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

Пояснювальна записка

до магістерської кваліфікаційної роботи

на тему: «Інформаційна технологія моніторингу виконання завдань»

Виконав: студент 2 курсу,
групи 2КН-18м
Спеціальності 122 «Комп'ютерні науки»
Кміть В.Я.

Керівник: PhD, проф. Савчук Т.О.

Рецензент: к.т.н.,доц.
Коваленко О.О.

Вінниця
2019 рік

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

_____
(підпис)
“____” _____2019 року

ЗАВДАННЯ

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

08-22.МКР.029.18.000.ПЗ

Магістранта групи 2КН-18м Кмітя Владислава Ярославовича

Тема магістерської кваліфікаційної роботи: «Інформаційна технологія моніторингу виконання завдань»

Вхідні дані: кількість завдань – необмежена, можливе додавання коментарій, кількість облікових записів – не менше 2, кількість користувачів – не менше 2, дедлайн роботи – визначений, мова програмування – об’єктно-орієнтована, сервер баз даних - кросплатформлений.

Короткий зміст частин магістерської кваліфікаційної роботи:

1. Графічна: Узагальнений алгоритм моніторингу виконання завдань; Модель мікросервісного моніторингу виконання завдань; Структура сервісів з чергою повідомлень; UML – діаграма алгоритму формуваня завдань керівником проекту; UML – діаграма алгоритму моніторингу виконання завдань з урахуванням дедлайну; UML-діаграма прецедентів інформаційної технології моніторингу виконання завдань; Схема взаємодії вузлів інформаційної технології моніторингу виконання завдань; Структурна схема інтерфейсу інформаційної технології моніторингу виконання завдань; Графічна схема інтерфейсу сторінки користувача для управління завданнями; UML-діаграма роботи оповіщень інформаційної технології моніторингу виконання завдань .
2. Текстова (пояснювальна записка): вступ, сучасні моделі, методи і засоби моніторингу виконання завдань, розробка моделі моніторингу виконання завдань, розробка інформаційної технології моніторингу виконання завдань, економічна частина.

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ МКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз сучасних моделей, методів і засобів моніторингу виконання завдань			Аналітичний огляд літературних джерел, задачі досліджень, розділ 1 ПЗ
2	Порівняльний аналіз методів розв'язування задач моніторингу			Метод, інформаційна технологія, розділ 2
3	Проектування програмних засобів інформаційної технології моніторингу виконання завдань			Програмне забезпечення, розділ 3, 4
4	Підготовка економічної частини			розділ 5
5	Апробація та/або впровадження результатів дослідження			тези доповідей/акт впровадження
6	Оформлення пояснювальної записки, графічного матеріалу та презентації			Пояснювальна записка, графічний матеріал, презентація

Консультанти з окремих розділів магістерської кваліфікаційної роботи

1. Науковий керівник _____ PhD, професор кафедри КН
(підпис) наук. ступінь, вчене звання (посада)
"____" _____20___ р. _____Т.О. Савчук_____
ініціали та прізвище

2. Економічна частина _____ канд. екон. наук, доц. кафедри ЕПВМ
(підпис) наук. ступінь, вчене звання (посада)
_____М. В. Бальзан_____
ініціали та прізвище
"____" _____20___ р.

Дата попереднього захисту роботи "____" _____20___ р.

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

Завдання видав
науковий керівник _____ PhD, професор кафедри КН
(підпис) наук. ступінь, вчене звання (посада)
_____Т.О. Савчук_____
ініціали та прізвище
"____" _____20___ р.

Завдання отримав магістрант _____ _____ В.Я. Кміть_____
(підпис) ініціали та прізвище
"____" _____20___ р.

АНОТАЦІЯ

Магістерську кваліфікаційну роботу присвячено інформаційній технології моніторингу виконання завдань, удосконалено методи формування завдань керівником проекту, моніторинг виконання завдань з урахуванням дедлайну та розроблено програмне забезпечення для вирішення задачі моніторингу виконання завдань.

Програмна реалізація інформаційної технології моніторингу виконання завдань написана мовою програмування C#, а web-додаток – з використанням сучасного фреймворку React.

Тестування інформаційної технології моніторингу виконання завдань виконано у середовищах розробці Microsoft Visual Studio, Microsoft Visual Studio Code та експерментальним методом.

ANNOTATION

The master's qualification work is devoted to information technology of task performance monitoring, methods are improved and software is developed to solve the task performance monitoring task.

The software implementation of task monitoring information technology is written in C # programming language, and the web application is a modern React framework.

Testing of information technology for monitoring the fulfillment of tasks was performed in the Microsoft Visual Studio and Microsoft Visual Studio Code development environments and experimentally.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Моніторинг — це комплексна система спостережень, збору, обробки, систематизації та аналізу інформації щодо виконання державним службовцем визначених для нього завдань, яка дає оцінку і прогнозує їх зміни, перегляд (уточнення) [1].

Кожному керівнику структурного підрозділу доцільно розробити систему моніторингу, яка б включала збір, аналіз і використання даних/інформації про ступінь досягнення ключових показників кожного службовця, який перебуває у підпорядкуванні.

Варто врахувати, що моніторинг забезпечує відслідковування прогресу у виконанні завдань (наскільки державний службовець просувається у досягненні ключових показників), на відміну від оцінювання, яке визначає, наскільки результат вже досягнуто і завдання виконано відповідно до запланованого.

Дані моніторингу можуть бути підставою для коригування, уточнення ключових показників та завдань службовця.

Результати моніторингу не рекомендується розглядати як результати оцінювання і, таким чином, вони не можуть бути використаними для застосування дисциплінарних стягнень. Вони, в першу чергу, можуть розглядатись як засіб виявлення незадовільного стану, прогалин, відставання у часі для подальшого ухвалення рішень щодо виправлення поточного стану.

Тема магістерської кваліфікаційної роботи є актуальною, оскільки існує потреба у створенні автоматизованого програмного засобу моніторингу виконання завдань для організацій та компаній з удосконалиними методами та застусуванням сучасних web-ехнологій. Програмний засіб є здатним, на підставі аналізу терміну виконання завдання, надавати рекомендації щодо вчасного виконання поставлених задач [2].

Зв'язок роботи з науковими програмами, планами, темами. Магістерська робота виконана відповідно до напрямку наукових досліджень кафедри комп’ютерних наук Вінницького національного технічного

університету 22 К1 «Моделі, методи, технології та пристрої інтелектуальних інформаційних систем управління, економіки, навчання та комунікацій» та плану наукової та навчально-методичної роботи кафедри.

Мета та завдання дослідження. Метою дослідження є підвищення швидкодії та розширення функціоналу моніторингу виконання завдань.

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

– провести аналіз сучасних моделей моніторингу виконання завдань;
– розглянути аналіз сучасних методів моніторингу виконання завдань;
– провести аналіз сучасних засобів моніторингу виконання завдань;
– провести удосконалення моделі моніторингу виконання завдань;
– провести удосконалення методів моніторингу виконання завдань;
– сформулювати стадії інформаційної технології та на їх основі розробити структуру та алгоритм роботи програмного засобу;
– виконати програмну реалізацію інформаційної технології моніторингу виконання завдань;
– провести тестування інформаційної технології моніторингу виконання завдань.

Об'єкт дослідження – процес моніторингу виконання завдань.

Предмет дослідження – програмні засоби моніторингу виконання завдань.

Наукова новизна одержаних результатів полягає в наступному:

- вперше запропоновано інформаційну технологію моніторингу виконання завдань, яка здатна, на підставі аналізу терміну виконання завдання, надавати рекомендації щодо вчасного виконання поставлених задач;
- удосконалено модель моніторингу виконання завдань за рахунок впровадження різного інструментарію можливостей, при виборі якого будуть задовільнятися усі критерії моніторингу організації;
- удосконалено метод моніторингу виконання завдань за рахунок інтеграції облікових записів Telegram, Email користувачів, що дозволить

ефективніше контролювати виконування завдання від керівника організації та швидше отримувати оповіщення про завдання на мобільний телефон чи інші девайси з підтримкою облікових записів;

Практичне значення одержаних результатів полягає у наступному:

1. Розроблено новий спосіб моніторингу виконання завдань для компаній та організацій незалежно від їх маштабів.
2. Розроблено алгоритм формування завдань керівником проекту.
3. Розроблено алгоритм моніторингу виконання завдань з урахуванням дедлайну роботи.
4. Розроблено програмний засіб моніторингу виконання завдань.

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

Апробація результатів роботи. Результати дослідження позитивно апробовані на Всеукраїнській науково-практичній інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи» та на XLVIII Науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії (м. Вінниця, Україна, 2019 р.) та опубліковані у збірниках даних конференцій.

Публікації. За результатами досліджень подано заявку на реєстрацію авторського права на твір (комп'ютерну програму) “Інформаційна технологія моніторингу виконання завдань” (номер реєстрації заявки АПС/10094-19), а також опубліковано 2 тез доповіді з науково-технічних конференцій.

1 СУЧАСНІ МОДЕЛІ, МЕТОДИ І ЗАСОБИ МОНІТОРИНГУ ВИКОНАННЯ ЗАВДАНЬ

1.1 Аналіз моделей і методів моніторингу виконання завдань

Керівнику сучасних проектів приходиться контролювати паралельне виконання потужними командами множини задач. Існуючі підходи до рішення задачі моніторингу виконання завдань характеризуються відсутністю можливості визначити статус поставленої задачі та групувати контакти виконавців, складністю реєстрації користувачів засобів, що виконують моніторинг, недоліками коду програмного продукту, які реалізують процес [3].

Згідно з концепцією, моніторинг виконує одну з важливіших функцій системи управління з досягнення поставлених перед об'єктом цілей і є невіддільною складовою управління об'єктом у частині формування інформаційно-аналітичної бази цього процесу. Сучасний менеджмент розділяє цілі будь-якого економічного об'єкта на дві групи: стратегічні (довгострокові, перспективні) та оперативні (короткострокові). За цією ознакою відбувається класифікація систем моніторингу на стратегічний та оперативний. Отже, моніторинг дає змогу здійснювати спостереження за досягненням як стратегічних, так і оперативних цілей діяльності об'єкта [4-5].

Модель має декілька застосувань: по-перше, вона чітко визначає компоненти, які складають систему; по-друге, достатньо схематично та точно наводить зв’язки між компонентами; по-третє, модель генерує, породжує питання і нарешті стає інструментом для порівняльного вивчення різних галузей явища, процесу. До моделювання звертаються тоді, коли неможливо одразу розпочати пізнання сутності зацікавленого об’єкта і немає умов для безпосереднього оволодіння ним. До таких об’єктів належить система моніторингу.

Оглянемо сучасну концептуальну та базову моделі моніторингу виконання завдань [6]. Концептуальна модель моніторингу виконання завдань, основу якої становлять процесуальність, технологічність, повторюваність

(періодичність), ініційованість, у якій виокремлюються параметри, фактори і критерії різних порядків [7-8]. За параметри беруться величини, які характеризують основні якості об’єкта, що відповідають глобальним цілям об’єкта. Базова модель моніторингу виконання завдань не так характеризує основі якості і цілі обєкта, як концептуальна модель. Базова модель моніторингу включає у себе поверхневі методи критерій оцінки та методологічні аспекти моніторингу виконання завдань. Порівняльна характеристика моделей моніторингу зображена у таблиці 1.1. Здійснення моніторингу відбувається за допомогою таких напрямів:

– якість умов;
– якість процесів;
– якість результатів.

Таблиця 1.1 - Порівняльна характеристика моделей моніторингу виконання завдань.

Характеристики	Базова модель	Концептуальна модель
Технологічність	+	-
Процесуальність	-	+
Періодичність	+	+
Точність поставлення задачі	-	+
Критерії оцінки	-	+
Основні цілі об’єкта	+	+

Базова модель не спроможна виконати усіх поставлених задач моніторингу виконання завдань. Види моніторингу можна класифікувати за різними принципами [9-10]. В залежності від принципів, що можуть бути використані для порівняння, виділяються такі види моніторингу:

– динамічний, коли в якості основи для експертизи беруться дані про динаміку розвитку того або іншого об'єкта, явища або показника. Це найпростіший спосіб, який може бути аналогом експериментального плану тимчасових серій. Для відносно простих систем, локального моніторингу (цін, прибутків населення й ін.) або моніторингу фізичних об'єктів цього підходу може виявитися достатньо.

