських та зарубіжних вчених, зокрема: Гончарук А.Г., Чмутова І.М., Антонюк Я.М., Антонюк Г.Я., Григор'єв Г.С. та інші. Більшість науковців досліджують методологічні та методичні підходи до аналізу й оцінки ефективності діяльності організації, бо імплементація методу DEA дозволяє більш ширше оцінювати різні види ефективності з урахуванням постійної та змінної величини масштабу, що зумовлює необхідність

грунтовнішого дослідження даного питання та формування відповідних рекомендацій.

Сутність методу DEA полягає у зіставленні фактичного показника продуктивності (продукція/ресурси) з максимально можливим виходом продукції за даної кількості ресурсів [8]. Згідно з методом DEA, ефективність – це відношення фактичної продуктивності працівника до його максимально можливої продуктивності, яка визначається межею виробничих можливостей, тобто максимальною кількістю роботи, що робітник в змозі виконати за наявної кількості ресурсів та незмінності всіх інших факторів.

Міра ефективності у методі DEA показує кількість відсотків, на яку керівництво об'єкту має збільшити або зменшити вхідні або вихідні показники з метою досягнення рівня ефективності, оскільки існують інші об'єкти, реальні або гіпотетичні, що функціонують оптимально [9].

Таким чином, DEA є не тільки методом оцінки, а також методом управління, що є особливо важливим стосовно дослідження економічної ефективності економічних систем. Також DEA можна використовувати як метод рейтингування, оскільки результатом розрахунків є підсумкове значення, що вимірюється у відсотках та дозволяє порівнювати аналізований об'єкт з іншими.

Математично ефективність кожного об'єкта представляється як відношення зваженої суми виходів (мотиваційні фактори) до зваженої суми вхідів (показники з найбільшим впливом на результат) [7]. Отже, можна використати метод DEA для оцінки ефективності кожного програміста, за допомогою формули (1.1):

$$\max_{v,u} \theta = \frac{\sum_{j=1}^s u_j y_j}{\sum_{i=1}^m v_i x_i}, \quad (1.1)$$

де θ – значення ефективності об'єкта;

x_i, y_j – відповідно вхід та вихід для об'єкта;

v_i, u_j – це ваги, які необхідно розрахувати таким чином, щоб отримати максимальну оцінку дробу;

s, m – кількість входу та виходу відповідно.

Обмеження:

$$\sum_{i=1}^m v_i x_i = 1,$$

$$\sum_{i=1}^m v_i x_i - \sum_{j=1}^s u_j y_j \leq 0,$$

$$\frac{\sum_{j=1}^s u_j y_j}{\sum_{i=1}^m v_i x_i} \leq 1,$$

$$v_1, v_2, \dots, v_m \geq 0,$$

$$u_1, u_2, \dots, u_s \geq 0.$$

Для оцінки ефективності потрібно використовувати параметри, які відповідають рівню навичок фахівця, його комунікацію та інші показники. Використовуючи вище розглянутий метод, можна з легкістю зрозуміти, чи є відповідний співробітник ефективним.

Розрахунок ефективності співробітників – це один з основних чинників успішності стартапу. Оцінка повинна проводитись регулярно, для постійного відстеження результатів які надають інформацію про професійний розвиток кожного працівника. Також регулярна оцінка ефективності надає співробітнику інформацію про його рівень професійного розвитку, сильні та слабкі сторони та які в нього кар'єрні перспективи.

1.4 Висновки до розділу

В даному розділі описано основні поняття предметної області. Було розглянуто особливості управління розробкою ІТ-проектів, управління командою розробки, описано методи мотивації розробників та методи оцінки команди розробки. Виявлено, що оцінка роботи співробітників має важливу роль в управлінні командою розробки ІТ-стартапу. Для оптимізації управління командою обрано метод DEA, який детально описано в другому розділі.

2 МАТЕРІАЛИ І МЕТОДИ

Для реалізації проєкту використано такі технології:

- мова програмування: JavaScript;
- каскадна таблиця стилів: CSS;
- JavaScript фреймворк для створення інтерфейсу користувача Vue.js;
- програмна платформа: Node.js;
- редактор коду: Visual Studio Code.

2.1 Мова програмування JavaScript

JavaScript - це динамічна мова комп'ютерного програмування. Вона легка і переважно використовується як компонент вебсторінок, реалізації яких дозволяють клієнтському сценарію взаємодіяти з користувачем та створювати динамічні сторінки. Це інтерпретована мова програмування з об'єктно-орієнтованими можливостями [18].

JavaScript спочатку була відома як LiveScript, але Netscape змінив свою назву на JavaScript, можливо, через хвилювання, породжене Java. JavaScript вперше з'явилася в Netscape 2.0 у 1995 році під назвою LiveScript. Ядро мови загального призначення було вбудовано в Netscape, Internet Explorer та інші веббраузери [18].

Специфікація ECMA-262 визначила стандартну версію основної мови JavaScript [18]:

- JavaScript - це легка, інтерпретована мова програмування;
- призначена для створення мережево орієнтованих додатків;
- доповнює та інтегрує Java;
- доповнює та інтегрує HTML;
- відкритий і крос-платформний.

Переваги JavaScript [18]:

- швидкість. Клієнтський JavaScript дуже швидкий, оскільки його можна негайно запустити в клієнтському браузері. Якщо не потрібні зовнішні ресурси, JavaScript не перешкоджає мережевим викликам до серверного сервера [18];

- простота. JavaScript порівняно простий у вивченні та реалізації;
- популярність. JavaScript використовується повсюдно в Інтернеті;
- сумісність. JavaScript чудово грається з іншими мовами і може бути використаний у величезній кількості різноманітних додатків;
- навантаження сервера. Будучи на стороні клієнта, зменшується попит на сервері вебсайту;
- дає можливість створювати розширені інтерфейси.

Недоліки JavaScript [18]:

- клієнтська безпека. Оскільки код виконується на комп'ютері користувачів, в деяких випадках його можна використовувати для зловмисних цілей. Це одна з причин, чому деякі люди вирішили відключити Javascript;
- підтримка браузера. JavaScript іноді інтерпретується різними браузерами. Це дещо ускладнює написання коду між браузерами [18].

2.2 Каскадна таблиця стилів CSS

Каскадні таблиці стилів (CSS) – це проста мова дизайну, призначена для спрощення процесу зробити вебсторінки презентабельними [19]. CSS обробляє зовнішній вигляд частини вебсторінки. За допомогою CSS ви можете керувати стилем шрифтів, кольором тексту, розміром та розміщенням стовпців, інтервалом між абзацами, які фонові зображення або кольори використовуються, дизайном макета, варіаціями відображення для різних пристроїв та розмірами екрану а також безліч інших ефектів [19].

CSS легко вивчити і зрозуміти, але він забезпечує потужний контроль над поданням HTML-документа. Переважно CSS поєднується з мовами розмітки HTML або XHTML [19].

Переваги CSS [19]:

- послідовність. Головною перевагою CSS є те, що стиль послідовно застосовується на багатьох вебсторінках. Один командний рядок може одночасно керувати декількома областями, що є досить вигідним, якщо потрібно внести зміни [19]. Вам потрібно змінити лише одне, а решта піде далі. Оскільки вам не потрібно міняти кожну сторінку по одному, вебдизайнери можуть бути дуже ефективними у створенні та зміні вебсайту лише за допомогою декількох рядків коду [19];

- покращена швидкість вебсайту. Вебдизайнерам потрібно використовувати лише невелику кількість рядків програмування для кожної сторінки. А якщо коду менше, читати менше рядків, що призводить до швидшого завантаження кожної сторінки [19]. Оскільки користувачі Інтернету не бажають чекати, поки вебсайт завантажиться, покращення швидкості сайту буде найвигіднішим. Власники, які роблять ставку на ефективність вебсайтів для покращення рейтингу пошукових систем та клієнтської бази, отримають вигоду від CSS [19];

- легкий в обслуговуванні. Каскадна таблиця стилів не тільки спрощує розробку вебсайтів, але й обслуговування. Всі коди розміщені на одній сторінці, а це означає, що вдосконалення або зміна кількох рядків не потребуватиме перегляду кількох сторінок. І оскільки на вебсайті застосовується зміна в одному рядку коду, час і зусилля на технічне обслуговування значно скорочуються [19].

Недоліки CSS:

- вхід на різні рівні. Існує CSS, CSS 1 до CSS3, що спричинило плутанину серед розробників та веббраузерів. Одного типу CSS має бути достатньо. Було б кращим, ніж вибирати, який рівень CSS використовувати [19];

- фрагментація. У CSS те, що працює з одним браузером, не завжди може працювати з іншим. Ось чому веброзробники повинні перевірити на сумісність, запустивши програму в декількох браузерах, перш ніж вебсайт буде запущено в реальному часі. Якщо тільки люди використовують Mozilla або Chrome, але вони цього не роблять [19];

- відсутність безпеки. Оскільки це відкрита текстова система, CSS не має вбудованого захисту, який захищатиме її від перевизначення. Будь-хто, хто має доступ для читання / запису на вебсайті, може змінити файл CSS, змінити посилання або порушити форматування, випадково чи за дизайном [19].

2.3 JavaScript-фрейворк Vue.js

Vue.js – це JavaScript-фреймворк що використовує шаблон Model-View-ViewModel (MVVM) для створення інтерфейсів користувача на основі моделей даних [20], через реактивне зв'язування даних.

MVVM – це шаблон проєктування, що застосовується під час проєктування архітектури застосунків (додатків). Публічно вперше був представлений Джоном Госсманом (John Gossman) у 2005 році як модифікація шаблону Presentation Model. MVVM орієнтований на такі сучасні платформи розробки, як Windows Presentation Foundation та Silverlight від компанії Microsoft [20].

MVVM полегшує відокремлення розробки графічного інтерфейсу від розробки бізнес логіки (бек-енд логіки), відомої як модель (можна також сказати, що це відокремлення представлення від моделі). Модель представлення є частиною, яка відповідає за перетворення даних для їх подальшої підтримки і використання. З цієї точки зору, модель представлення більше схожа на модель, ніж на представлення і оброблює більшість, якщо не всю, логіку відображення даних. Модель представлення може реалізовувати патерн медіатор, організовуючи доступ до бек-енд логіки навколо множини правил використання, які підтримуються представленням [20].

Vue використовує синтаксис шаблонів на основі HTML, що дозволяє декларативно зв'язувати рендеринг DOM з основними екземплярами даних в Vue. Всі Vue шаблони валідні HTML, і можуть бути розпарсені браузерами та HTML парсерами. Всередині Vue компілює шаблони в рендерингові функції віртуального DOM. В поєднанні з реактивною системою, Vue здатний розумно обчислити кількість компонентів для ре-рендингу та застосувати мінімальну кількість маніпуляцій з DOM, коли стан застосунку зміниться [20].