– конкурентний, коли в якості основи для експертизи вибираються результати ідентичного обстеження інших систем. У цьому випадку моніторинг стає аналогом плану з множинними серіями іспитів

– порівняльний, коли в якості основи для експертизи, вибираються результати ідентичного обстеження однієї або двох систем більш високого рівня.

Контроль робочого процесу - дуже делікатне питання. Керівники усіма руками "за", співробітники усіма руками-ногами проти. Одні переоцінюють контроль, інші навпаки, втрачають його значення. Але він необхідний на кожному підприємстві. Персонал далеко не такий відповідальний, як хотілося, завдання часто виконуються не зовсім так, як планувалося, бюджет витрачається даремно.

На сьогоднішній день є велика кількість методів для моніторингу виконання завдань, деякі з них ефективні, а даякі не на стільки, як хотілось [11].

Найпростіший та найлегший метод – нагляд. Керівник зобов'язується час від часу виходити з власного кабінету і ходити по офісу в пошуках непрацюючого персоналу. Однак, часто не враховується той факт, що:

– керівник втрачає свій час на виконання потрібних речей;

– власні задачі керівника залишаються невиконаними;

Планові перевірки. Проводяться з певною періодичністю (один раз в тиждень, місяць, квартал) керівником, що перевіряє виконану роботу[12].

Відеонагляд. Ефективним є на підприємствах, де допоможе спостереження: магазин, склад, виробництво.

CRM-системи, що допомагають делегувати завдання і контролювати їх виконання. Крім того, більшість таких сервісів надають статистику по співробітниках [13]. Це найбільш ефективний метод щодо моніторингу виконання завдань.

Порівняльна характеристика методів моніторингу виконання завдань зображена у таблиці 1.2.

Таблиця 1.2 - Порівняльна характеристика методів моніторингу виконання завдань

Характеристики	Нагляд	Планові перевірки	Відеонагляд	CRM-системи
Стеження за персоналом	+	-	+	-
Збереженя завдань	-	+	-	+
Статистика виконаних завдань	-	-	-	+
Визначення точно часу виконаного завдання	+	-	-	+
Аналіз співробітників	+	-	-	+

Аналіз сучасних методів і моделей передбачає створення сучасної інформаційної системи моніторингу виконання завдань, яка задовільнить керівника підрозділу своїм сучасним функціоналом, швидкодією та якістю оцінки виконуючого обов’язків.

Для розробки інформаційної системи моніторингу виконання завдань використано сучасні технології серверної частини Microsoft ASP.NET та реалізовано інтерфейс користувача за допомогою біблеотеки React, яка надає можливість підвищити продуктивність програмного засобу за допомогою рендеринга на стороні сервера.

Отже, проаналізувавши всі методи та моделі моніторингу виконання завдань, можна зробити висновок, що для реалізації усіх поставлених завдань підходить концептуальна модель та CRM метод моніторингу виконання завдань.

1.2 Сучасні засоби моніторингу виконання завдань

У наш час на просторах інтернету можливо знайти безліч варіантів сучасних засобів моніторингу виконання завдань, починаючи від безкоштовних, у яких присутні платні послуги та можливості уже з самої реєстрації користувача у системі та закінчуючи повною платою за програмний засіб [14].

Виділимо декілька щоб порівняти функціонал та можливості кожного.

– «АСІДК Екотест» – це програмне забезпечення для автоматизованої системи індивідуального дозиметричного контролю. Призначене для контролю дозового навантаження персоналу на об’єктах атомної енергетики та в установах, де проводяться роботи з джерелами гамма-випромінення. Надсилання дозиметричних даних на сервер здійснюється зі спеціальних програмних Терміналів, які можуть бути встановлені на довільній кількості автоматизованих робочих місць і працювати незалежно один від одного. Обмін даними між дозиметрами та Терміналами здійснюється безконтактним методом за допомогою інфрачервоного порта Недоліком програмного засобу є плата за користування та вузька спеціальність. Дане забезпечення зображено на рисунку 1.1.

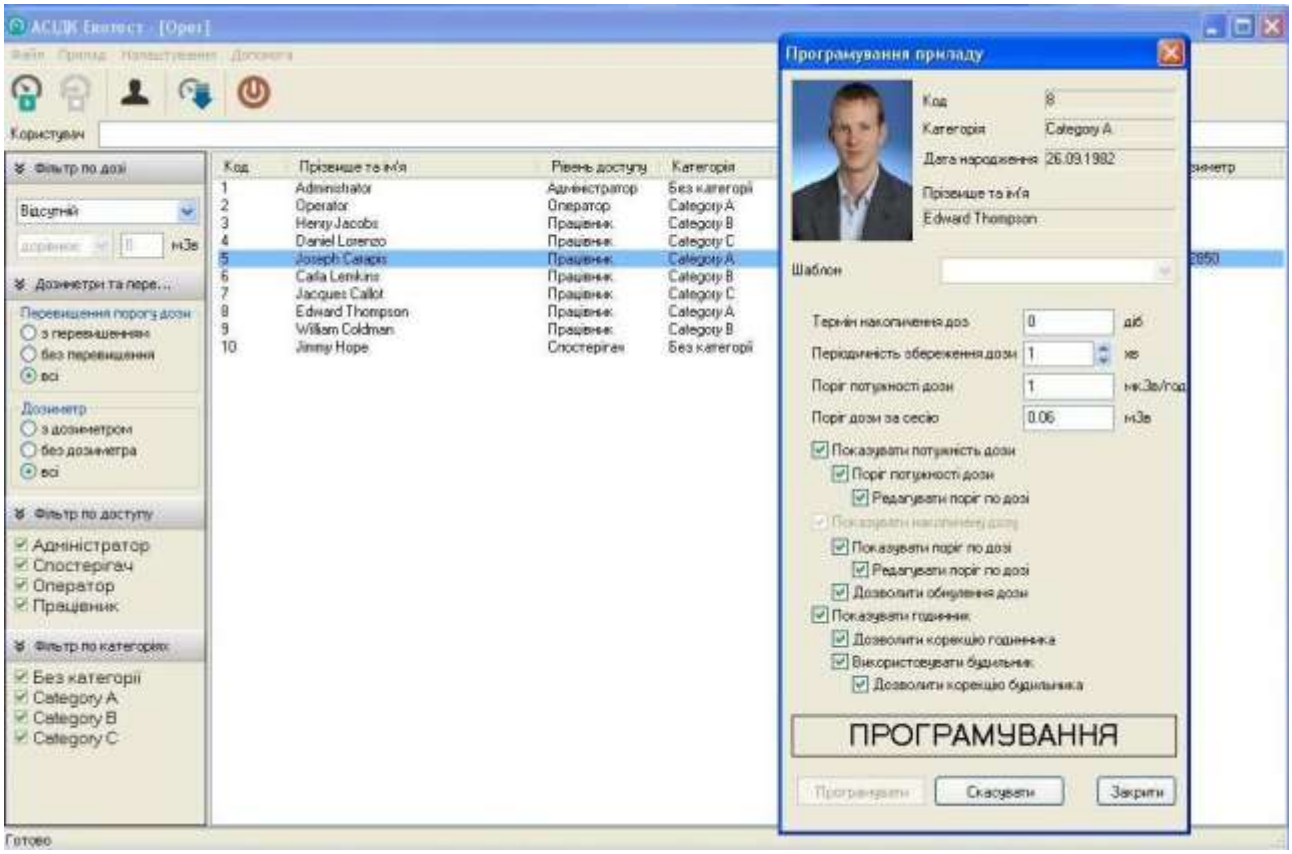


Рисунок 1.1 – Програмне забезпечення «АСІДК Екотест»

– «Босс Контроль» – це хмарний сервіс [15], який складається з терміналів контролю доступу та сервісу обробки даних. Недоліком даної системи є її невелика функціональність. Для запуску системи не будуть потрібні фахівці, які знаються на складному біометричний обладнанні, налаштування серверів, забезпеченні каналів зв'язку. І, головне, не потрібно супроводжувати систему - Ви отримуєте сервіс з гарантованою працездатністю і якістю. Даний сервіс відслідковує лише активність користувачів, проте не може аналізувати виконання поставлених задач та оповіщати працівників про нові задачі. Програмне забезпечення зображено на рисунку 1.2.

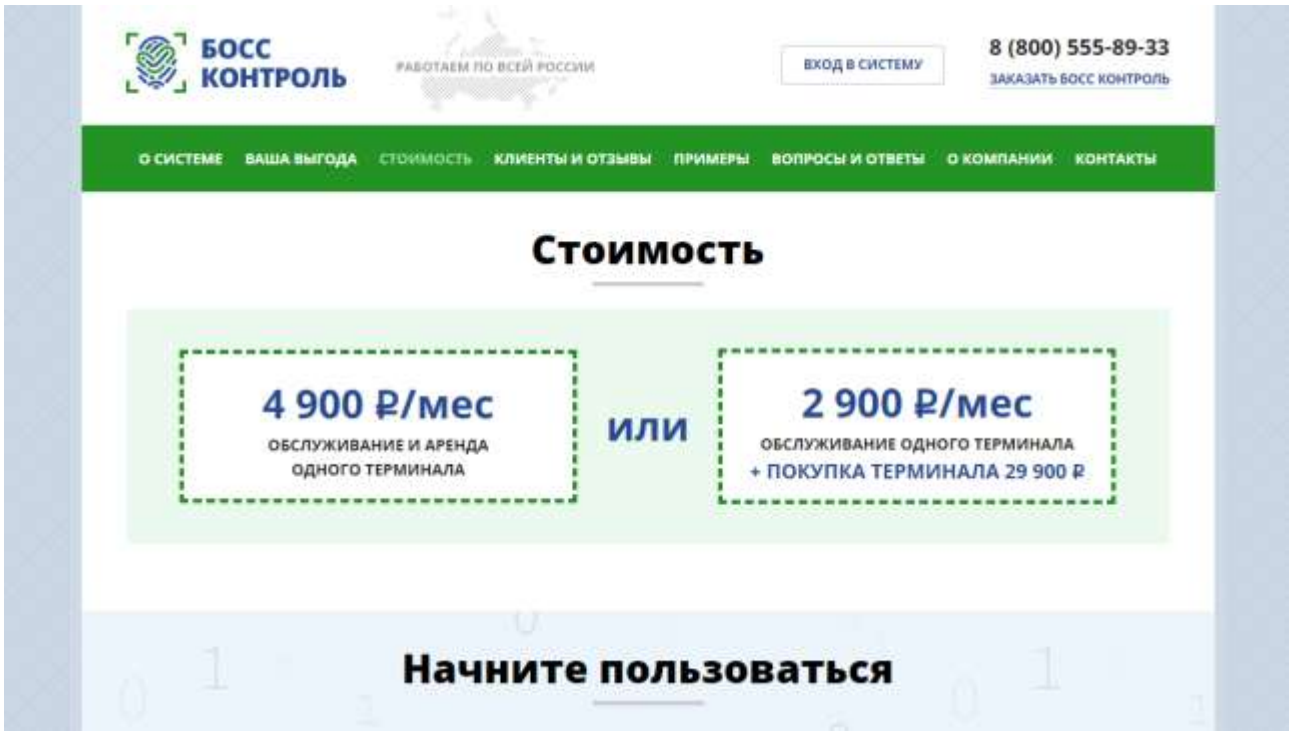


Рисунок 1.2 – Програмне забезпечення «Босс Контроль»

– «LiderTask» – прогрмний засіб, яктй підходить для моніторингу за заводами, фабриками та холдингами. Позитивним моменто цього засобу є зручний інтерфейс та орієнтація на великі сфери моніторингу. Всі прострочені завдання автоматично переносяться в список справ на сьогодні. Можна розбивати великі справи на більш дрібні завдання та організовувати їх в дереві. У ЛідерТаске відсутнє обмеження на глибину вкладеності. Недоліком даного ресурсу є надлишковість даних на інтерфейсі користувача та відсутність неточної оцінки виконаної роботи виконавця. Програмне забезпечення «LiderTask» представлено на рисунку 1.3.

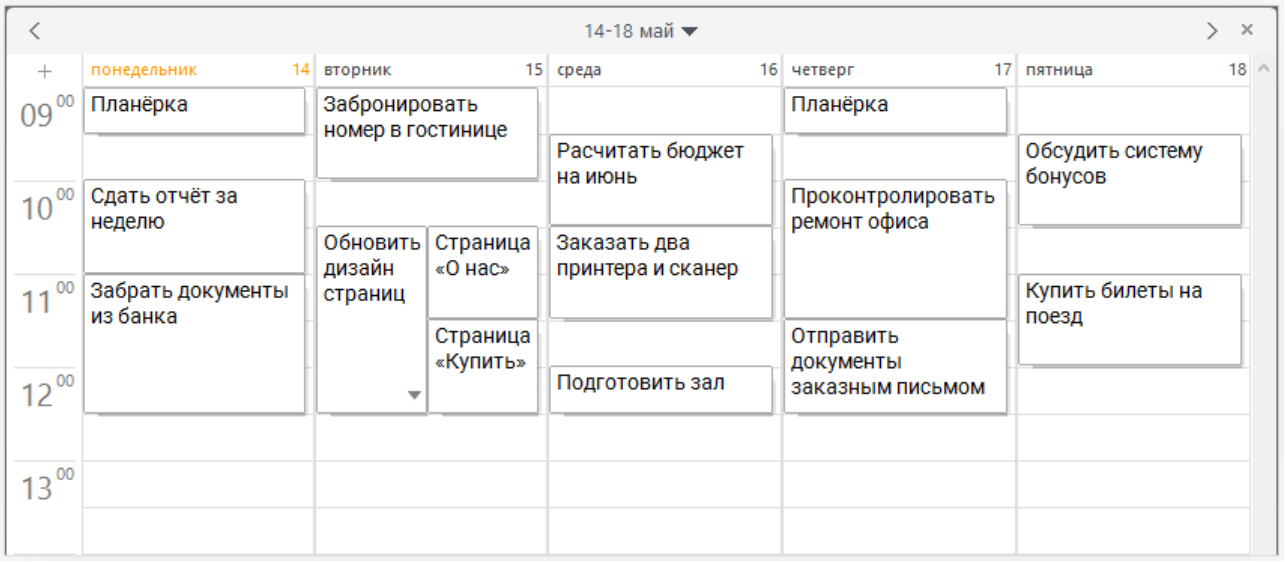


Рисунок 1.3 – Програмне забезпечення «LiderTask»

– «Kickidler» – система обліку робочого часу нового покоління. Основне призначення - автоматизація функції контролю співробітників в організації. Програма надає потужний набір інструментів для моніторингу робочих комп'ютерів і виявлення порушень робочого розпорядку [16]. Система обліку робочого часу Kickidler має функцію тайм-трекера. Тайм-трекер дозволяє контролювати робочі години співробітників і підрозділів. Переробки, запізнення, ранні відходи, прогули, перекури, прогулянки та інші відволікання персоналу від робочого процесу автоматично фіксуються в звітах і табелі робочого часу. Дані в звітах відображаються у вигляді графіків і діаграм. Аналіз ефективності використання робочого часу покаже скільки часу співробітник працював, а скільки - займався особистими справами або просто не діяв. Недолік засобу є обмеження у функція роботи, моніторинг не більше 6 комп’ютерів у мережі. Програмне забезпечення зображено на рисунку 1.4.

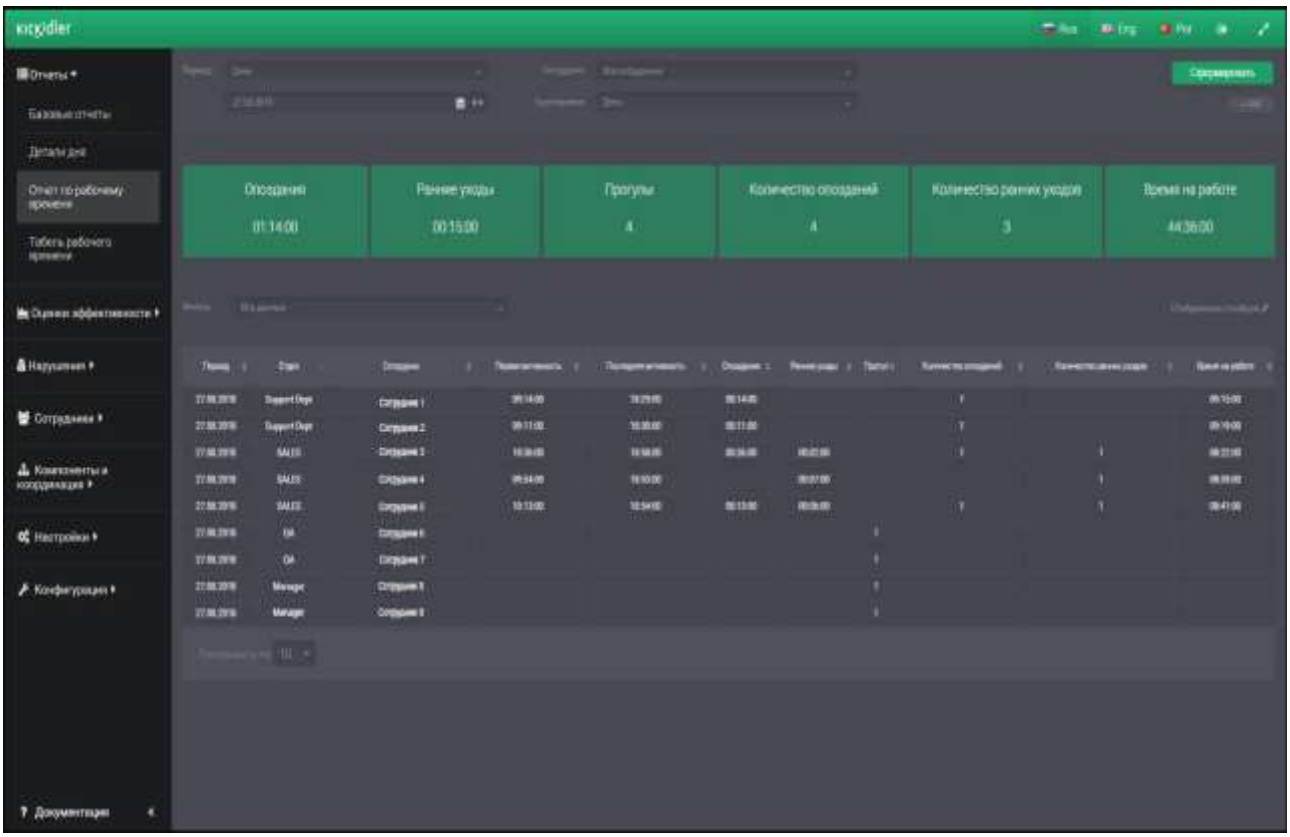


Рисунок 1.4 – Програмне забезпечення «Kickidler»

– «ActivTrak» – програма, яка призначена для моніторингу активності, аналітики продуктивності і поведінки співробітників за робочими комп'ютерами, а також для виявлення інсайдерських погроз. Як рішення щодо моніторингу співробітників, ActivTrak дає організаціям загальну картину того, як виконується робота в офісі та поза ним. Ці зображення складають усі види даних, включаючи, які додатки та програмні засоби найбільш використовуються, кількість часу, витраченого на непродуктивні завдання, і в який час доби кожен працівник є найбільш продуктивним. Це просте у використанні і установці рішення для хмарного моніторингу, яка також має локальну версію. Недоліки даного продукту заключаються у платі за використання та націленність на великі організації. Програмна реалізація зображена на рисунку 1.5.

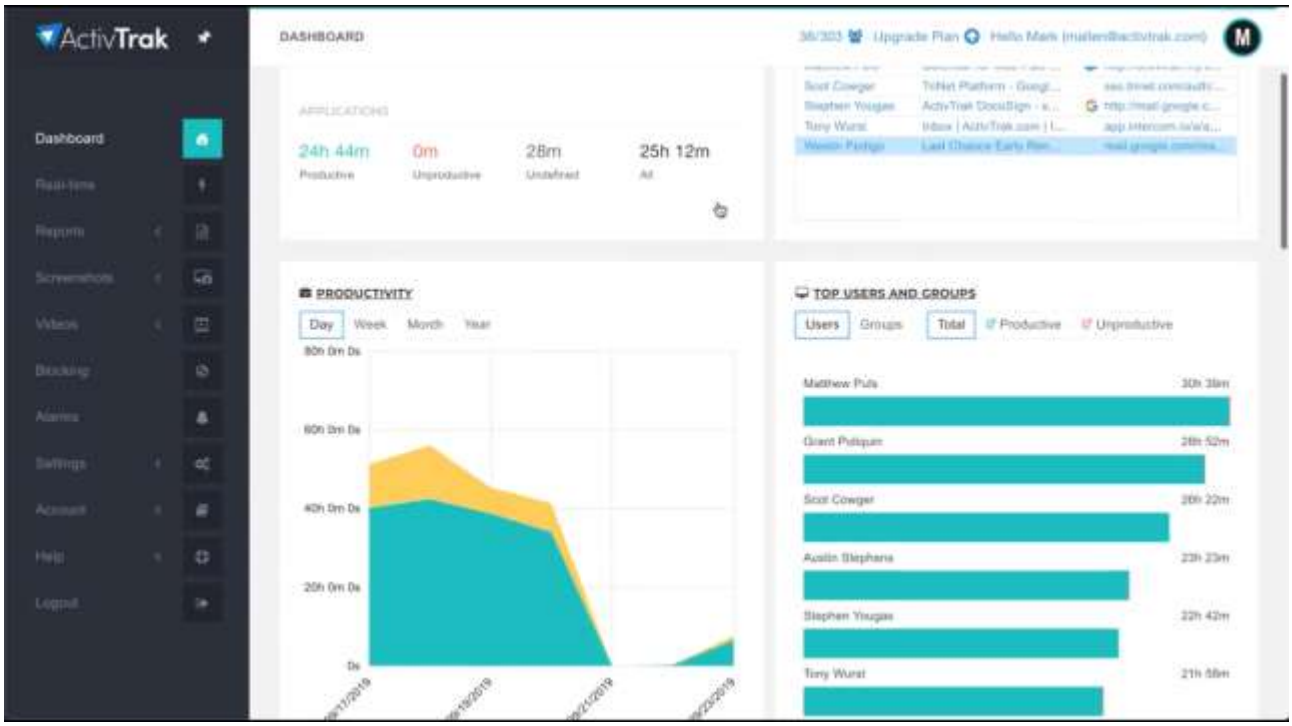


Рисунок 1.5 – Програмне забезпечення «ActivTrak»

– «Hubstaff» – це система відстеження часу з функціями захоплення екрану, моніторингу завдань, автоматичної оплати праці, розширеними звітами, відстеженням в режимі реального часу. Сервіс інтегрується з більш ніж 30 популярними інструментами. Додаток дозволяє дивитися, як робота розгортається в режимі реального часу за допомогою додаткових знімків екрана. Можливо встановити зйомку на раз, два чи три рази на 10 хвилин, поки таймер працює, а також коли вам довелося перестати працювати лише для того, щоб записати, над чим працюєте. У вашій команді може відстежуватися час, скільки та коли ви працювали над проектом чи своїм власним завдання. Hubstaff трохи відрізняється від двох попередніх програм і більше підходить для віддалених команд, невеликих копаній та фрілансерів. Програмний засіб зображений на рисунку 1.6.