В Vue можна використовувати синтаксис шаблонів або напямуч писати рендерингові функції використовуючи JSX. Для того, щоб це зробити просто замініть шаблон на рендерингову функцію [9]. Рендерингова функція відкриває можливості для потужних патернів базованих на компонентах – для прикладу, нова транзитна система тепер повністю базована на компонентах, що використовує рендерингові функції всередині [20].

Vue сам по собі не включає роутингу, та є vue-router пакет, який вирішує це питання. Він підтримує зв'язування вкладених шляхів з вкладеними компонентами і пропонує деталізований контроль над переходами. Vue дозволяє створення додатків за допомогою компонентів. Якщо додати vue-router до цього, все що потрібно зробити це зв'язати ваші компоненти з роутами і дозвольте vue-router вирішувати де їх рендерити [20].

Vue надає різноманітні шляхи для застосування ефектів переходу, коли елемент додають, оновлюють або видаляють з DOM. Наприклад:

- автоматичне застосування класів для CSS переходів та анімацій;
- інтегрування сторонніх бібліотек для CSS анімацій, таких як Animate.css [20];
- використовувати JavaScript для прямих маніпуляцій з DOM під час переходів [20];
- інтегрування сторонніх JavaScript бібліотек анімацій, таких як Velocity.js [20].

Коли елемент, який загорнутий перехідний компонент видаляють чи вставляють, ось що відбувається [20]:

1. Vue автоматично перевірить чи має застосовують до цього елементу CSS анімації та переходи. Якщо так, CSS класи для переходів будуть додані/видалені у відповідний час [20].
2. Якщо перехідний компонент має JavaScript зачіпки, ці зачіпки будуть викликані у відповідний час [20].
3. Якщо ніяких CSS переходів/анімацій не було знайдено та ніяких JavaScript не було надано, DOM операції для додавання і/та видалення відбудеться одразу ж в наступному фреймі [20].

2.4 Програмна платформа Node.js

Node.js – це платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript [21]. Засновником платформи є Раян Дал. Якщо раніше Javascript застосовувався для обробки даних в браузері користувача, то node.js надав можливість виконувати JavaScript-скрипти на сервері та відправляти користувачеві результат їхнього виконання. Платформа Node.js перетворила JavaScript на мову загального використання з великою спільнотою розробників [21].

Node.js має наступні властивості [21]:

- асинхронна одно-нитева модель виконання запитів;
- неблокуючий ввід/вивід;
- система модулів CommonJS;
- рушій JavaScript Google V8.

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

Платформа Node.js призначена для виконання високопродуктивних мережевих застосунків, написаних мовою програмування JavaScript [21]. Платформа окрім роботи із серверними скриптами для веб-запитів, також використовується для створення клієнтських та серверних програм. В платформі використовується розроблений компанією Google рушій V8 [21].

Для забезпечення обробки великої кількості паралельних запитів у Node.js використовується асинхронна модель запуску коду, заснована на обробці подій в неблокуючому режимі та визначенні обробників зворотніх викликів (callback) [21]. Для мультиплексування з'єднань використовується бібліотека libuv, для створення пулу нитей (thread pool) задіяна бібліотека libeio, для виконання DNS-запитів у неблокуючому режимі інтегрований c-ares. Всі системні виклики, що спричиняють блокування, виконуються всередині пулу нитей і потім, як і обробники сигналів, передають результат своєї роботи назад через неіменовані канали (pipe) [21]. При написанні програм для Node.js необхідно враховувати специфіку подієво-орієнтованого програмування [21].

Для розширення функціональності застосунків на базі Node.js підготовлена велика колекція модулів. Наприклад, HTTP, SMTP, XMPP, DNS, FTP серверів і клієнтів, модулі для інтеграції з різними веб-фреймворками, обробники WebSocket і AJAX, конектори до СКБД (MySQL, PostgreSQL, SQLite, MongoDB), шаблонізатори, CSS-рушії, реалізації криптоалгоритмів і систем авторизації (наприклад, OAuth), XML-парсери [21].

2.5 База даних PostgreSQL

PostgreSQL – це об'єктно-реляційна система керування базами даних (СКБД) [22]. Є альтернативою як комерційним СКБД (Oracle Database, Microsoft SQL Server, IBM DB2 та інші), так і СКБД з відкритим кодом (MySQL, Firebird, SQLite) [22]. Порівняно з іншими проєктами з відкритим кодом, такими як Apache, FreeBSD або MySQL, PostgreSQL не контролюється якоюсь однією компанією, її розробка можлива завдяки співпраці багатьох людей та компаній, які хочуть використовувати цю СКБД та впроваджувати у неї найновіші досягнення. Сервер PostgreSQL написаний на мові С. Зазвичай розповсюджується у вигляді набору текстових файлів із сирцевим кодом [22].

Функції дозволяють виконувати деякий код безпосередньо сервером бази даних. Ці функції можуть бути написані на SQL, який має деякі примітивні

програмні оператори, такі як галуження та цикли. Але гнучкішою буде функція написана на одній із мов програмування, з якими PostgreSQL може працювати. У PostgreSQL є підтримка індексів та при необхідності можна створити нові типи індексів [22]. PostgreSQL підтримує одночасну модифікацію БД декількома користувачами за допомогою механізму Multiversion Concurrency Control (MVCC). Завдяки цьому виконуються вимоги ACID, і практично відпадає потреба в блокуванні зчитування [22].

PostgreSQL підтримує великий набір вбудованих типів даних: числові типи, цілі, грошовий тип, символьні типи довільної довжини, двійкові типи, типи «дата/час», булевий тип, перерахування, геометричні примітиви, мережеві типи, IP і IPv6-адреси, CIDR-формат, MAC-адреса, UUID-ідентифікатор, XML-дані, JSON-дані, масиви, OID-типи, псевдотипи [22]. Крім того, користувач може самостійно створювати нові необхідні йому типи та програмувати для них механізми індексування за допомогою GiST [22].

Таблиці можуть успадковувати характеристики та набори полів від інших таблиць (батьківських). При цьому дані, які додаються до породженої таблиці, автоматично будуть брати участь (якщо це не вказано окремо) в запитах до батьківської таблиці. Цей функціонал в поточний час не є повністю завершеним. Однак він достатній для практичного використання [22].

Можливостями PostgreSQL також є: дотримання принципів ACID, Відповідність стандартам ANSI SQL-92 і SQL-99, Підтримка запитів з OUTER JOIN, UNION, UNION ALL, EXCEPT і підзапитів, послідовності, контроль цілісності, реплікація, загальні табличні вирази й рекурсивні запити, аналітичні функції, тригери, підтримка Unicode (UTF-8), підтримка регулярних виразів у стилі Perl, вбудована підтримка SSL і Kerberos, протокол поділюваних блокувань, завантажувані розширення, підтримують SHA1, MD5, XML і іншу функціональність, засоби для генерації сумісного з іншими системами SQL-коду та імпорту з інших систем [22].

2.6 Редактор коду Visual Studio Code

Visual Studio Code – це засіб для створення, редагування та зневадження сучасних вебзастосунків і програм для хмарних систем [23]. Visual Studio Code розповсюджується безкоштовно і доступний у версіях для платформ Windows, Linux і OS X [23]. Компанія Microsoft представила Visual Studio Code у квітні 2015 на конференції Build 2015 [23]. Це середовище розробки стало першим кросплатформовим продуктом у лінійці Visual Studio. Інтерфейс IDE середовища показано на рис. 2.1.

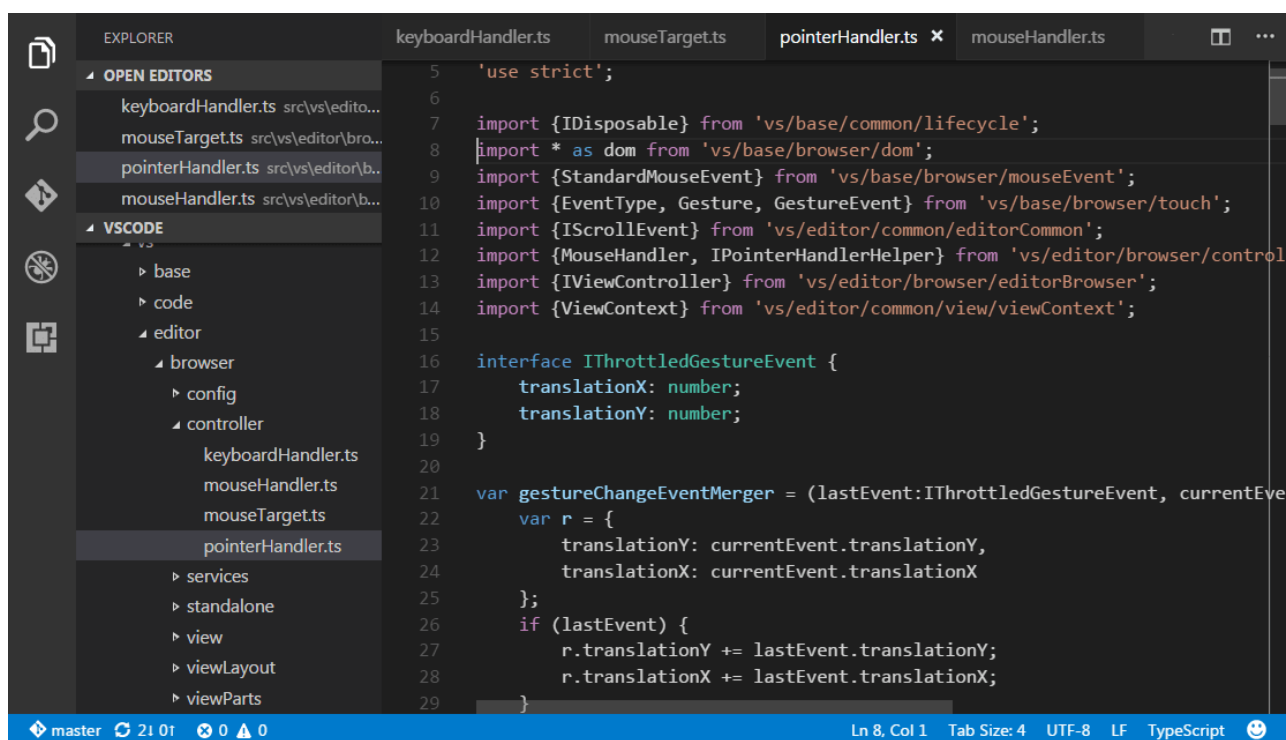


Рисунок 2.1 – Редактор IDE VS Code

За основу для Visual Studio Code використовуються напрацювання вільного проєкту Atom, що розвивається компанією GitHub [23]. Зокрема, Visual Studio Code є надбудовою над Atom Shell, що використовує браузерний рушій Chromium і Node.js [23]. Примітно, що про використання напрацювань вільного проєкту Atom і на сайті Visual Studio Code, і в пресрелізі, і в офіційному блозі не згадується [23].