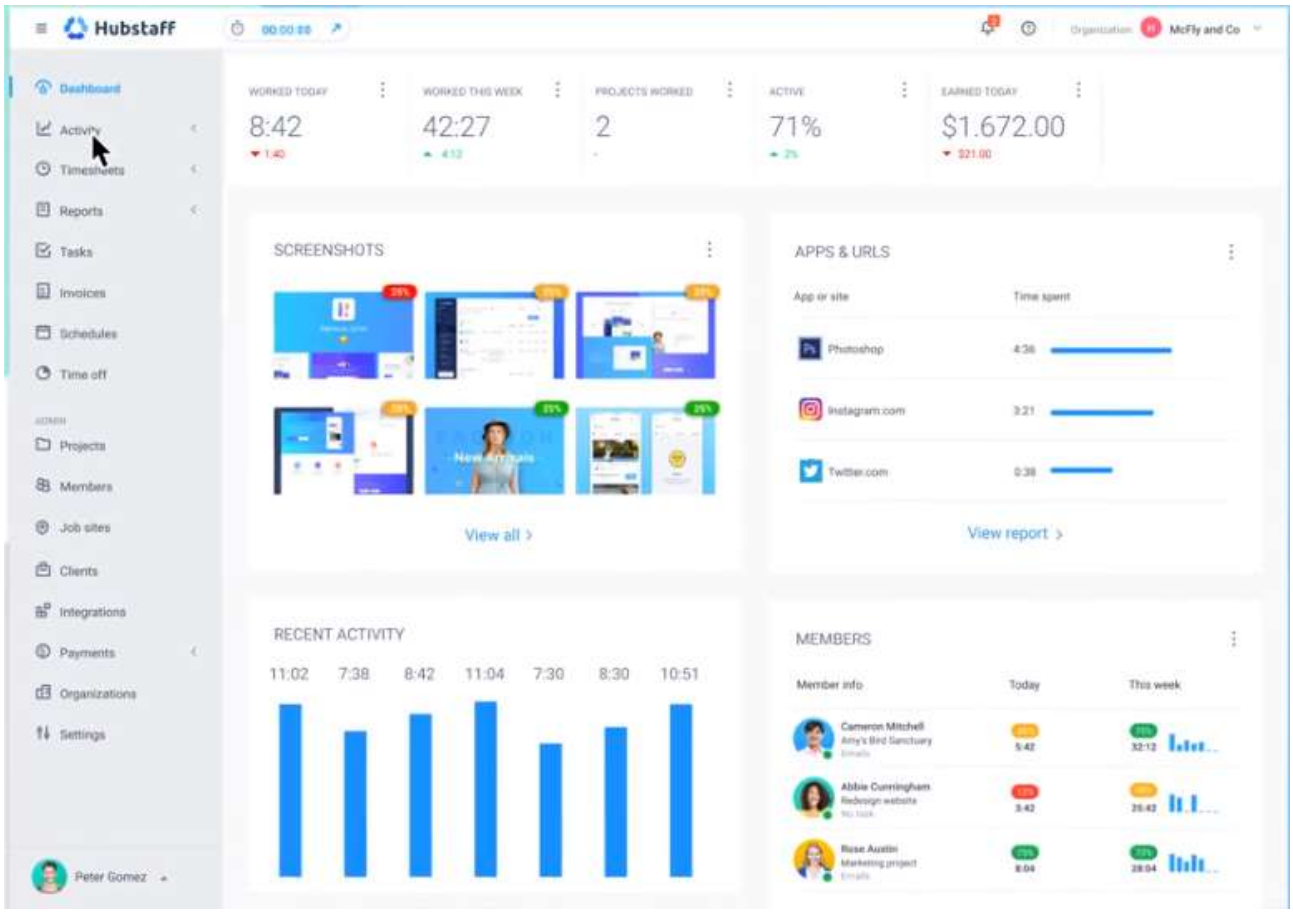


Рисунок 1.6 – Програмне забезпечення «Hubstaff»

– «DeskTime Pro» – це онлайн-сервіс з мобільними і десктопними додатками для відстеження робочого часу, відстеження прогресу в проекті, аналізу різних показників ефективності, генерації докладних звітів. Як і Hubstaff, програма орієнтована на управління віртуальними командами. Додаток має можливість дізнаватися, скільки коштує кожен проект для компанії та скільки клієнт повинен заплатити за допомогою інтегрованої функції виставлення рахунку за час. Встановіть вашу та командну погодинну ставку, і DeskTime автоматично розраховуватиме проектні витрати на основі витраченого часу на роботу. Безкоштовна версія програми DeskTime Pro надає моніторинг одного користувача з серйозними обмеженнями по функціональності. Програмний засіб зображений на рисунку 1.7.

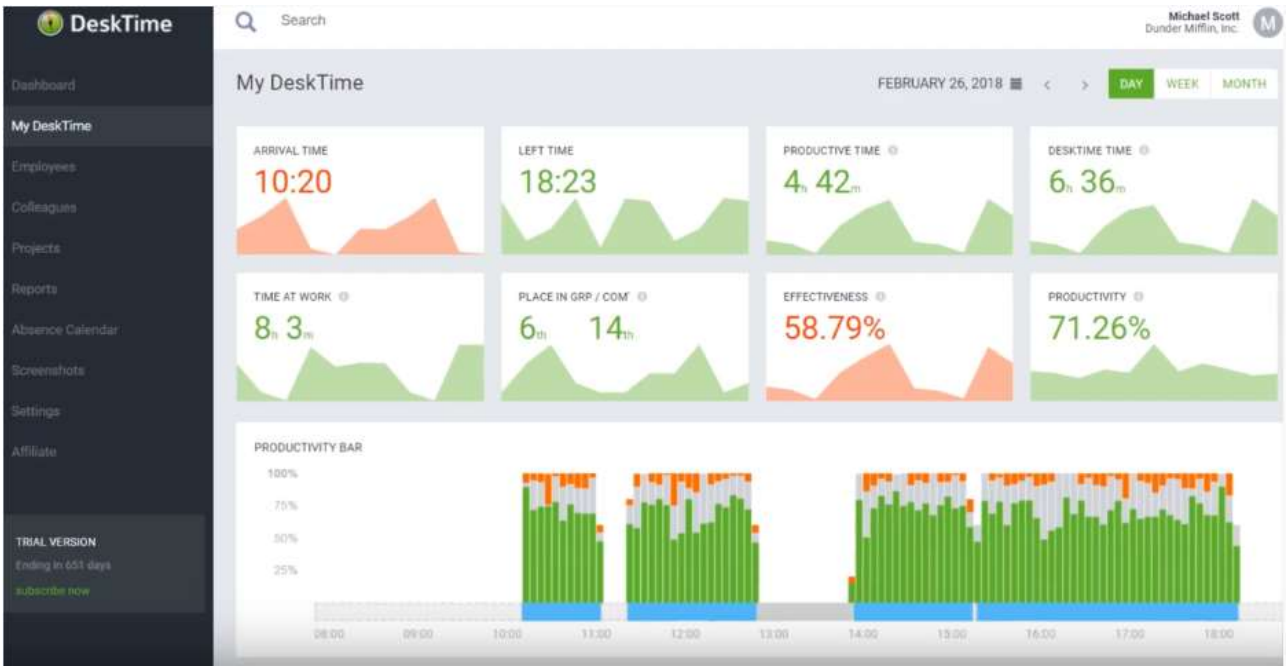


Рисунок 1.7 – Програмне забезпечення «DeskTime Pro»

Маштабними недоліками даних програмних засобів є відсутність інтеграції із соціальними мережами, таких як Telegram, через які могли приходити оповіщення на мобільний пристрій користувачів, а також те що всі наведені вище програмні засоби для моніторингу виконання завдань мають обмежений функціонал, розраховані на невеликі чи навпаки великі організації і мають плату за користування цим засобом, а також наведені вище програмні засоби використовують застарілі методи та алгоритми роботи моніторингу виконання завдань, тому на основі аналізу цих програмних засобів, можливо виділити, які непогано справляються зі своїми задачами.

У таблиці 1.3 наведено порівняльну характеристику основних конкурентів розроблюваного програмного засобу:

Таблиця 1.3 - Порівняльна характеристика основних прогрмних засобів моніторингу виконання завдань

Характеристики	Екотест	Босс Контроль	Kickidler	ActivTrak	Hubstaff
Плата за додаткові можливості і функціонал	+	+	+	+	+
Великий розмір програмного засобу	+	–	+	–	–
Термін виконання поставлених завдань	–	+	–	–	–
Орієнтація на невеликі організації, підприємства	–	+	–	+	+
Використання локального доступу до програмного засобу	+	–	+	+	+
Плата за реєстрацію програмного засобу	–	+	–	+	+
Оцінка виконаної роботи, відображення	+	–	–	–	+
Використання соцільних мереж для моніторингу	–	–	–	–	–

Розглянувши результати, можна зробити висновки, що програмний засіб Hubstaff вдало виконує свою роботу, але має свої недоліки у великому розмірі програмного засобу, у терміні виконання поставлених завдань та у використанні соціальних мереж для моніторингу виконання завдань.

Тому, актуальною задачею є створення сучасної інформаційної технологій моніторингу виконаня завдань у частині покращення швидкодії та функціональності моніторингу завдань при збережені таких характеристик як зручний інтерфейс програмних засобів, орієнтація на різний тип організацій та підприємств.

1.3 Постановка задачі

Оскільки завданням магістерської кваліфікаційної роботи є розробка інформаційної технології моніторинга виконання завдань, можна виділити такі задачі для виконання:

– Проаналізувати існуючі моделі, методи та засоби моніторингу виконання завдань.

– Розробити модель моніторингу виконання завдань

– Удосконалити методи моніторингу викоання завдань

– Розробка інформаційної технології моніторингу виконання завдань

Нехай задано вхідний вектор

$$X(x_1, x_2, x_3, x_4, x_5, x_6),$$

де

x_1 – потужність множини оповіщення. Кількість оповіщень, які створює керівник організації.

x_2 – потужність множини завдань, не менше 1.

x_3 – заданий дедлайн.

x_4 – відсоток виконання завдання.

x_5 – потужність множини облікових записів месенджерів, не менше 2.

x_6 – створення коментарів.

Тоді, задачу моніторингу виконання завдань можна подати у вигляді:

$$F(X) = Y,$$

де $Y(y_1, y_2)$ – вихідний вектор,

y_1 - множина інструментарію інформаційної технології моніторингу виконання завдань,

y_2 - множина облікових записів користувача інформаційної технології моніторингу виконання завдань.

2 РОЗРОБКА МОДЕЛІ МОНІТОРИНГУ ВИКОНАННЯ ЗАВДАНЬ

Розробка моделі моніторингу виконання завдань займає значне місце. Інформаційна технологія містить в собі велику кількість інструментарію, яка буде корисна для керівників підрозділів та виконавців, шо реалізована сучасними програмними засобами.

Стратегічне управління є інновацією в управлінні, найважливішим чинником успішного розвитку організації. Проте дуже часто можна спостерігати брак стратегічної спрямованості підприємств в діях керівників та виконавців.

Процес стратегічного керування розпочинається з обгрунтуванням мети та завдань моніторингу [17].

Процес стратегічного керування розпочинається з обгрунтуванням мети та завдань моніторингу.

Модель містить три етапи, які дають можливість розробити та застосувати ефективну модель моніторингу виконання завдань незалежно від повноти організації чи компанії.

Реалізація моделі моніторингу виконання завдань на першому етапі передбачає проведення SWOT-аналізу вхідного стану моніторингу завдань, який показує чи потрібна дана інформаційна технологія для забезпечення непреривного моніторингу в різний період часу для полібшення навчально-виховного процесу, для організації контролю і керівництва, для консольтування ведення документації, для надання порад у плануванні діяльності.

На другому етапі розробляється підлаштована під організацію чи компанію інформаційна технологія моніторингу виконання завдань, визначається зміст і шляхи її діяльності з урахуванням виявлених особливостей робочого процесу закладу, організації та розширено певні пріорітетні напрями у зв’язку з зміною статусу моніторингу виконання завдань. Відповідно до вимог часу виконання певного завдання.

На третьому етапі відбувається повна розробка моделі інформаційної технології, яка стосується певної організації і підлаштована саме під неї та відбувається вибір інформаційної технології. Так як система містить велику кількість функціоналу виконання якого не повинно перемішуватись в межах одного сервісу. Адже доменна модель процесу додатку оперує діаметрально протилежними сутностями. Існує декілька способів розділити бізнес-логіку на декілька частин:

– розбиття додатку на незалежні модулі;
– розбиття додатку на сервіси.

У першому випадку, досягається мінімальна втрата швидкодії, за рахунок однієї кодової бази. Але такий підхід несе в собі великі ризики. Модульна модель моніторингу має свій поріг масштабування, відсутність єдиного інтерфейсу обміну даними між модулями призведе до проникнення об'єктів різних модулів в інші, породжуючи цим самим циклічні залежності одних модулів від інших. Таким чином, модульна модель моніторингу обмежена і складна у дотриманні принципів SOLID.

Доцільно розбити інформаційну технологію моніторингу виконання завдань на сервіси (SOA). Структура SOA досить різноманітна і налічує декілька принципових підходів до розробки програмних додатків.