Редактор містить вбудований зневаджувач, інструменти для роботи з Git і засоби рефакторингу, навігації по коду, автодоповнення типових конструкцій і контекстної підказки [23]. Продукт підтримує розробку для платформ ASP.NET і Node.js, і позиціюється як легковагове рішення, що дозволяє обійтися без повного інтегрованого середовища розробки [23]. Серед підтримуваних мов і технологій: JavaScript, C++, C#, TypeScript, jade, PHP, Python, XML, Batch, F#, DockerFile, Coffee Script, Java, HandleBars, R, Objective-C, PowerShell, Luna, Visual Basic, Markdown, JSON, HTML, CSS, LESS і SASS, Naxe [23].

2.7 Висновки до розділу

В якості мови програмування для проєкту було використано мову програмування JavaScript за її зручність та універсальність. В якості фреймворку для розроблення клієнтської частини застосунку було обрано Vue.js за його велику кількість можливостей та використання шаблону створення інтерфейсів користувача на основі моделей даних через реактивне зв'язування даних. Для розробки проєкту було обрана програмна платформа Vue.js та середовище розробки, редактор коду, Visual Studio Code.

3 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

3.1 Загальний опис та архітектура програмного додатку

В ході аналізу цілей та задач розробки було виділено дві основні групи функціональних можливостей додатку: задачі з управління ресурсами стартап проєкту на кожному етапі розробки програмного продукту, задачі просування та планування рекламної кампанії самого проєкту на кожному етапі життєвого циклу стартапу. З огляду на це було вирішено реалізувати окремі програмні модулі для кожної з цих груп та зв'язати їх в один програмний комплекс за допомогою спільної головної сторінки інтерфейсу користувача та спільної бази даних [10]. Загальну архітектуру додатку представлено на рис. 3.1.

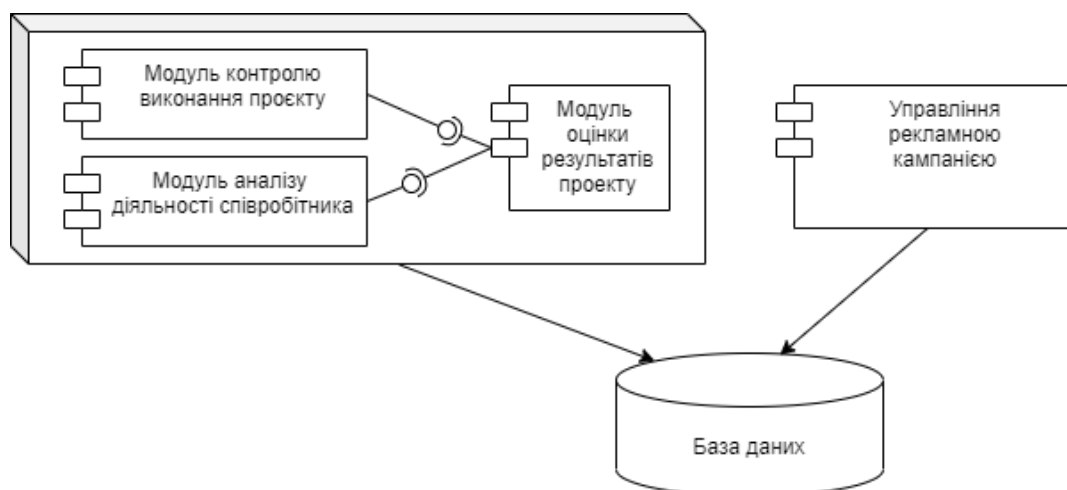


Рисунок 3.1 – Загальна архітектура додатку

Було прийнято рішення про програмну реалізацію на мові JavaScript з використанням сучасних інструментів розробки. Основними причинами вибору саме цієї мови є його хороша «підтримуваність», тобто можливість спільного програмування, супроводження та удосконалення програми декількома програмістами, розділеними в просторі і часі; масштабованість проєктів; велика кількість вже готових шаблонів та бібліотек стандартних функцій обробки даних.

Додаток створено з використанням компонентів програмної платформи Node.js та інструментів роботи з реляційною базою даних PostgreSQL, фреймворку Vue.js, середовища розробки Visual Studio Code, а також набору скриптів HighCharts.

Вікна додатку складаються з пов'язаних між собою web-сторінок, що містять кнопки, посилання, графіки, списки і таблиці. Додаток повністю відповідає завданню роботи, зручний у використанні та має інтуїтивно зрозумілий інтерфейс.

При створенні програмного продукту одним з головних кроків є детальне вивчення процесу управління персоналом. Саме від цього залежить перелік функцій які має виконувати Web-додаток. При роботі з персоналом менеджери з управління мають справу з великими об'ємами інформації, тому є необхідність у створенні єдиного сховища даних. В ньому має зберігатися особисті дані працівника, документація, та інформація про його професійний розвиток. Вся інформація має оброблятися та бути представлена в табличному та графічному виглядах.

В результаті обробки певних вхідних даних формується інформація, що дозволяє спростити прийняття рішень про подальшу кар'єру того чи іншого працівника. Для збереження усіх даних, необхідних для роботи, було проаналізовано предметну область та розроблено архітектуру бази даних (рис. 3.2).

Вхідними даними для роботи з Web-додатком є: особиста інформація про працівника, резюме, текстовий файл з питаннями до анкет.

Вихідними даними роботи з додатком можна вважати: статистичні дані про працівників на всьому підприємстві, результати анкетування в рамках процесу атестації, рішення про класифікацію за професійним рівнем, особисті професійні рекомендації, статистика даних проєкту за задачами, членами команди та місяцями, рекомендації про потенційні ризики на проєкті.

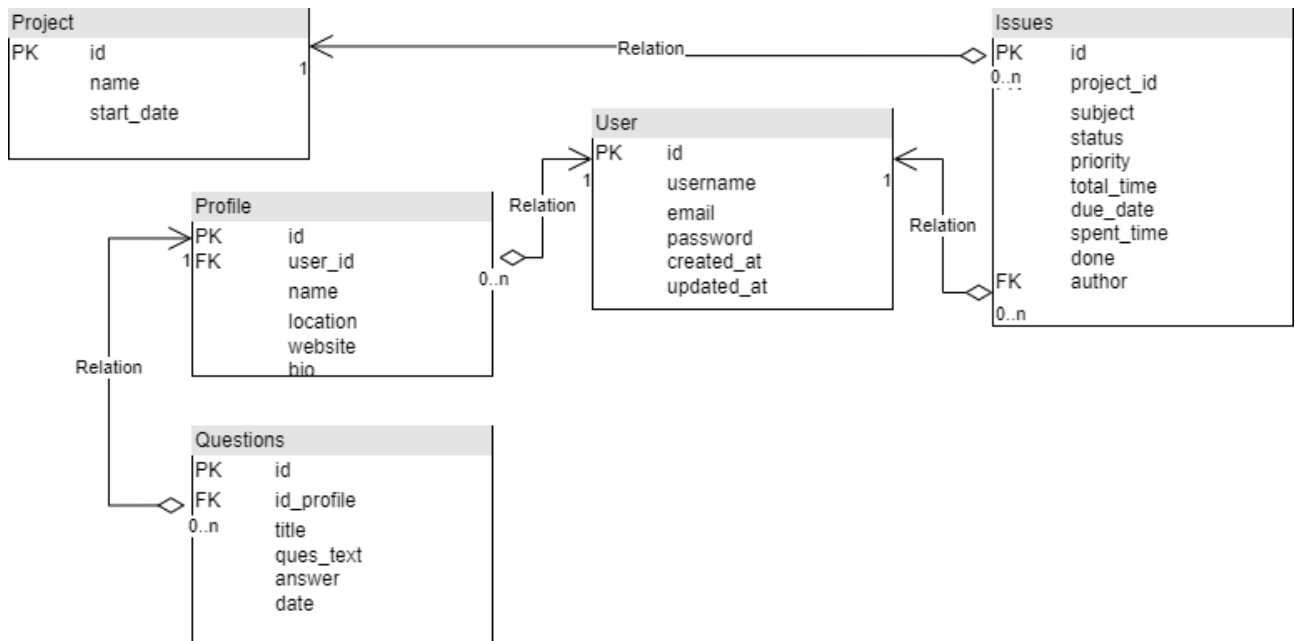


Рисунок 3.2 – Діаграма бази даних

Враховуючи сучасні тенденції у розвитку програмного забезпечення, запропоновані методи було програмно реалізовано у вигляді Web-додатку, що надає системі переваги у мобільності (користувач має можливість працювати у системі на будь-якому комп'ютері, що має доступ до мережі Інтернет), безпеці, надійності та низьких вимог до ресурсів (адже всі дані зберігаються на сервері замість локального встановлення на комп'ютер оператора).

3.2 Функціональні елементи додатку

Програмний додаток повинен легко перебудовуватися відповідно до змін у бізнес-процесах окремого стартапу та корпоративному технічному процесі роботи над стартапом, а також змін, що відбуваються на ринку праці та продуктах конкурентів [10]. Таким чином, в ньому повинна бути передбачена можливість внесення в неї зазначених змін. Архітектура додатку з управління ресурсами проєкту наведено на рис. 3.3. Архітектура логіки програмного додатку складається з окремих модулів, кожен з яких відповідає за свій набір функцій. Усі модулі виконують запити до бази даних, у якій зберігаються усі

дані системи, та пов'язані між собою графічним інтерфейсом, з яким за допомогою браузера взаємодіє оператор системи.



Рисунок 3.3 – Діаграма архітектури програмного додатку

3.3 Опис функціональності модулів додатку

Модуль авторизації є стандартним компонентом майже для всіх інформаційних систем, ресурсів та програмних продуктів, база даних яких містить будь-яку особисту інформацію [10]. Додаток містить усі особисті дані та конфіденційну інформацію, тому доступ до використання системи обов'язково повинен бути авторизованим. Для входу у систему користувач повинен ввести особистий логін та пароль.

Модуль контролю виконання проєкту об'єднує у собі функції з управління проєктом, а саме: збір та збереження даних щодо прогресу виконання задач проєкту, імпорт даних про задачі проєкту, редагування даних за кожною задачею, виведення даних у вигляді зручної для перегляду звітної таблиці.

Модуль аналізу діяльності співробітника виконувати основні заходи щодо управління персоналом (трудовими ресурсами) стартапу та об'єднує у собі наступні функції: збір, систематизація та збереження особистих, юридичних та професійних даних кожного працівника, відображення списку всіх працівників підприємства, можливість додавати нових та архівувати працівників, можливість редагування профілю працівника та всі дані про його вміння та навички, відображення статистики на всьому підприємстві, відображення інформації, що дозволить приймати об'єктивні та обґрунтовані рішення про подальшу долю працівників підприємства, зручний інтерфейс слідкування за кар'єрним розвитком кожного співробітника, проведення анкетування та збору даних щодо особистої мотивації кожного члена команди.