SOA - це набір принципів, що не залежать від технологій і продуктів:

– сполучуваність додатків, орієнтованих на користувачів;
– багаторазове використання бізнес-сервісів;
– незалежність від набору технологій;
– автономність (незалежні еволюція, масштабованість і розгортуваність).

Завдяки мікросервісам відбувся перехід в парадигмі SOA від віддаленого виклику методів об'єкта (CORBA) до передачі повідомлень між сервісами. Але потрібно розуміти, що в рамках SOA веб-сервіси - не просто API загального призначення, всього лише надають CRUD-доступ до бази даних через HTTP у нашому випадку саме те, що потрібно для моніторингу виконння завдань. У

якихось випадках ця реалізація може бути корисною, але заради цілісності ваших даних необхідно, щоб користувачі розуміли, що в основі API лежать бізнес-правила.

Переваги:

– незалежність набору технологій, розгортання і масштабованості сервісів;

– стандартний, простий і надійний канал зв'язку (передача тексту по HTTP через порт 80);

– оптимізований обмін повідомленнями;

– стабільна специфікація обміну повідомленнями;

– ізольованість контекстів доменів (Domain contexts).

Після створення моделі сервісного моніторингу, буде створена ще одна. Сервіси буду розбиватися на мікросервіси. В основі мікросервісної моделі моніторингу лежить концепції SOA. Призначення у неї той же, що і у шини: створити єдиний загальний корпоративну інформаційну систему з декількох спеціалізованих додатків бізнес-доменів. Головна відмінність мікросервісів і шини в тому, що шина була створена в контексті інтеграції окремих додатків, щоб вийшов єдиний корпоративний розподілений додаток. А мікросервісна модель моніторингу створювалася в контексті швидких і постійно мінливих бізнесів, які (в основному) з нуля створюють власні хмарні додатки. Характер побудови / проектування мікросервісів не вимагає глибокої інтеграції.

Модель мікросервісного моніторингу виконання завдань зображена на рисунку 2.1. Мікросервіси повинні відповідати бізнес-концепції, обмеженому контексту.

Вони повинні зберігати свій стан, бути незалежними від інших мікросервісів, і тому вони менше потребують інтеграції з іншими системами.

Тобто низька взаємозалежність і висока зв'язність привели до чудового побічного ефекту - зменшення потреби в інтеграції. Вона чудово справляється з нашою задачою для ефективнішого відправлення та отримання оповіщень на пристрої користувачів через соціальні мережі.

– проектування сервісів навколо бізнес-доменів, це може дати нам стабільні інтерфейси, високозв'язні, що мало залежать один від одного модулі коду, а також чітко визначені розмежовані контексти;

– приховування подробиць реалізації, це дозволяє сервісам розвиватися незалежно один від одного;

– незалежне розгортання, можна розгортати нову версію сервісу, не змінюючи нічого іншого;

– ізолювання збоїв - якщо один сервіс падає, інші продовжують працювати, це робить всю систему стійкою до збоїв;

– зручність моніторингу - в системі багато компонентів, тому важко встежити за всім, що в ній відбувається. Нам потрібні складні інструменти моніторингу, що дозволяють заглянути в кожен куточок системи і відстежити будь-яку ланцюжок подій.

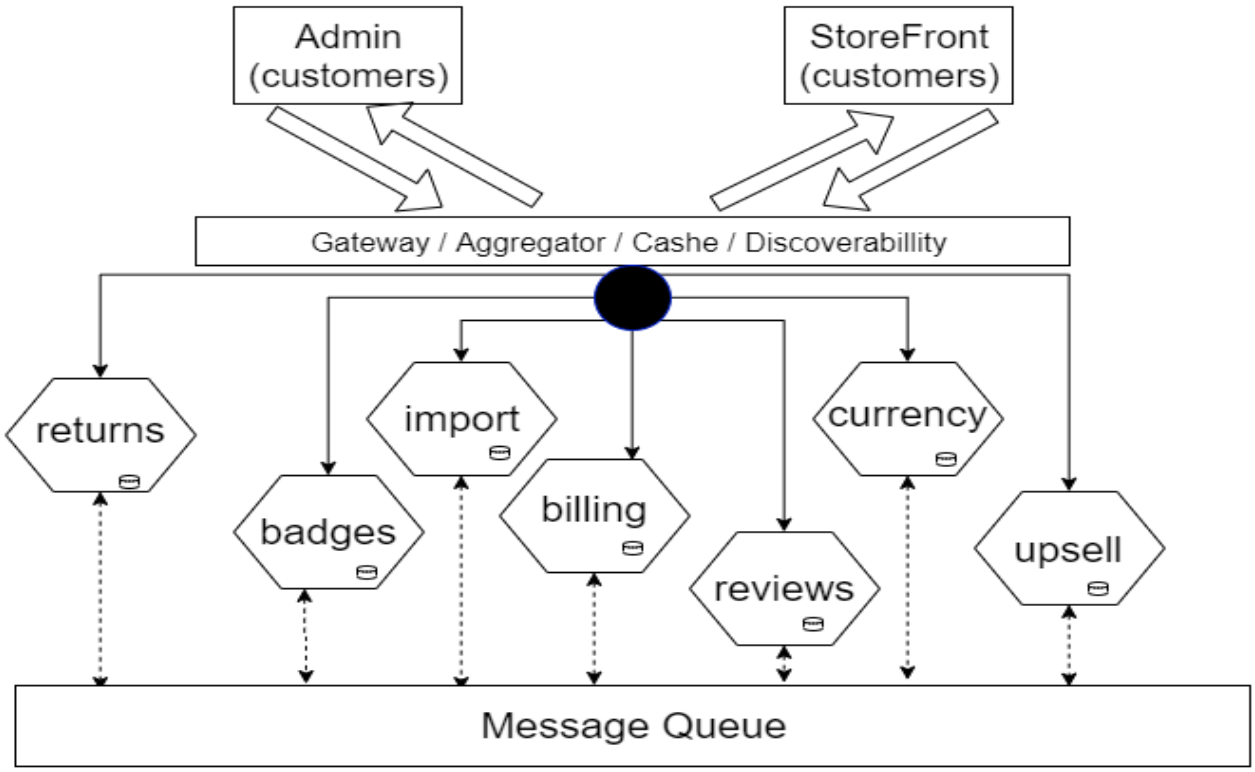


Рисунок 2.1 - Модель мікросервісного моніторингу виконання завдань

Переваги:

– незалежність набору технологій, розгортання і масштабованості сервісів;

– простота підключення і відключення сервісів;

– асинхронність обміну повідомленнями допомагає керувати навантаженням на систему;

– синхронність обміну повідомленнями допомагає керувати продуктивністю системи;

– повністю незалежні і автономні сервіси;

– бізнес-логіка зберігається тільки в сервісах;

– дозволяють компанії перетворитися в складну адаптивну систему, що складається з декількох маленьких автономних частин / команд, здатну швидко адаптуватися до змін.

Для розроблюваного додатку, головної складністю є розділення доменної логіки. Таку можливість дають одразу декілька принципів побудови моделі: черга, веб-сервіси та мікросервіси.

Веб-сервіси не передбачають використання сервісів для виконання бізнес-правил, для цього необхідно розробляти окремі сервіси, які зав'язані на сутності самих веб-сервісів, що породжує залежності і знижує можливість масштабування архітектури. Черга в синхронному режимі, при виконанні важливих бізнес-процесів, передає відповідальність за отримання відповіді на клієнта, що збільшує навантаження на клієнта, а також не дає ніяких гарантій отримання результату або його своєчасності. Мікросервіси задовольняють всім поставленим вимогам. Отже імплементація сервісної моделі буде вестись з використанням принципів мікросервісів.

Отже, було розроблено ефективну модель моніторингу виконання завдань, що дасть можливість покращити синхронність та асинхронність обміну повідомленнями, підключення і відключення до сервісів соціальних мереж.

3 УДОСКОНАЛЕННЯ АЛГОРИТМІВ МОНІТОРИНГУ ВИКОНАННЯ ЗАВДАНЬ

3.1 Узагальнений алгоритм моніторингу виконання завдань

Методи, якими вирішувалася задача моніторингу виконання завдань не надавала повноти дій та оцінювання результатів. Тому використання сучасних web-технологій дасть можливість покращити процес моніторингу виконання завдань за рахунок швидкодії технології CGI, яка використовується в складі ресурсу інтернет інтерактивних елементів на базі додатків, що забезпечують передачу потоку даних від об'єкта до об'єкта. При цьому, використовуватиметься скрипт, який запускається на виконання і обробляє отриману інформацію. Крім того, використання нової технології AMP забезпечить швидкий доступ до web-сервісу моніторингу виконання завдань [18].

Сучасні web-технології дозволять створення окремих груп для виконавців, отримання оповіщень на Email та Messenger, отримають змогу відправляти короткі спливаючі Push-повідомлення, підвищити об’єктивність оцінки результатів роботи виконавців, розширити функціонал можливостей, як для керівника організації так і для виконавців.

З урахуванням запропонованих удосконалень, алгоритм процесу моніторингу виконання завдань з використанням сучасних web-технологій включатиме додаткові кроки, а саме:

1. Після формування керівником проекту завдань та реєстрація у web-формі ввести крок «Авторизація та застосування технології AMP для швидкого доступу до web-сервісу».

2. Після призначення керівником проекту виконавця ввести крок «Виконання запиту до служби доставки оповіщення».

3. Після виконання запиту до служби доставки оповіщення додати крок «Формування скрипту CGI, який запускається на виконанні і обробляє

отриману інформацію, що забезпечує передачу потоку даних від об'єкта до об'єкта». Саме за цим кроком виконується моніторинг виконання завдання з урахуванням дедлайну через облікові записи та формується відповідний висновок про повноту виконання завдання у визначений період часу.

UML – діаграма узагальненого алгоритму моніторингу виконання завдань проекту представлена на рисунку 3.1.

Рисунок 3.1 - UML – діаграма узагальненого алгоритму моніторингу виконання завдань проекту

3.2 Розробка алгоритму формування завдань керівником проекту

Удосконалення буде відбуватися за рахунок сучасних програмних засобів, а саме за рахунок системи обміну оповіщеннями на міжсерверному зв’язку.

Задіяна для цього буде черга повідомлень і прихід оповіщень на різні служби, де будуть зараєстровані керівники та виконавці.

Черга повідомлень покращує масштабованість і підсилює ізольованість додатків. Їм не потрібно знати, де знаходяться інші додатки, скільки їх і навіть що вони собою представляють [19].

Однак всі ці додатки повинні використовувати одну мову обміну повідомленнями зображено на рисунку 3.2, тобто заздалегідь певний текстовий формат представлення даних. Черга повідомлень використовує в якості компонента інфраструктури програмний брокер повідомлень (RabbitMQ, Beanstalkd, Kafka і т. д.).

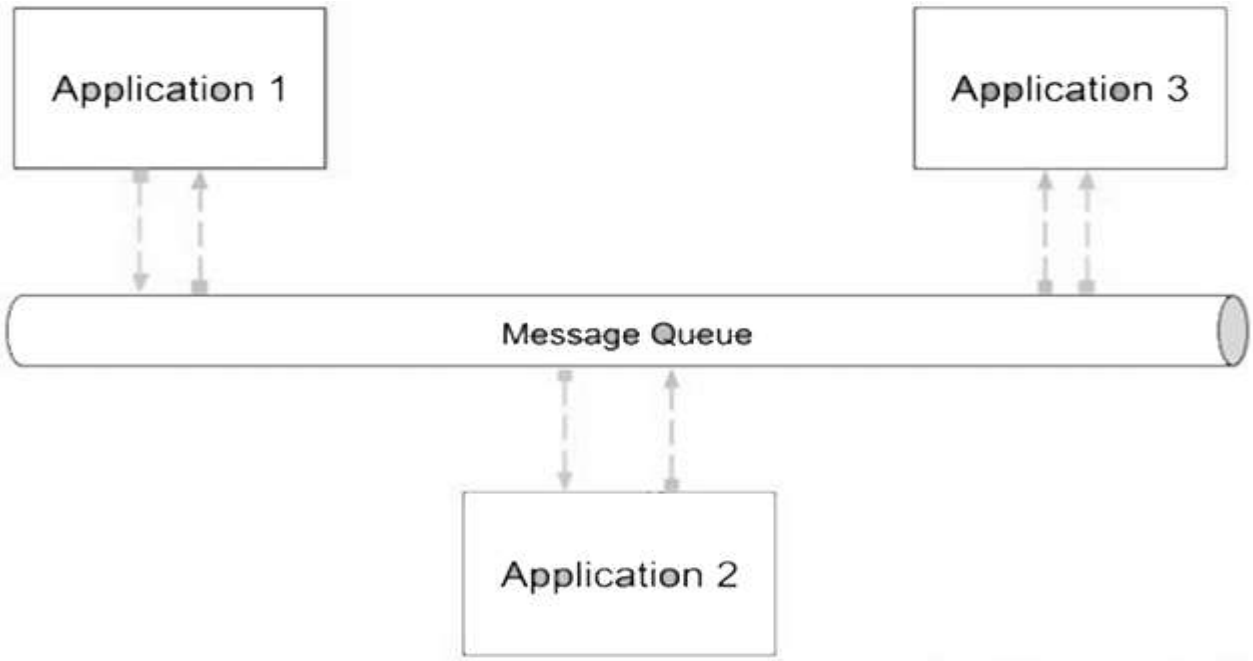


Рисунок 3.2 - Структура сервісів з чергою повідомлень

Для реалізації зв'язку між додатками можна по-різному налаштувати чергу:

Запит/Відповідь:

– Користувач посилає в чергу повідомлення, включаючи посилання на «розмову» («conversation» reference). Повідомлення приходить на спеціальний вузол, який відповідає відправнику іншим повідомленням, де міститься посилання на ту же розмову, так що одержувач знає, на яку розмову посилається повідомлення, і може продовжувати діяти. Це корисно для бізнес-процесів середньої і великої тривалості (ланцюжків подій, sagas).

Публікація/Підписка:

– За списками - черга підтримує списки опублікованих завдань і виконавців, які їх отримали. Коли черга отримує повідомлення для якогось завдання, то поміщає його у відповідний список. Повідомлення зіставляється з завданням за типом повідомлення або по заздалегідь визначеному набору критеріїв, включаючи і зміст повідомлення.

– На основі мовлення - коли черга отримує повідомлення, вона транслює його всім вузлам, прослуховуючих чергу. Вузли повинні самі фільтрувати дані і обробляти тільки повідомлення з заданим завданням.

Отже, при відправці повідомлень на вузли, які були створені, завдання поміщається у правильний список для кожного користувача додатку, який зареєстрований у ньому і доступний тільки йому, якщо повідомлення доставляється групі користувачів, тобто списку користувачів, які мають виконувати одне і теж завдання, трансляція відбувається усім вузлам додатку певній групі.

Кожен вузол – це окремий компонент інформаційної технології, який взаємодіє на своєму рівні доступу і незалежний від інших вузлів, крім списку груп.

UML – діаграма алгоритму формуваня завдань керівником проекту зображена на рисунку 3.3.

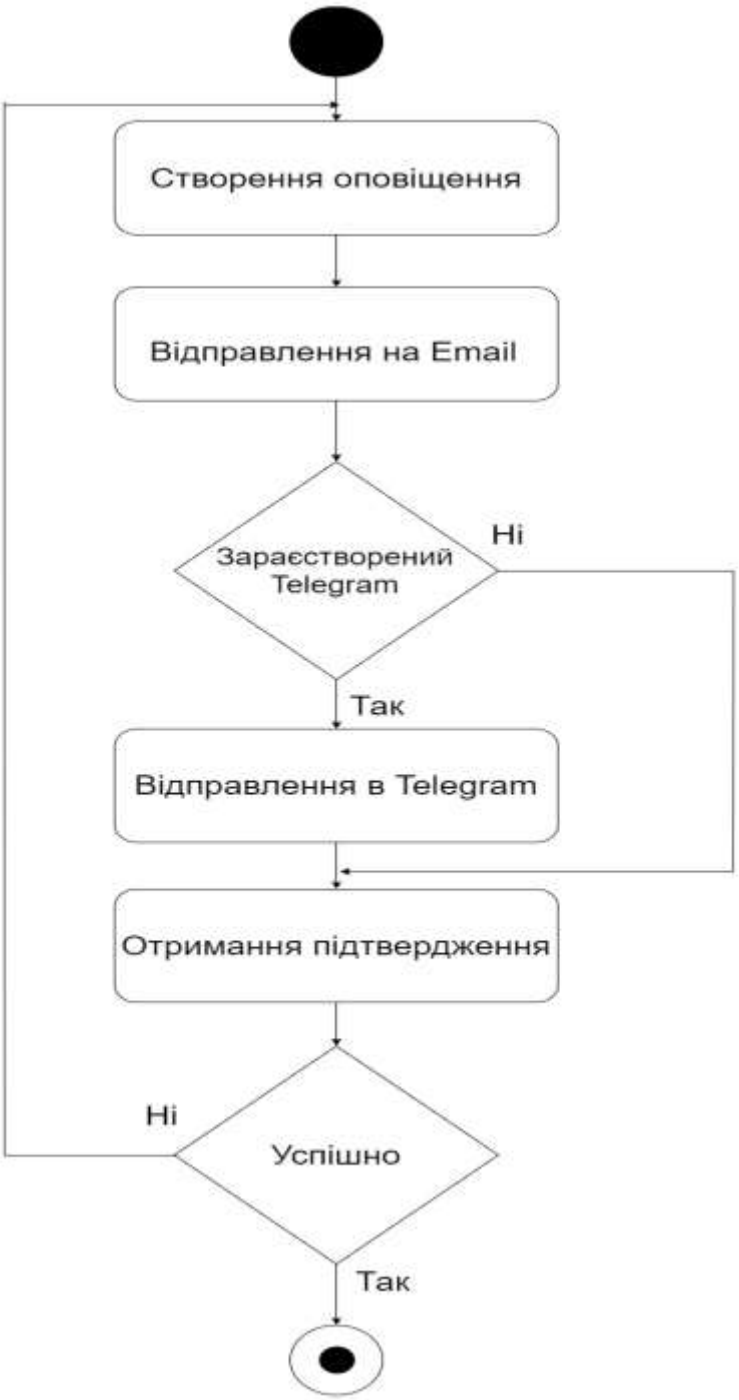


Рисунок 3.3 – UML – діаграма алгоритму формуваня завдань керівником проекту

3.3 Розробка алгоритму моніторингу виконання завдань з урахуванням дедлайну

Розробка алгоритму моніторингу виконання завдань з урахуванням дедлайну відбувається за рахунок доданого нового функціоналу можливостей для керівника проекту, тим самим він зможе визначати повноту виконання завдання, додавати коментарі до ще невиконаних завдань, які також враховуються на оцінку. Керівнику проекта доступна можливість вибирати відсоткове співвідношення виконаного завдання, максимальне число 100%.

Алгоритм моніторигу виконання завдань з урахуванням дедлайну реалізований програмним засобом, що у функціоналі передбачає формування оповіщень про задачі, які слід виконати відповідно до визначених дедлайнів та формату представлення результатів роботи.

Потужність множини завдань може бути будь-яка, все залежить від керівника проекту та кількості завдань, які має виконати виконавець проекту, визначений час нагадування про дедлайн виконання завдання задається керівником від 1 годин до часу визначеного керівником, потужність множини облікових записів месенджерів може змінюватися [20].

Саме за цим кроком виконується моніторинг виконання завдання з урахуванням дедлайну через облікові записи та формується відповідний висновок про повноту виконання завдання у визначений період часу.

UML- діаграма алгоритму моніторингу виконання завдань з урахуванням дедлайну зображена на рисунку 3.4.

Рисунок 3.4 – UML- діаграма алгоритму моніторингу виконання завдань з урахуванням дедлайну

Отже, розроблений алгоритм моніторингу виконання завдань з урахуванням дедлайну завдань, що у функціоналі повинен передбачати формування оповіщень про задачі, які слід виконати відповідно до визначених дедлайнів та формату представлення результатів роботи.

4 РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ МОНІТОРИНГУ ВИКОНАННЯ ЗАВДАНЬ

4.1 Розробка структури інформаційної технології моніторингу виконання завдань

Інформаційна технологія моніторингу виконання завдань це складна система, яка включатиме у себе два типи користувачів, щоб задовільно використовувати всі можливості та функції моніторингу виконання завдань.

– керівник організації;
– виконавці у певному підрозділі;

Кожна група керівників чи виконавців має свої права та обов’язки при заданні завдання, тобто кожен має різний рівень доступу до системи. Зрозуміло, що виконавець не може дати завдання керівнику організації і це недоцільно, тому були створені спеціальні міри для запобіганню такої ситуації.

Інформаційна технологія дозволяє реалізувати, що керівником організації або певного підрозділу може бути декілька людей.

Керівник організації або певного підрозділу має певні права доступу до інформаційної технології моніторингу виконання завдань. Такі як:

– давати певні задачі виконавцям;
– переглядати, видаляти, змінювати та редагувати оповіщення;
– створювати нові оповіщення;
– додавати час на виконання певного завдання;
– видаляти користувачів та додавати користувачів у список друзів;
– присилати запрошення користувачам, які ще не зареєстровані у системі;
– оцінювати роботу кожного виконавця;
– додавати коментарі до завдань;
– визначати ступінь виконаного завдання.

Виконавець має свої права доступу, наприклад, він може:

– видаляти користувачів та додавати користувачів у список друзів;

– присилати запрошення користувачам, які ще не зареєстровані у системі;

– отримувати оповіщення від керівника.

Діаграма прецедентів інформаційної технології моніторингу виконання завдань зображена на рисунку 4.1.

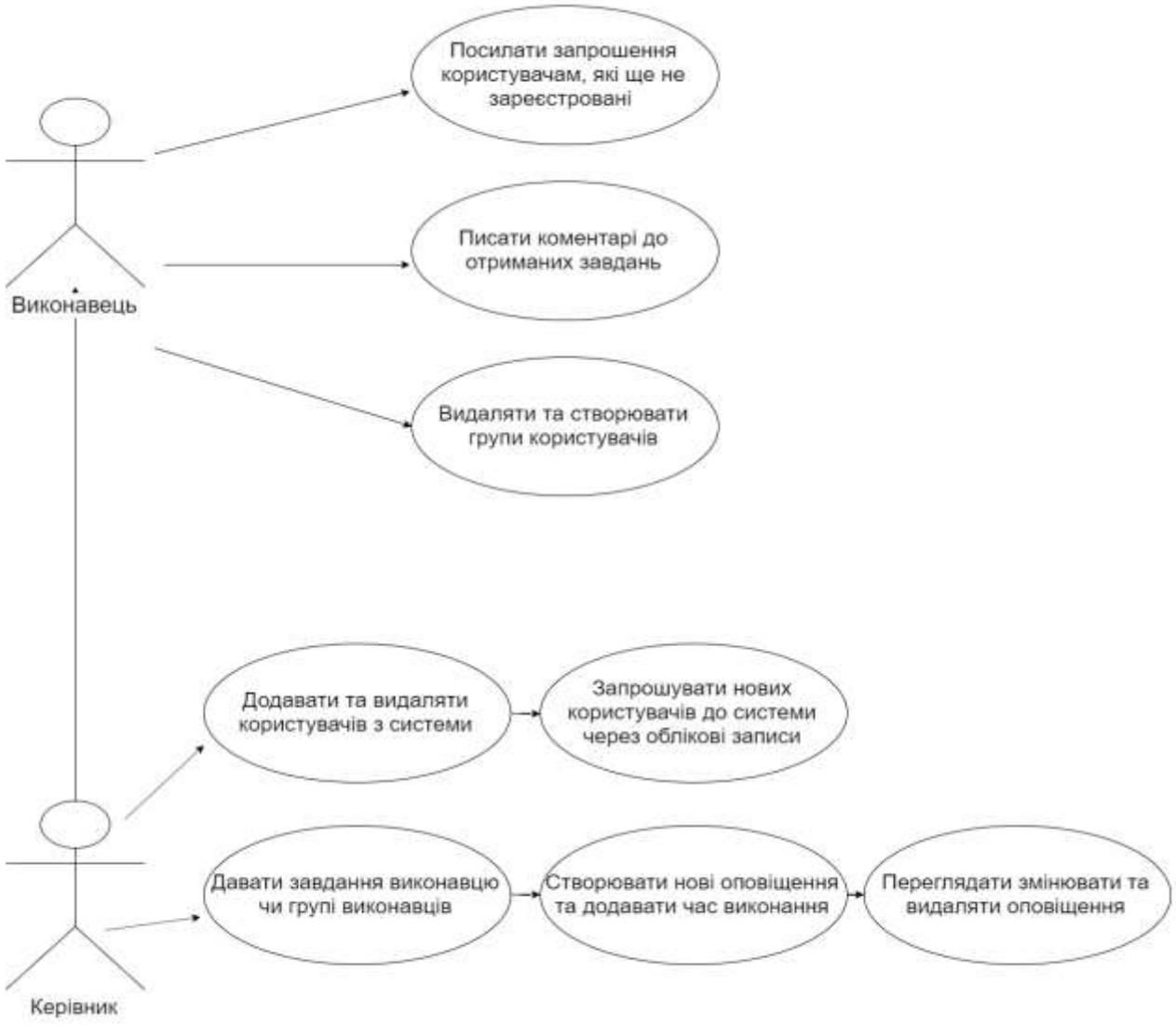


Рисунок 4.1 – UML-діаграма прецедентів інформаційної технології моніторингу виконання завдань

Для повної взаємодії та функціонування інформаційної технології моніторингу виконання завдань мають бути присутні всі вузли системи:

– вузол інтерфейсу користувачів;
– вузол бази даних;
– вузол оповіщень завдань;
– вузол облікових записивів Telegram.