Модуль оцінки результатів діяльності проєкту виводить статистичні звіти та діаграми використання ресурсів проєкту та роботи над проєктом. В автоматичному режимі формуються наступні стандартні графіки, що відповідають рекомендаціям з ведення проєкту за методологією Scrum:

- колова діаграма співвідношення кількості задач за типами, наприклад: функціонал (feature), дефекти (bug) та покращення (improvement);
- колова діаграма об'єму витраченого часу за типами задач;
- стовбчикова діаграма кількості завершених задач за кожен місяць.

Модуль управління рекламною компанією. При створенні програмного продукту одним з головних кроків є детальне вивчення відповідної послідовності виконання розрахунків при плануванні рекламної кампанії та складанні медіаплану розміщення рекламного повідомлення. Головним призначенням програми даного проєкту є скорочення рутинних операцій та підтримка прийняття рішень при виконанні ефективного планування рекламної кампанії на підприємстві. В результаті обробки певних вхідних даних програмою формується низка відповідних рекомендацій щодо проведення розрахунку усіх потрібних параметрів медіаплану. Це значить, що доцільним у даному випадку буде формування під час роботи вихідного текстового файлу, у якому буде зберігатись підсумковий звіт із результатами розрахунків на основі

певних вхідних даних. Враховуючи особливості поетапного виконання розрахунків медіаплану для кожного етапу створюється окрема форма, на якій розміщується текстові поля для введення необхідних на даному етапі параметрів та області з вихідними результатами.

3.4 Висновки до розділу

В даному розділі описано архітектуру програмного додатку, функціональні елементи та представлено функціональні можливості модулів. В ході аналізу цілей та задач розробки було виділено дві основні групи функціональних можливостей додатку: задачі з управління ресурсами стартап проєкту на кожному етапі розробки програмного продукту, задачі просування та планування рекламної кампанії самого проєкту на кожному етапі життєвого циклу стартапу. З огляду на це було вирішено реалізувати окремі програмні модулі для кожної з цих груп та зв'язати їх в один програмний комплекс за допомогою спільної головної сторінки інтерфейсу користувача та спільної бази даних.

4 РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ

4.1 Автоматизація управління командою розробки ІТ-стартапу

Основною функцією розробленої системи є: розрахунок ефективності співробітників на кожному етапі реалізації стартапу. Таким чином, було прийнято рішення реалізувати окрему програмну систему зі зручним інтерфейсом користувача. Система була розроблена на мові JavaScript, бо це інтерпретована високорівнева мова програмування, яка підтримує імперативний, функціональний і подієво-орієнтовані стилі. Іншою перевагою є велика кількість готових пакетів для різних потреб розробника (пакети для інтерфейсу, роботи з даними). Веб-застосунок було реалізовано на програмній платформі Node.js, з фреймворком для візуального відображення Vue.js, пакетом для контролю стану даних Vuex та різними допоміжними пакетами для реалізації ергономічного інтерфейсу користувача. Реалізована система була написана у середовищі розробки Visual Studio Code.

Система складається з двох сторінок: перша сторінка має таблицю для введення розрахункових даних, а друга – результат розрахунку з графіком, який відображають роботу метода DEA. Одним з головних кроків при реалізації веб-застосунку є детальний аналіз ефективності співробітників у стартап проєкті. Саме від цього залежить вибір стратегії управління стартапом.

4.2 Розробка інтерфейсу додатку

Розроблена система являється веб-застосунком, яка була реалізована за допомогою схеми розділення даних Model-View-Controller (MVC). За модель в розробленій системі відповідає допоміжний пакет Vuex. В ньому реалізована основна бізнес логіка системи. Для відображення користувацького інтерфейсу було використано допоміжні пакети з фреймворку Vue.js. Самі сторінки системи складаються з незалежних компонентів, які можна легко

перевикористовувати і не турбуватися, що якась сторінка зламається. Також компоненти відповідають основним вимогам User Interface (UI) (рис. 4.1).

Data Envelopment Analysis

Outputs 3 → Inputs 2 →

#	1	2	3	1	2
1	56	73	45	58	74
2	74	89	91	96	64
3	61	78	65	75	85
4	76	83	51	54	73
5	79	82	78	68	64
6	65	84	78	80	58
7	66	81	74	86	69

FILE INPUT

CALCULATE

Рисунок 4.1 – Інтерфейс веб-додатку

Також, для поліпшення роботи з додатком та уникнення ручного введення, було реалізовано можливість додавання даних з допомогою файлу. Файл повинен містити дані для розрахунку ефективності членів стартап-проекту (рис. 4.2).

Create Profiles

Showing 1-16 of 16 items.

#	Name	Department	Level	Level	Last Attestation	
1	Olena Minchuk	Web	senior	senior	2015-12-06	👁️✎️🗑️
2	Olena Zhdanovka		middle	middle	0000-00-00	👁️✎️🗑️
3	Timur Kabeev		middle	middle	0000-00-00	👁️✎️🗑️
4	Oleg Gulkin		middle	middle	0000-00-00	👁️✎️🗑️
5	Marina Gulkina	Mobile	middle	middle	0000-00-00	👁️✎️🗑️
6	Viktoria Chomkina	Development	junior	junior	0000-00-00	👁️✎️🗑️
7	Anna Hemur	QA	junior	junior	0000-00-00	👁️✎️🗑️
8	Andriy Mishkin	Development	junior	junior	0000-00-00	👁️✎️🗑️
9	Anton Scrupnikov	UI/UX Design	junior	junior	0000-00-00	👁️✎️🗑️

Рисунок 4.2 – Завантаження файлу списку команди проекту

За контролер в системі відповідає компонент, реалізований з допомогою фреймворку Vue.js. Контролер являється сполучною частиною між модель та відображенням. Від реалізує валідацію даних та передачу їх в модель, а потім передає на відображення.

Також, для гнучкості у використанні, веб-застосунок було адаптовано для всіх типів екранів, що забезпечує можливість користуватися сайтом з будь-якого пристрою (рис. 4.3).

#	1	2	3
1	42	74	48
2	65	92	43
3	66	81	74

Рисунок 4.3 – Мобільна версія веб-застосунку

4.3 Розрахунок ефективності роботи співробітника

Для демонстрації роботи створеного додатку та методу DEA, в табл. 4.1 наведені дані, для розрахунку ефективності кожного окремого співробітника.

Таблиця 4.1 – Данні для підрахунку ефективності

№	Outputs			Inputs		
	мотивація	кар'єрні перспективи	атмосфера в колективі	заробітна плата	навантаження	бонуси
1	42	74	48	82	65	76
2	65	92	43	81	84	84
3	51	72	63	83	42	56
4	56	73	45	58	74	78
5	74	89	91	96	64	80
6	61	78	65	75	85	76
7	76	83	51	54	73	82
8	79	82	78	68	64	67
9	65	84	78	80	58	59
10	66	81	74	86	69	75

Для розрахунку ефективності було використано наступні дані: вихідні (outputs; мотивація, кар'єрні перспективи, атмосфера в колективі) та вхідні (inputs; заробітна плата, навантаження, бонуси). Після отримання вибірки, можна скористатися створеною системою для розрахунку ефективності кожного співробітника (рис. 4.4).

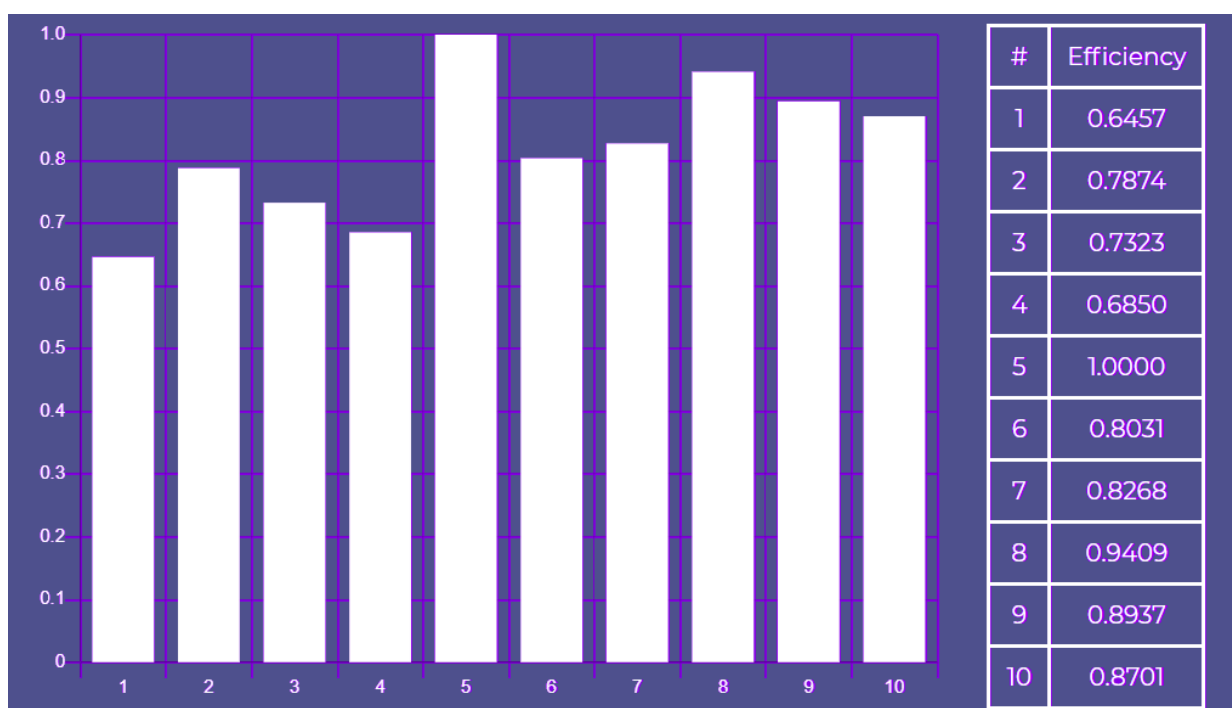


Рисунок 4.4 – Результат розрахунку ефективності

Як видно з рис.4.4 створена система розраховувала коефіцієнт ефективності для кожного співробітника і найефективнішим співробітником виявився співробітник від номер 5. Тому що він задоволений кар’єрним ростом, атмосферою в колективі, заробітною платою та бонусами, також має середні значення мотивації та навантаження. Таким чином, метод DEA наглядно демонструє, які чинники не задовольняють співробітників і тим самим дозволяє спланувати подальші етапи розвитку стартапу. На рис. 4.5 показано приклад статистичних діаграм проєкту.