Розроблена загальна структурна схема взаємодії вузлів інформаційної технології моніторингу виконання завдань зображена на рисунку 4.2.

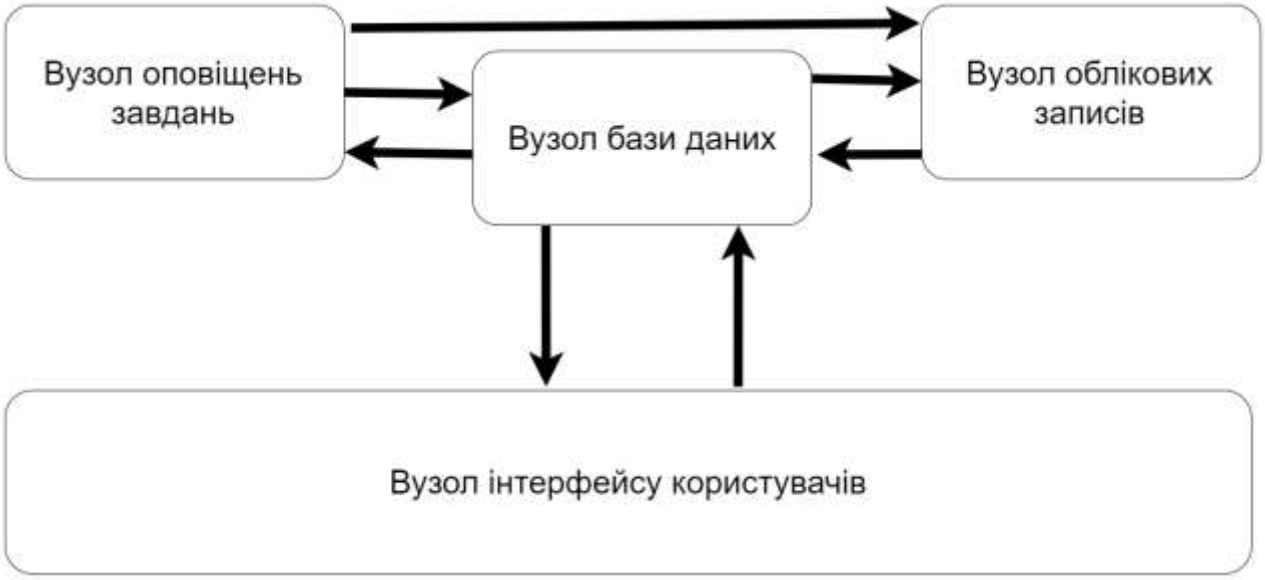


Рисунок 4.2 – Схема взаємодії вузлів інформаційної технології моніторингу виконання завдань

Отже, взаємодія вузлів інформаційної технології забезпечує виконання задачі моніторингу в цілому.

4.2 Вибір середовища програмування та мови програмування

Розділ заключає у собі властивості мов програмування таких як: C++, C#, PHP та Java для реалізації інформаційної технології моніторингу виконання завдань.

C# працює з використанням .NET Framework, яка надає можливість працювати з великою кількістю бібліотек, що містять класи, які використовуються для виконання загальних завдань, таких як робота з масивами, відображення вікон або редагування файлів. На відміну від інших мов програмування немає необхідності у виборі декількох бібліотек для реалізації невеликого завдання [21].

C# є об’єктно-орієнтованою мовою програмування та підтримує деякі функції, які зазвичай зустрічаються у функціональних мовах, такі як делегати та анонімні класи.

C# найкращий вибір для прикладних програм під управлінням операційної системи Windows. При використані .NET Core, C# є універсальною мовою, яка надає можливість запускатися додатку на будь-якій операційній системі.

C# – об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (при фірмі Microsoft).

Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників — мов C++, Delphi, Модула і Smalltalk – C#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів (на відміну від C++).

C# розроблялась як мова програмування прикладного рівня для CLR і тому вона залежить, перш за все, від можливостей самої CLR. Це стосується, перш за все, системи типів C# [22-23]. Присутність або відсутність тих або інших виразних особливостей мови диктується тим, чи може конкретна мовна особливість бути трансльована у відповідні конструкції CLR. Так, з розвитком CLR від версії 1.1 до 2.0 значно збагатився і сам C#. CLR надає C#, як і всім іншим .NET-орієнтованим мовам, багато можливостей, яких позбавлені «класичні» мови програмування. Наприклад, збірка сміття не реалізована в самому C#, а проводиться CLR для програм, написаних на C# точно так, як і це робиться для програм на VB.NET, J# тощо.

Переваги мови C# у порівнянні з C++:

1. Класи можуть бути визначені всередині класів
2. Можливості відображення
3. Не потрібно турбуватися про заголовні файли “.h”
4. Визначення класів і функцій можуть бути зроблені в будь-якому порядку
5. Автоматичний збір сміття
6. Немає глобальних функцій або змінних, все належить до класу
7. Всі змінні ініціалізуються значеннями за замовчуванням, перш ніж використовувати (це автоматично, за замовчуванням, але може бути зроблено вручну, використовуючи статичні конструктори)
8. Не можна використовувати не логічні змінні, як умови.
9. Більш чисте управління подіями (за допомогою делегатів)

Переваги C# в порівнянні з Java:

1. Має більш примітивні типи (типи значень)
2. Спрощена багатопоточність
3. Перевантаження операторів. Це може зробити розробку трохи складнішою, але вони не є обов’язковими, але іноді є дуже корисними.

Порівняння мов C++, Java і C# зображено в таблиці 4.1.

Враховуючи всі переваги і недоліки, була вибрана мова C#.

Таблиця 4.1 – Порівняння мов програмування C++, Java і C#

Ознака порівняння	C++	Java	C#
ООП мова	Так	Так	Так
Наявність контейнерів	Ні	Так	Так
Перевантаження операторів	Так	Ні	Так
Автоматичний збір сміття	Ні	Так	Так
Висока продуктивність коду	Так	Ні	Ні
Зручність налагодження	Ні	Так	Так
Безкоштовне використання	Так	Так	Ні

Для реалізації інтерфейсної частини порівнюються такі фреймворки та бібліотеки, як React, Vue та Angular.

React — відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту веб-сторінки, з якими стикаються в розробці односторінкових застосунків. React дозволяє розробникам створювати великі веб-застосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим. React обробляє тільки користувацький інтерфейс у застосунках [24].

Vue.js — JavaScript-фреймворк що використовує шаблон MVVM для створення інтерфейсів користувача на основі моделей даних, через реактивне зв'язування даних. 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.

AngularJS — JavaScript-фреймворк з відкритим програмним кодом, який розробляє Google. Призначений для розробки односторінкових додатків, що складаються з одної HTML сторінки з CSS і JavaScript. Його мета — розширення браузерних застосунків на основі шаблону Модель-вид-контролер (MVC), а також спрощення їх тестування та розробки. Фреймворк працює зі сторінкою HTML [25], що містить додаткові атрибути і пов'язує області вводу або виводу сторінки з моделлю, яка є звичайними змінними JavaScript. Значення цих змінних задаються вручну або отримуються зі статичних або динамічних JSON-даних.

Порівняння фреймворків Angular, Vue та бібліотеки React зображено в Таблиці 4.2.

Таблиця 4.2 – Порівняння фреймворків Vue, Angular і біоліотеки React.

Ознака порівняння	Vue	Angular	React
Рендеринг	Так	Так	Так
Наявність архітектури компонентів	Так	Так	Ні
Зворотня сумістність	Ні	Ні	Так
Висока продуктивність коду	Так	Так	Так
Зручність налагодження	Так	Ні	Так
Легковагі	Так	НІ	Так

Проаналізувавши мови програмування та фреймворки було обрано технологію ASP NET C# для реалізації програмної частини і бібліотеку React для реалізації інтерфейсу користувача.

4.3 Розробка сервісу авторизації інформаційної технології моніторингу виконання завдань

Інформаційна технологія моніторингу виконання завдань передбачає певний обов’язковий інструментарій. Такий як авторизація та автенфікація, безпосередньо виконання моніторингу, обробка завдань користувачів, а також обробка персональних даних даних користувачів. Для цього в мікросервісній моделі передбачені сервіси спеціального призначення такі як:

- сервіс авторизації;
- API Gateway;
- CDN сервіси.

Для забезпечення роботи інформаційної технології моніторингу виконання завдань належним чином, були виділені такі сервіси:

- API Gateway;
- сервіс завантаження файлів;
- сервіс авторизації;
- сервіс адміністрування;
- сервіс нотифікації користувачів;
- сервіс користувацьких даних;
- сервіс зберігання сесій (SSO);
- сервіс створення завдань;
- CDN сервіси;
- сервіс пошуку;
- системні сервіси для обміну повідомленнями.

На сьогоднішній день, існує багато способів авторизувати користувача. Аунтенфікація на основі сесій [26]. Протокол HTTP не відстежує стан (stateless), і, якщо ми автентифікуєм користувача за допомогою імені та пароля, інформаційна технологія моніторингу не знатиме, чи це людина, з попереднього запиту. Нам доведеться автентифікувати користувача ще раз. Кожен запит HTTP не зберігає історію про те, що відбувалося до цього, він

лише передає запит. Так що, якщо вам потрібні особисті дані, доведеться знову логінитися, щоб додаток знав, що це точно ви. Щоб позбутися цієї незручності, візьмемо автентифікацію на основі сесій / кук, за допомогою яких реалізували відстеження станів (statefull).

Це означає, що автентифікований запис або сесія повинні зберігатися і на сервері, і на клієнті. Сервер повинен відслідковувати активні сесії в базі даних або пам'яті, а на фронтенді створюється кука, в якій зберігається ідентифікатор сесії зображено на рисунку 4.3. Така автентифікація на основі куки, найпоширеніший і широко відомий метод, який використовується вже давно.