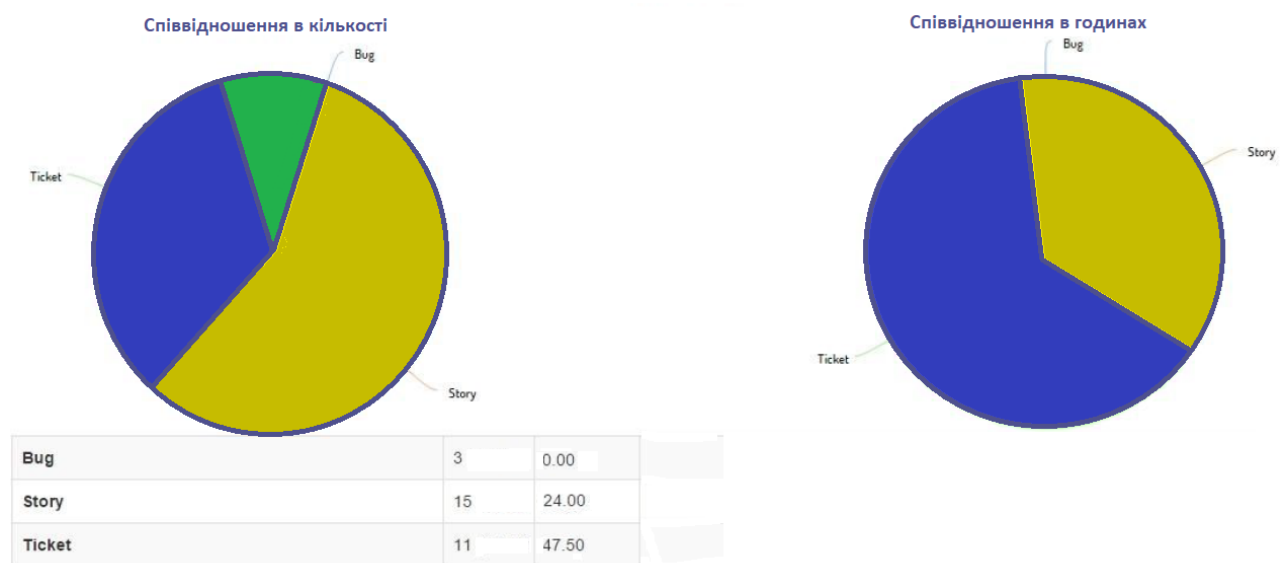


Рисунок 4.5 – Статистичні діаграми проєкту

4.4 Висновки до розділу

В розділі описано програмну реалізацію додатку. Основною функцією розробленої системи є: розрахунок ефективності співробітників на кожному етапі реалізації стартапу. Таким чином, було прийнято рішення реалізувати окрему програмну систему зі зручним інтерфейсом користувача. Система була розроблена на мові JavaScript, бо це інтерпретована високорівнева мова програмування, яка підтримує імперативний, функціональний і подієво-орієнтовані стилі. Іншою перевагою є велика кількість готових пакетів для

різних потреб розробника (пакети для інтерфейсу, роботи з даними). Веб-застосунок було реалізовано на програмній платформі Node.js, з фреймворком для візуального відображення Vue.js, пакетом для контролю стану даних Vuex та різними допоміжними пакетами для реалізації ергономічного інтерфейсу користувача. Реалізована система була написана у середовищі розробки Visual Studio Code.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Тестування інтерфейсу додатку

Після завершення розробки додатку потребується його тестування. Першим кроком було тестування Web UI. Перш за все треба перевірити зовнішність додатку на всіх браузерах, приділяючи достатньо уваги Edge та Safari. Для тестування Web UI я використовував Chrome dev tools. За допомогою цього інструменту, я перевіряв адаптивність для мобільних телефонів, планшетів, десктопу та дисплею з високим розширенням.

Web UI зображено на рис. 5.1.

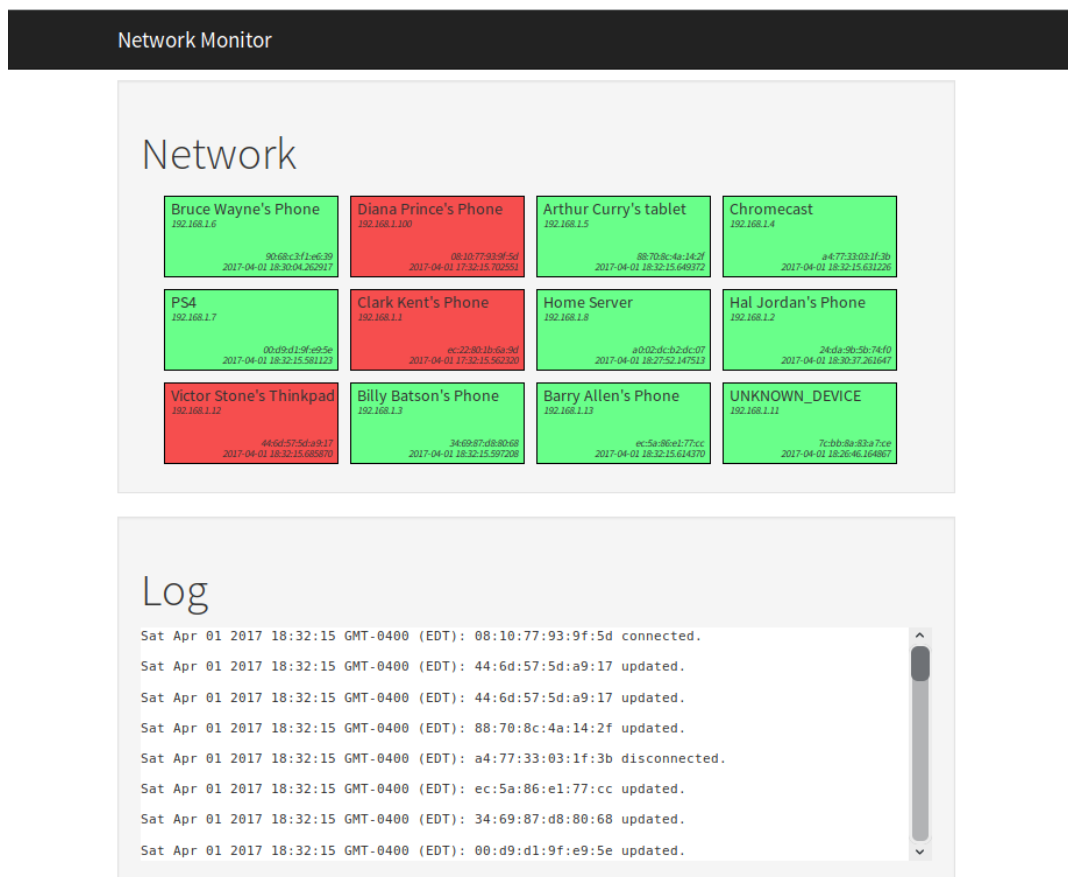


Рисунок 5.1 – Web UI

На наступному екрані відображається активні та неактивні IP адреси в локальній мережі. Крім того відображається найменування пристрою, їх MAC адреса й дата останньої активності.

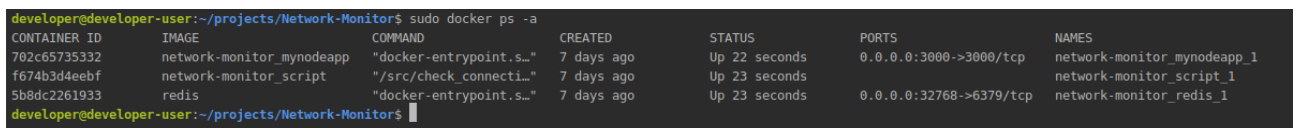
Після блоку активних та неактивних IP, відображається лог активності. Він зберігається в файлі log.txt. Для його тестування я виконав наступні кроки:

- запуск додатку;
- підключення нового пристрою до локальної мережі;
- відключення пристрою від локальної мережі.

Після цього в логіві з'явилися два нових записи. Перший – це був запис про знайдений новий пристрій та його підключення до локальної мережі. Другий – це запис про його відключення від локальної мережі. Після відключення пристрою від локальної мережі, колір його блоку змінився на червоний.

5.2 Тестування скрипту для пошуку активних і неактивних IP

Після того, як ми запустили додаток, автоматично запускається скрипт, який в автоматичному режимі сканує локальну мережу та знаходить активні й неактивні IP. Для тестування цього скрипту, я перевіряв, що докер контейнер створився коректно. Для цього я використав наступну команду – `sudo docker ps -a`. Ця команда відобразила інформацію, зображену на рисунку 5.2.



CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
702c65735332	network-monitor_mynodeapp	"docker-entrypoint.s..."	7 days ago	Up 22 seconds	0.0.0.0:3000->3000/tcp	network-monitor_mynodeapp_1
f674b3d4eebf	network-monitor_script	"/src/check_connecti..."	7 days ago	Up 23 seconds		network-monitor_script_1
5b8dc2261933	redis	"docker-entrypoint.s..."	7 days ago	Up 23 seconds	0.0.0.0:32768->6379/tcp	network-monitor_redis_1

Рисунок 5.2 – Функція для відображення активних докер контейнерів

На цьому скріншоті ми бачимо інформацію про те, що контейнер зі скриптом був запущений та працює.

Для перевірки того, що функції скрипту та сам скрипт знаходяться в докер контейнері, використано наступну команду – `sudo docker exec -it f6 bash`. За допомогою цієї команди, можна зайти в контейнер з скриптом, після цього – переконались в тому, що скрипт знаходиться в контейнері.

Для тестування коректної роботи скрипта, було підключення нового пристрою до локальної мережі після запуску додатку. Пристрій з'явився у вебдодатку та у файлах `src/resource/mac_addresses.json`.

5.3 Тестування сервера

Мій сервер відповідає за відображення вебдодатку та зв'язку з даними, які обробляються та записуються до необхідних файлів за допомогою скрипту. Для тестування сервера, мені потрібно впевнитись, що докер контейнер коректно почав роботу. Для цього я використав команду – `sudo docker ps -a` та впевнилась, що контейнер піднявся та працює. Після перевірки коректності роботи докер контейнеру, потрібно перевірити, що сервер працює на `localhost:3000`. Для цього я скористався веббраузером Google Chrome, ввів в пошукову строку `localhost:3000` і мені відобразився мій додаток. Це означає, що сервер коректно працює.

5.4 Симуляція DoS і DDoS атак

Для того, щоб симулювати DoS і DDoS атаки, я написав наступний скрипт. Цей скрипт симулює DoS і DDoS атаки з різних IP та різних портів. Після запуску скрипту, до чорного списку були додані IP, з яких була атака.

На рис. 5.3 зображена функція, яка потрібна для відображення активних докер контейнерів.

```

import random
from scapy.all import *
target_IP = input("Enter IP address of Target: ")
i = 1

while True:
    a = str(random.randint(1,254))
    b = str(random.randint(1,254))
    c = str(random.randint(1,254))
    d = str(random.randint(1,254))
    dot = "."
    Source_ip = a + dot + b + dot + c + dot + d

    for source_port in range(1, 65535)
        IP1 = IP(source_IP = source_IP, destination = target_IP)
        TCP1 = TCP(srcport = source_port, dstport = 80)
        pkt = IP1 / TCP1
        send(pkt,inter = .001)

    print ("packet sent ", i)
    i = i + 1

```

Рисунок 5.3 – Функція для відображення активних докер контейнерів

5.5 Висновок за розділом 5

Для тестування додатку мені знадобились наступні інструменти: Google Chrome dev tools та команди докер для перевірки коректної роботи контейнерів та можливості входу до контейнеру. Додаток пройшов перевірку на працезданість Web UI, скрипту для пошуку IP адресів та роботи серверу. В ході перевірки вебдодатку не було знайдено помилок, які впливають на роботу основних функцій додатку.

ВИСНОВКИ

В ході розробки створено додаток для дослідження методів мотивації і управління командою розробки IT-стартапу. В проєкті були використані наступні технології: мова програмування JavaScript, програмна платформа Node.js та JavaScript фреймворк для створення інтерфейсу користувача Vue.js, реляційна база даних PostgreSQL, персональний комп'ютер з процесором Intel Core i5 під управлінням операційної системи Microsoft Windows 10.

Основною функцією розробленого додатку є: розрахунок ефективності співробітників на кожному етапі реалізації стартапу. Таким чином, було прийнято рішення реалізувати окрему програмну систему зі зручним інтерфейсом користувача. В результаті дослідження проаналізовано особливості стартап-проєктів та керування командою розробників, розглянуто методи керування співробітників у стартап-проєкті, створено програмний веб-застосунок для автоматизації методів мотивації і управління командою розробки IT-стартапу. Галуззю використання є IT-компанії та установи, що мають у своєму складі відділи IT.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Що таке стартап приклади і пояснення, pammtoday [Електроний ресурс]. – Режим доступу: <https://jak.koshachek.com/articles/shho-take-startap-prikladi-i-pojasnennja-pammtoday.html>.
2. Що таке стартап-компанія? [Електроний ресурс]. – Режим доступу: <https://genomukr.ru/biznes/17668-shho-take-startap-kompanija.html>.
3. Плєскач, В. Л. Інформаційні системи і технології на підприємствах / В. Л. Плєскач. – К.: Знання, 2018. – 718 с.
4. Стартап [Електроний ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%80%D1%82%D0%B0%D0%BF>.
5. Кращі бізнес-стартапи в Україні [Електроний ресурс]. – Режим доступу: https://bankchart.com.ua/biznes/biznes_start/statti/kraschi_biznes_startapi_v_ukrayini.
6. Як розпочати стартап [Електроний ресурс]. – Режим доступу: https://courses.prometheus.org.ua/assets/courseware/ad0ad6b20704806b682ee9940ffc6bf/c4x/Prometheus/Startup101/asset/%D0%9B%D0%B5%D0%BA%D1%86%D1%96%D1%8F_1.pdf.
7. Управління проектом: основні аспекти та функції [Електроний ресурс]. – Режим доступу: <https://buklib.net/books/34104/>.
8. Сучасні методи управління проектами [Електроний ресурс]. – Режим доступу: <https://sgv.in.ua/off-lifaq/25-suchasni-metodi-upravlinnya-proektami>.
9. Балабанова, Л. В. Управління персоналом / Л. В. Балабанова, О. В. Сардак. – К.: Центр учбової літератури, 2015 – 468 с.
10. Мотивація діяльності особистості в організації [Електроний ресурс]. – Режим доступу: https://ua-referat.com/Мотивація_діяльності_особистості_в_організації.
11. Сутність теорій мотивації трудової діяльності [Електроний ресурс]. – Режим доступу: https://ua-referat.com/Сутність_теорій_мотивації_

трудової діяльності.

12. Попко, О. В. Сучасна парадигма стартапів у бізнесі / О. В. Попко, М. В. Мальчик // Вісник Національного університету «Львівська політехніка». Логістика. – 2016. – № 811. – С. 275-279.

13. Курченко, О. О. Становлення та розвиток стартапів в Україні: проблеми та шляхи вирішення / О. О. Курченко // Український соціум. – 2016. – № 2. – С. 80-87.

14. Корнух, О. В. Стартап як прогресивна форма інноваційного підприємництва / О. В. Корнух, Л. В. Маханько // Інвестиції: практика та досвід. – 2014. – № 23. – С. 26-30.

15. Understanding employee motivation and organizational performance [Electronic resource] – Access mode: <https://www.elsevier.es/en-revista-journal-innovation-knowledge-376-articulo-understanding-employee-motivation-organizational-performance-S2444569X16000068>.

16. Data Envelopment Analysis [Electronic resource] – Access mode: <https://core.ac.uk/download/pdf/82733681.pdf>.

17. Unified Modeling Language tutorial [Electronic resource] – Access mode: <https://www.tutorialspoint.com/uml/index.htm>.

18. Браун, Е. Изучаем JavaScript: руководство по созданию современных веб-сайтов / Е. Браун // ООО «Альфа унига». – 2017. – № 3. – 368 с.

19. Майер, Е. CSS-каскадные таблицы стилей: подробное руководство / Е. Майер // СПб: «Символ-Плюс». – 2008. – № 3. – 576 с.

20. Vue.js [Електроний ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/Vue.js>.

21. Node.js [Електроний ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/Node.js>.

22. PostgreSQL [Електроний ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/PostgreSQL>.

23. Visual Studio Code [Електроний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Visual_Studio_Code.

ДОДАТОК А
Текст програми

A.1 Скрипт main.js

```
import Vue from "vue";
import App from "./App.vue";
import router from "./router";
import store from "./store";
import vuetify from "./plugins/vuetify";
Vue.config.productionTip = false;
new Vue({ router, store, vuetify, render: (h) => h(App), }).$mount("#app");
```

A.2 Скрипт router.js

```
import Vue from "vue";
import Router from "vue-router";
Vue.use(Router);
export default new Router({
  mode: "history",
  routes: [
    {
      name: "Home", path: "/",
      component: () => import("@/pages/Home"),
    }, {
      name: "Results", path: "/results",
      component: () => import("@/pages/Results"),
    },
  ],
  scrollBehavior() {
    return { x: 0, y: 0 };
  },
});
```

A.3 Скрипт App.vue

```

<template>
  <router-view />
</template>

<script>
export default {
  name: "App",
};
</script>

<style lang="scss">
html { height: 100%; background-color: #8600e6; }
body { font-family: "Montserrat", sans-serif; }
.custom-container { padding: 0 15px; }
.logo { font-size: 45px; font-weight: bold; padding: 40px 0; }
.d-flex { display: flex; }
.text-white { color: white; }
.text-center { text-align: center; }
.d-none { display: none; }
.d-inline-flex { display: inline-flex; }
.d-inline-block { display: inline-block; }
.justify-center { justify-content: center; }
.align-center { align-items: center; }
.pbl-10px { padding: 0 0 10px 10px; }
.ptr-10px { padding: 10px 10px 0 0; }
.mt-10px { margin-top: 10px; }
.mr-10px { margin-right: 10px; }
.mr-30px { margin-right: 30px; }
.mx-auto { margin-left: auto; margin-right: auto; }
.overflow-x-auto { overflow-x: auto; }

```

```
.mb-10px { margin-bottom: 10px; }
.pb-20px { padding-bottom: 20px; }
.text-bold { font-weight: bold; }
.cursor-pointer { cursor: pointer; }
</style>
```

```
store.js import Vue from "vue"; import Vuex from "vuex";
```

A.4 Скрипт **Vue.use(Vuex);**

```
export default new Vuex.Store({
  state: {
    outputs: {
      matrix: [
        ["0", "0", "0"],
        ["0", "0", "0"],
        ["0", "0", "0"],
      ],
      countColumns: 3,
    },
    inputs: {
      matrix: [
        ["0", "0"],
        ["0", "0"],
        ["0", "0"],
      ],
      countColumns: 2,
    },
    countRows: 3,
    efficiency: [],
  },
```

```

mutations: {
  change(state, payload) {
    state[payload.matrix].matrix[payload.i][payload.j] = payload.value;
  },
  addColumn(state, payload) {
    for (let i = 0; i < state[payload.matrix].matrix.length; i++) {
      state[payload.matrix].matrix[i].push("0");
    }
    state[payload.matrix].countColumns =
++state[payload.matrix].countColumns;
  },
  addColumnNumber(state, payload) {
    for (let i = 0; i < state[payload.matrix].matrix.length; i++) {
      const temp = [];
      for (let j = 0; j < payload.number; j++) {
        temp.push(state[payload.matrix].matrix[i][j] || "0");
      }
      state[payload.matrix].matrix[i] = temp;
    }
    state[payload.matrix].countColumns = +payload.number;
  },
  addRow(state) {
    const tempOutput = [];
    for (let i = 0; i < state.outputs.matrix[0].length; i++) {
      tempOutput.push("0");
    }
    state.outputs.matrix.push(tempOutput);
    const tempInput = [];
    for (let i = 0; i < state.inputs.matrix[0].length; i++) {
      tempInput.push("0");
    }
  }
}

```

```

    }
    state.inputs.matrix.push(tempInput);
    state.countRows = ++state.countRows;
  },
  addRowNumber(state, payload) {
    const tempOutput = [];
    for (let i = 0; i < payload.number; i++) {
      tempOutput.push(
        state.outputs.matrix[i] ||
        new Array(state.outputs.matrix[0].length).fill("0")
      );
    }
    state.outputs.matrix = tempOutput;
    const tempInput = [];
    for (let i = 0; i < payload.number; i++) {
      tempInput.push(
        state.inputs.matrix[i] ||
        new Array(state.inputs.matrix[0].length).fill("0")
      );
    }
    state.inputs.matrix = tempInput;
    state.countRows = +payload.number;
  },
  deleteColumn(state, payload) {
    if (state[payload.matrix].matrix[0].length > 1) {
      const temp = [];
      for (let i = 0; i < state[payload.matrix].matrix.length; i++) {
        const row = [];
        for (let j = 0; j < state[payload.matrix].matrix[i].length; j++) {
          if (j !== payload.index) {

```

```

        row.push(state[payload.matrix].matrix[i][j]);
    } }
    temp.push(row);
}
state[payload.matrix].matrix = temp;
state[payload.matrix].countColumns = --state[payload.matrix]
    .countColumns;
} },
deleteRow(state, payload) {
    if (state.outputs.matrix.length > 1) {
        const tempOutput = [];
        for (let i = 0; i < state.outputs.matrix.length; i++) {
            if (i !== payload.index) {
                tempOutput.push(state.outputs.matrix[i]);
            }
        }
        state.outputs.matrix = tempOutput;
        const tempInput = [];
        for (let i = 0; i < state.inputs.matrix.length; i++) {
            if (i !== payload.index) {
                tempInput.push(state.inputs.matrix[i]);
            }
        }
        state.inputs.matrix = tempInput;
        state.countRows = --state.countRows;
    } },
calculate(state) {
    main: for (let i = 0; i < state.outputs.matrix.length; i++) {
        let weight =
            1 / state.outputs.matrix[i].reduce((sum, number) => sum + +number, 0);
        let efficiency = [];
        for (let j = 0; j < state.outputs.matrix.length; j++) {

```

```

    const efficient = state.outputs.matrix[j].reduce(
      (sum, number) => sum + +number * weighth,
      0, );
    if (efficient > 1) { continue main; }
    efficiency.push(efficient);
  }
  state.efficiency = efficiency;
  break main;
} },

readFile(state, text) {
  const array = text.split("\n");
  array.push("\r");
  let matrix = [];
  let count = 1;
  for (let i = 0; i < array.length; i++) {
    if (array[i] !== "\r") {
      const text = array[i].split("\r");
      const line = text[0].split(" ");
      matrix.push(line);
    } else {
      switch (count) {
        case 1: {
          state.outputs.matrix = matrix;
          state.outputs.countColumns = matrix[0].length;
          matrix = [];
          count++;
          break;
        }
        case 2: {
          state.inputs.matrix = matrix;

```

```

state.inputs.countColumns = matrix[0].length;
state.countRows = matrix.length;
matrix = [];
count++;
break;
}}}}}}});

```

A.5 Скрипт Home.vue

```

<template>
  <div class="custom-container">
    <div class="logo text-white text-center">Data Envelopment
    Analysis</div>
    <div class="d-flex">
      <div class="d-inline-flex mb-10px mx-auto overflow-x-auto">
        <table>
          <tr> <td></td>
          <td class="d-inline-flex align-center pbl-10px">
            <div class="text-white text-bold mr-10px">
              Outputs
            </div>
            <v-text-field
              class="input-number"
              type="number"
              min="1"
              max="999"
              :value="outputs.countColumns"
              v-on:blur="
                (event) => addColumnNumber(event.target.value, 'outputs')
              "
            />

```

```

    <v-icon
      dark
      class="cursor-pointer"
      @click="() => addColumn('outputs')"
    >mdi-arrow-right</v-icon >
  </td>    </tr>
<tr>
  <td class="d-inline-block text-center ptr-10px">
    <v-text-field
      class="input-number"
      type="number"
      min="1"
      max="999"
      :value="countRows"
      v-on:blur="(event) => addRowNumber(event.target.value)" />
    <v-icon dark class="cursor-pointer" @click="addRow"
      >mdi-arrow-down</v-icon
    >    </td>
  <td>
    <Table
      :name=""outputs""
      :matrix="outputs.matrix"
      :countColumns="outputs.countColumns"
      :countRows="countRows"
    />    </td> </tr>    </table>
<table>
  <tr> <td class="d-inline-flex align-center pbl-10px">
    <div class="text-white text-bold mr-10px">
      Inputs
    </div>

```

```

    <v-text-field
      class="input-number"
      type="number"
      min="1"
      max="999"
      :value="inputs.countColumns"
      v-on:blur="
        (event) => addColumnNumber(event.target.value, 'inputs')
      "
    />

    <v-icon
      dark
      class="cursor-pointer"
      @click="() => addColumn('inputs')"
      >mdi-arrow-right</v-icon > </td>
  </tr>

  <tr> <td>
    <Table
      :name=""inputs""
      :matrix="inputs.matrix"
      :showLeftNumbers="false"
      :countColumns="inputs.countColumns"
      :countRows="countRows"
    />
  </td> </tr>
</table>
</div> </div>

<div class="text-center btn-group">
  <div class="mb-10px">
    <input id="fileInput" type="file" class="d-none" @change="file" />
    <v-btn x-small @click="clickFile">File input</v-btn>
  
```

```

    </div>
    <div class="pb-20px">
        <v-btn x-large @click="calculate">Calculate</v-btn>
    </div> </div> </div>
</template>
<script>
import Table from "../components/Table";
export default {
  name: "Home",
  components: {Table},
  computed: {
    outputs() { return this.$store.state.outputs; },
    inputs() { return this.$store.state.inputs; },
    countRows() { return this.$store.state.countRows; },
  },
  methods: {
    addColumn(name) {
      this.$store.commit("addColumn", { matrix: name });
    },
    addColumnNumber(number, name) {
      this.$store.commit("addColumnNumber", {
        number: number,
        matrix: name,
      });
    },
    addRow() { this.$store.commit(`addRow`); },
    addRowNumber(number) {
      this.$store.commit("addRowNumber", {
        number: number,
      });
    },
    file(event) {
      const file = event.target.files[0];

```

```

const reader = new FileReader();
reader.readAsText(file, "UTF-8");
reader.onload = (e) => {
  const text = e.target.result;
  this.$store.commit("readFile", text);
}; },
clickFile() { document.getElementById("fileInput").click(); },
calculate() {
  this.$store.commit("calculate");
  this.$router.push("results");
}, },});
</script>
<style lang="scss">
input[type="number"]::-webkit-inner-spin-button {
  -webkit-appearance: none;}
.input-number { padding: 0; margin: 0; width: 30px;
.v-input__slot { margin: 0;
  &::before { border-color: white !important; }
  input { text-align: center; color: white; }
}
.v-text-field__details { display: none; }
} .btn-group { .v-btn {
  background-color: transparent !important;
  box-shadow: unset !important;
  color: white !important;
  border: 2px solid white !important;
}}
</style>

```

A.6 Скрипт Results.vue

```

<template> <div class="custom-container">
  <div class="logo text-white text-center">Results</div>
  <div class="d-flex">
    <div class="d-inline-flex mb-10px mx-auto overflow-x-auto">
      <table> <tr> <td>
        <div class="text-white text-bold mr-10px"> Outputs
      </div> </td> </tr>
      <tr> <td> <Table
        :name=""outputs""
        :matrix=""outputs.matrix""
        :readonly=""true" />
      </td> </tr> </table>
      <table>
        <tr> <td> <div class="text-white text-bold mr-10px"> Inputs
          </div> </td> </tr> <tr> <td>
            <Table
              :name=""inputs""
              :matrix=""inputs.matrix""
              :showLeftNumbers=""false""
              :readonly=""true" />
            </td> </tr> </table> </div> </div>
    <div class="d-flex">
      <div class="d-inline-flex mx-auto">
        <div class="mr-30px">
          <canvas id="myChart" height="450" width="600"></canvas>
        </div>
        <table class="result-table">
          <tr>
            <td>#</td>

```

```

        <td>Efficiency</td>      </tr>
      <tr v-for="(number, i) in efficiency" :key="i">
        <td>{{ i + 1 }}</td>
        <td>{{ number.toFixed(4) }}</td>
      </tr> </table>  </div> </div>

<div class="text-center mt-10px">
  <div class="pb-20px">
    <v-btn class="back-btn" @click="back">Back</v-btn>
  </div> </div> </div> </template>

<script>
import Table from "../components/Table";
import Chart from "chart.js";
export default {
  name: "Results", components: { Table, },
  computed: {
    outputs() { return this.$store.state.outputs; },
    inputs() { return this.$store.state.inputs; },
    efficiency() { return this.$store.state.efficiency; },
  },
  mounted() {
    var ctx = document.getElementById("myChart").getContext("2d");
    Chart.defaults.global.defaultFontColor = "white";
    Chart.defaults.global.defaultColor = "white";
    new Chart(ctx, { type: "bar", data: {
      labels: this.efficiency.map((number, i) => `${i + 1}`),
      datasets: [
        {
          data: this.efficiency,
          backgroundColor: "white",
          borderColor: "white",
        },
      ],
    }},
  },

```

```

    options: {      legend: {      display: false,      },
    scales: { yAxes: [      {
        ticks: { beginAtZero: true,      },
    }, ], }, }, }); },
methods: { back() { this.$router.push("/"); }  }};
</script>
<style lang="scss">
.result-table {
  color: white;
  border-spacing: 0;
  border-collapse: collapse;
  td {
    border: 3px solid white;
    min-width: 40px;
    text-align: center;
    padding: 10px;
  }
}
.back-btn {
  background-color: transparent !important;
  box-shadow: unset !important;
  color: white !important;
  border: 2px solid white !important;
} </style>

```

A.7 Скрипт Table.vue

```

<template>
  <table class="table">
    <thead>

```

```

<tr> <td v-if="showLeftNumbers"> # </td>
  <td class="countColumns > 1 && !readonly ? 'number-with-button' : ""
    v-for="(number, index) in matrix[0]"
    :key="index"      >
    <div class="number">{{ index + 1 }}</div>
    <v-btn
      v-if="countColumns > 1 && !readonly"
      class="small-button"
      @click="() => deleteColumn(index, name)"      >
      <v-icon dark>          mdi-minus          </v-icon>
    </v-btn> </td> </tr> </thead>
<tbody>
<tr v-for="(line, i) in matrix" :key="i">
  <td :class="countRows > 1 && !readonly ? 'number-with-button' : ""
    v-if="showLeftNumbers" >
    <div class="number">{{ i + 1 }}</div>
    <v-btn
      v-if="countRows > 1 && !readonly"
      class="small-button"
      @click="() => deleteRow(i)"      >
      <v-icon dark>
        mdi-minus
      </v-icon> </v-btn>      </td>
  <td v-for="(number, j) in line" :key="j">
    <v-text-field
      :value="number"
      @input="(value) => change(value, i, j)"
      :readonly="readonly"
    ></v-text-field>
  </td>

```

```

    </tr>
  </tbody>
</table>
</template>
<script>
export default {  name: "Table",
  props: {
    readonly: { type: Boolean, default: false,  },
    name: String,
    countColumns: { type: Number, default: 0,  },
    countRows: { type: Number, default: 0,  },
    showLeftNumbers: { type: Boolean, default: true,  },
    matrix: { type: Array, required: true,  },  },
  methods: {
    change(value, i, j) { this.$store.commit("change", {
      i, j, value: value, matrix: this.name,  });  },
    deleteColumn(index, matrix) {
      this.$store.commit("deleteColumn", { index, matrix,  });  },
    deleteRow(index) {
      this.$store.commit("deleteRow", { index,  });  },  },
  },
};
</script>
<style lang="scss">
.table { color: white;
border-spacing: 0;
border-collapse: collapse;
td {
border: 3px solid white;
min-width: 40px;
text-align: center;
padding: 10px;

```

```

.v-input {
  padding: 0;
  margin: 0;
  min-width: 60px;
  .v-input__slot { margin-bottom: 0;
    &::before { border-color: white !important; }
    input {
      text-align: center;
      color: white; } }
  .v-text-field__details { display: none; } } }
.small-button {
  padding: 0 !important;
  height: 30px !important;
  min-width: 30px !important;
  background-color: transparent !important;
  box-shadow: unset !important;
  color: white !important;
  border: 2px solid white !important;
  .v-icon { font-size: 24px; }
}
.number-with-button {
  .number { display: unset; }
  .small-button { display: none; }
  &:hover { padding: 0;
    .number { display: none; }
    .small-button { display: unset; }
  } } }
</style>

```

ДОДАТОК Б
Слайди презентації

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Дослідження і програмна реалізація методів мотивації і управління командою розробки ІТ-стартапу

Виконав студент групи КНТ-220м
Іхсанов Д.М.

Керівник канд.техн.наук, проф.
Дубровін В.І.

2021

Рисунок Б.1 – Дослідження і програмна реалізація методів мотивації і управління командою розробки ІТ-стартапу

Актуальність

Стартап або стартап-компанія (від англ. Start-up - запускати) - це тип бізнесу, спрямований на отримання доходу шляхом реалізації принципово нової ідеї.

За статистичними даними, лише 10% стартапів стають успішними. В умовах нестабільності та конкуренції на міжнародному ринку ІТ послуг процес управління компанією залежить від впливу великої кількості внутрішніх та зовнішніх чинників. Компанії - початківці стикаються з цілою низкою проблем, наприклад: неправильний підбір персоналу, відсутність бізнес-моделі, незатребуваний на ринку товар, якісне відстеження внутрішніх та зовнішніх ризиків, неправильне використання бюджету.

Таким чином, розробка програмного додатку для аналізу ефективності праці членів команди є актуальною проблемою, вирішення якої допоможе командам розробки оптимізувати свою роботу.

Рисунок Б.2 – Актуальність дослідження

- ▶ *Об'єкт дослідження* - програмні засоби методів мотивації і управління командою.
- ▶ *Предмет дослідження* - методи управління командою розробників.
- ▶ *Мета роботи* - створення програмної системи для автоматизації методів мотивації і управління командою розробки IT-стартапу.
- ▶ *Матеріали, методи та технічні засоби*: мова програмування JavaScript, програмна платформа Node.js та JavaScript фреймворк для створення інтерфейсу користувача Vue.js, реляційна база даних PostgreSQL, персональний комп'ютер з процесором Intel Core i5 під управлінням операційної системи Microsoft Windows 10.

3

Рисунок Б.3 – Мета, об'єкт та предмет дослідження

Методи ефективності функціонування систем

Існує ряд методів, які використовуються для оцінки ефективності об'єктів в досить широких класах систем:

- ▶ параметричний підхід - апіорне завдання функціональної форми, яка описує кордон, що дозволяє застосовувати для її оцінювання економетричних методів (метод стохастичною кордону - stochastic frontier approach, SFA, метод товстої кордону - thick frontier approach, TFA і метод вільного розподілу - distribution-free approach, DFA);
- ▶ непараметричний підхід - показники ефективності визначаються методами лінійного програмування (різні варіації моделей методу огинаючих - data envelopment analysis, DEA).

Переваги непараметричного підходу: одночасний облік всіх витрат і випусків, відсутність апіорних обмежень на функціональну форму кордону.

Недоліки непараметричного підходу - необхідність великої кількості об'єктів для стійкості результатів.

4

Рисунок Б.4 – Методи дослідження

Метод Data Envelopment Analysis (DEA)

Основна ідея – це оцінка відносної ефективності функціонування об'єктів і побудова межі ефективності, яка має форму опуклої оболонки і використовується як еталон для проведення оцінки ефективності об'єктів у досліджуваній сукупності. Міра ефективності у методі DEA показує кількість відсотків, на яку керівництво об'єкту має збільшити або зменшити вхідні або вихідні показники з метою досягнення рівня ефективності, оскільки існують інші об'єкти, реальні або гіпотетичні, що функціонують оптимально.

$$\max_{\theta, u} \theta = \frac{\sum_{j=1}^s u_j y_j}{\sum_{i=1}^m v_i x_i}, \text{ де}$$

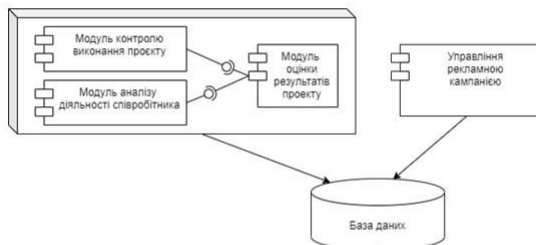
- θ - значення ефективності об'єкта;
- x_i, y_j - відповідно вхід та вихід для об'єкта;
- v_i, u_j - це ваги, які необхідно розрахувати таким чином, щоб отримати максимальну оцінку дробу;
- s, m - кількість входу та виходу відповідно.

5

Рисунок Б.5 – Метод DEA

Розробка архітектури програмного додатку

Загальна архітектура додатку.
Взаємодія модулів та бази даних:



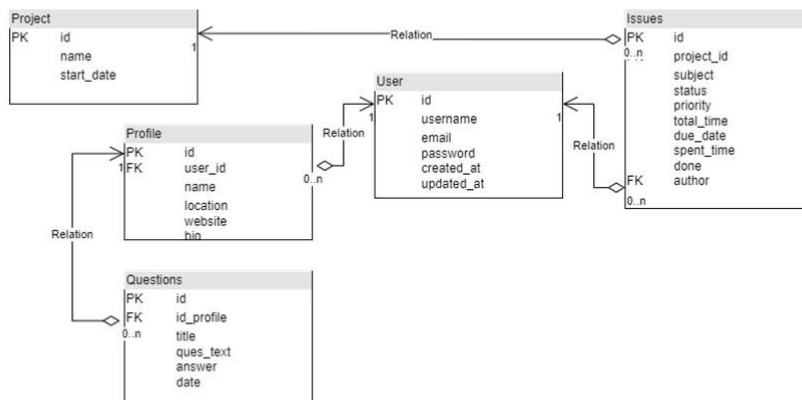
Архітектура додатку. Компоненти:



6

Рисунок Б.6 – Архітектура програмного додатку

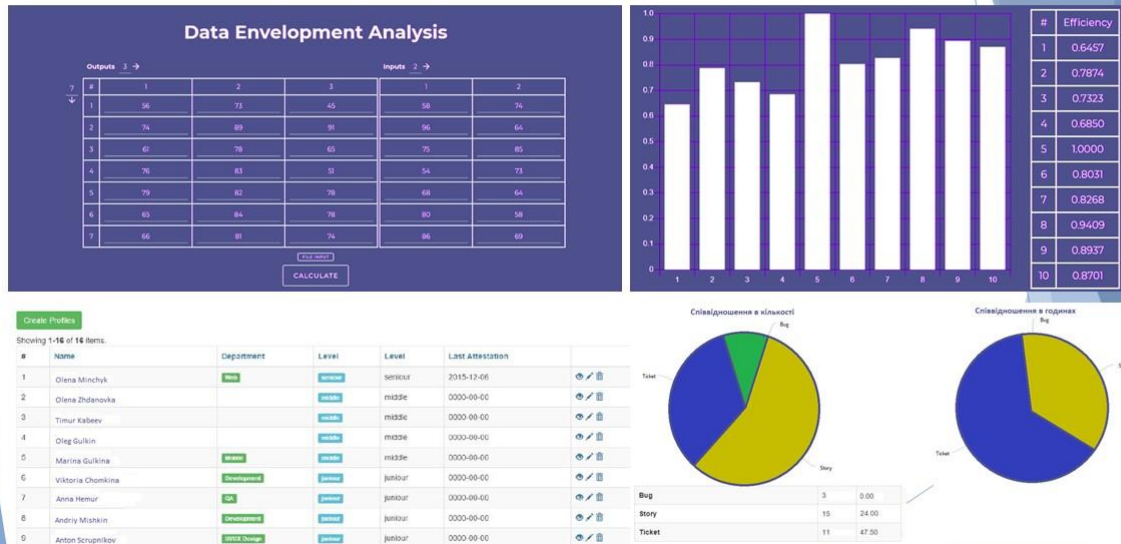
База даних програмного додатку



7

Рисунок Б.7 – Розробка бази даних

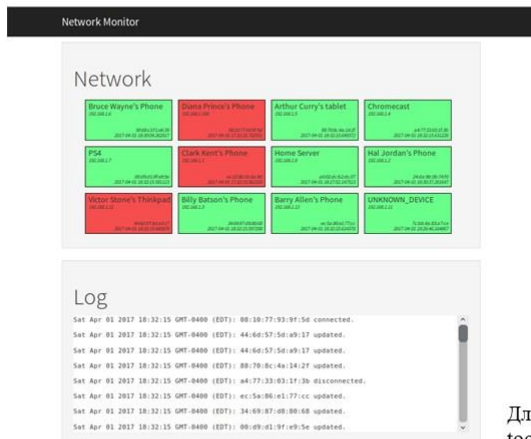
Інтерфейс програмного додатку



8

Рисунок Б.8 – Інтерфейс програмного додатку

Тестування



```

import random
from scapy.all import *
target_IP = input("Enter IP address of Target: ")
i = 1

while True:
    a = str(random.randint(1,254))
    b = str(random.randint(1,254))
    c = str(random.randint(1,254))
    d = str(random.randint(1,254))
    dot = "."
    Source_ip = a + dot + b + dot + c + dot + d

    for source_port in range(1, 65535):
        IP1 = IP(source_IP = source_IP, destination = target_IP)
        TCP1 = TCP(srcport = source_port, dstport = 80)
        pkt = IP1 / TCP1
        send(pkt, inter = .001)

    print ("packet sent ", i)
    i = i + 1
  
```

Для тестування додатку використані інструменти Google Chrome dev tools та команди docker для перевірки коректної роботи контейнерів та можливості входу до контейнеру. Додаток пройшов перевірку на працезданість Web UI, скрипту для пошуку IP адресів та роботи серверу.

9

Рисунок Б.9 – Тестування програмного додатку

Висновки

В ході розробки створено додаток для дослідження методів мотивації і управління командою розробки IT-стартапу. В проєкті були використані наступні технології: мова програмування JavaScript, програмна платформа Node.js та JavaScript фреймворк для створення інтерфейсу користувача Vue.js, реляційна база даних PostgreSQL, персональний комп'ютер з процесором Intel Core i5 під управлінням операційної системи Microsoft Windows 10.

Основною функцією розробленого додатку є: розрахунок ефективності співробітників на кожному етапі реалізації стартапу. Таким чином, було прийнято рішення реалізувати окрему програмну систему зі зручним інтерфейсом користувача. В результаті дослідження проаналізовано особливості стартап-проектів та керування командою розробників, розглянуто методи керування співробітників у стартап-проекті, створено програмний веб-застосунок для автоматизації методів мотивації і управління командою розробки IT-стартапу. Галуззю використання є IT-компанії та установи, що мають у своєму складі відділи IT.

10

Рисунок Б.10 – Висновки за дослідженням