Процедура автентифікації на основі сесій:

– користувач вводить в браузері своє ім'я та пароль, після чого клієнтська програма відправляє на сервер запит;

– сервер перевіряє користувача, автентифікує його, шле з додатком унікальний користувальницький токен (зберігши його в пам'яті або базі даних);

– клієнтську програму зберігає токени в куках і відправляє їх при кожному наступному запиті;

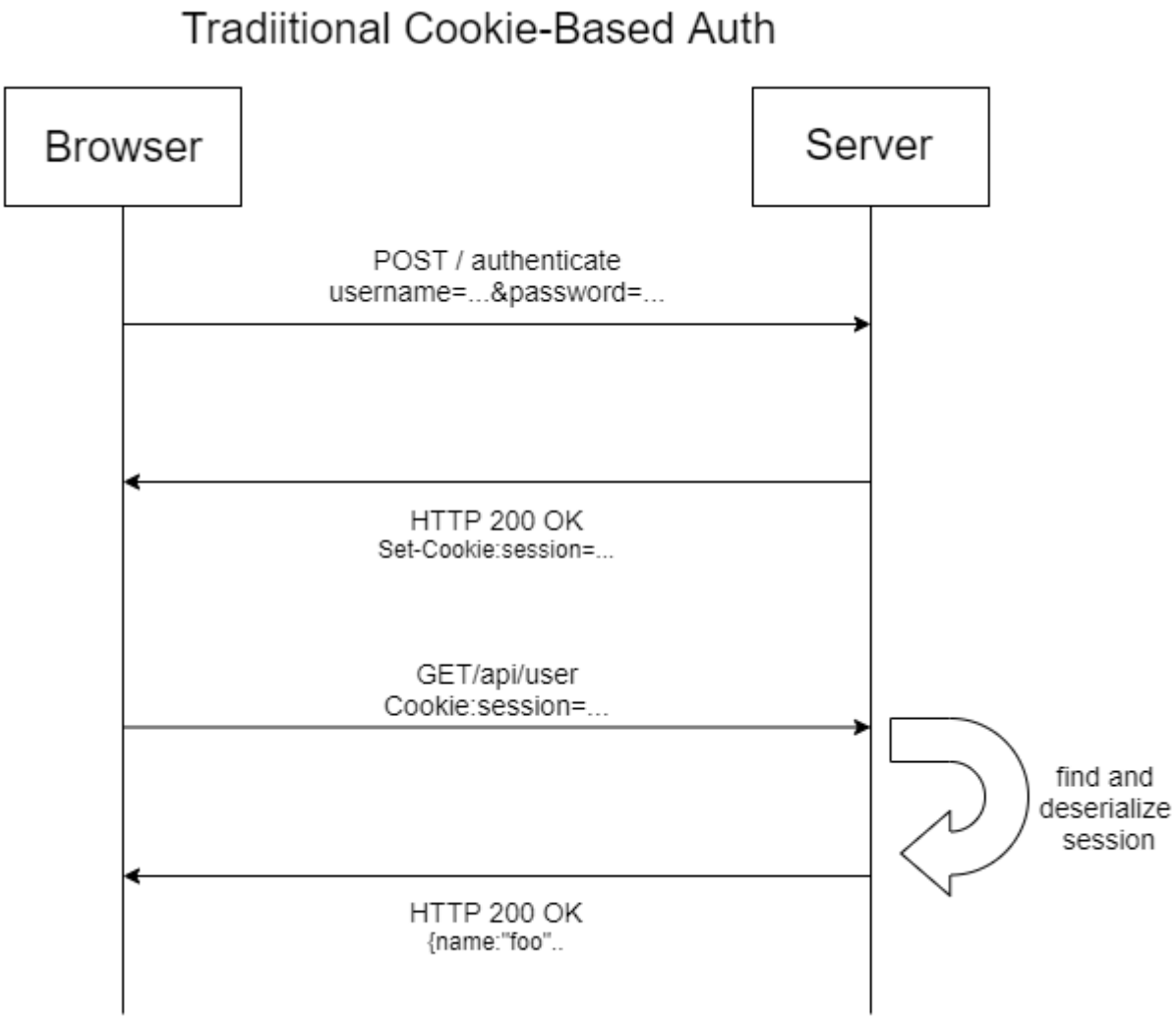
– дайджест-автентифікація підтримується всіма популярними серверами й браузерами;

– сервер отримує кожен запит, що вимагає автентифікації, за допомогою токена автентифікує користувача і повертає запитані дані клієнтського додатку;

– коли користувач виходить, клієнтську програму видаляє його токен, тому всі наступні запити від цього клієнта стають нерозпізнані;

– автентифікація з відкритим ключем використовується як захищений механізм автентифікації в таких протоколах як SSL;

– користувач запрошує сторінку з сервера, браузер автоматично відправляє cookies з ідентифікатором сесії серверу. Сервер перевіряє ідентифікатор у своїй базі ідентифікаторів і, за наявності в базі такого ідентифікатора, «пізнає» користувача.



Риснок 4.3. – Модель авторизації за допомогою кук

У цього методу кілька недоліків. При кожній автентифікації користувача сервер повинен створювати у себе запис. Зазвичай він зберігається в пам'яті, і при великій кількості користувачів є ймовірність занадто високого навантаження на сервер.

Оскільки сесії зберігаються в пам'яті, масштабувати не так просто. Якщо ви багаторазово реплікуєте сервер, то на все нові сервери доведеться реплікувати і всі призначені для користувача сесії. Це ускладнює масштабування.

Автентифікація на основі токенів [27]. Автентифікація на основі токенів в останні роки стала дуже популярна через поширення односторінкових додатків, веб-API і інтернету речей. Найчастіше в якості токенов використовуються Json

Web Tokens (JWT) зображено на рисунку 4.4. Хоча реалізації бувають різні, але токени JWT перетворилися в стандарт де-факто. При автентифікації на основі токенів стан не відслідковується. Інформація про користувача не зберігається на сервері або в сесії і навіть не зберігається сам JWT.

Процедура автентифікації на основі токенів:

– користувач вводить ім'я і пароль;

– за допомогою програми токен отримає ключ для входу в систему;

– сервер перевіряє їх і повертає токен (JWT), який може містити метадані начебто user_id, дозволів;

– токени з підключенням повинні бути фізично пов'язані з комп'ютером, на якому користувач проходить перевірку автентичності;

– токен зберігається на стороні клієнта, найчастіше в локальному сховищі, але може лежати і в сховищі сесій або кук;

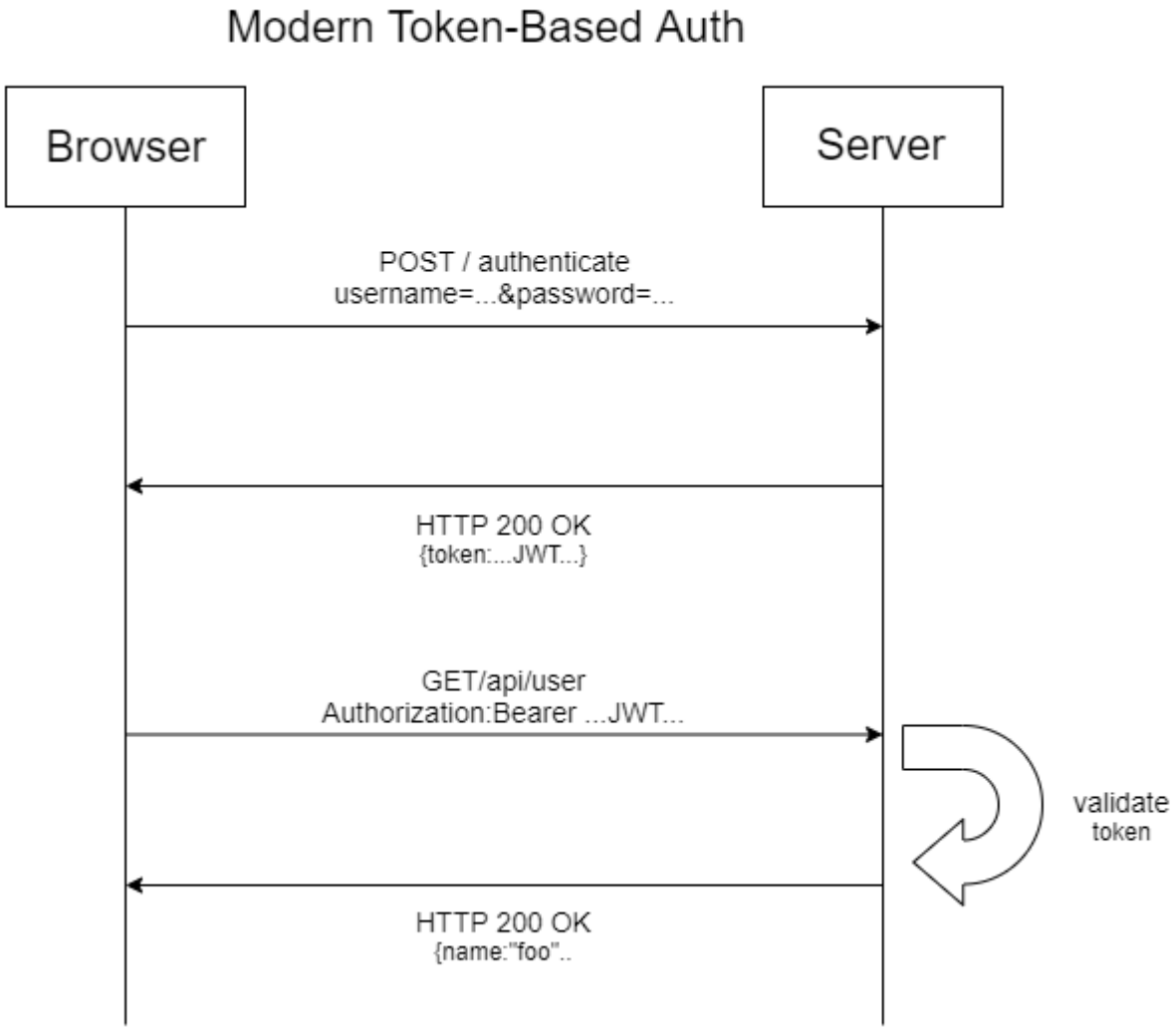
– синхронізовані за часом одноразові паролі постійно змінюються у встановлений час, наприклад, раз на хвилину. Для цього повинна існувати синхронізація між токеном клієнта та сервером автентифікації;

– наступні запити до сервера зазвичай містять цей токен в якості додаткового заголовка авторизації в вигляді Bearer {JWT}. Ще токен може пересилатися в тілі POST-запиту і навіть як параметр запиту;

– сервер розшифровує JWT, якщо токен вірний, сервер обробляє запит;

– з точки зору операційної системи комп'ютера, такий токен є підключеним через USB смарт-карток рідером з однією незмінною смарт-карткою всередині;

– коли користувач виходить з системи, токен на клієнтській стороні знищується, з сервером взаємодіяти не потрібно.



Риснок 4.4. – Модель авторизації за допомогою токенів

У методу є ряд переваг. Головна перевага: оскільки метод ніяк не оперує станами, серверу не потрібно зберігати записи призначені для користувача. Кожен токен самодостатній, містить всі необхідні для перевірки дані, а також передає затребувану призначену для користувача інформацію. Тому токени не ускладнюють масштабування.

У куках відбувається збереження ID сесії користувачів, а JWT дозволяє зберігати метадані будь-якого типу, якщо це коректний JSON формат.

При використанні кук, бекенд повинен виконувати пошук за традиційною SQL-базі або NoSQL-альтернативі, і обмін даними напевно триває довше, ніж розшифровка токена.

Крім того, раз ви можете зберігати всередині JWT додаткові дані на зразок користувальницьких дозволів, то можете заощадити і додаткові звернення пошукові запити на отримання та обробку даних. У використання кук на мобільних платформах має багато обмежень і особливостей, а токени сильно простіше реалізувати на iOS і Android. До того ж токени простіше реалізувати для додатків і сервісів інтернету речей, в яких не передбачено зберігання кук.

Безпарольна автентифікація [28]. Впроваджено переконання, що паролі - абсолютне джерело захисту наших акаунтів. Але якщо вивчити питання глибше, то з'ясується, що безпарольна автентифікація може бути не просто безпечною, але і безпечніше традиційного входу по імені і паролю.

Безпарольна автентифікація - це спосіб конфігурації процедури входу і автентифікації користувачів без введення паролів. Ідея така: замість введення пошти / імені та пароля користувачі вводять тільки свою пошту.

Ваша програма відправляє на цю адресу одноразову посилання, користувач по ній клікає і автоматично входить на ваш сайт / в додаток. При безпарольній автентифікації додаток вважає, що в ваш ящик прийшов лист з посиланням, якщо ви написали свій, а не чужий адрес.

Є схожий метод, при якому замість одноразової посилання по SMS відправляється код або одноразовий пароль. Але тоді доведеться об'єднати ваше додаток з SMS-сервісом на кшталт twilio (і сервіс не безкоштовний). Код або одноразовий пароль теж можна відправляти поштою. І ще один, менш (поки) популярний (і доступний тільки на пристроях Apple) метод безпарольної автентифікації: використовувати Touch/Face ID для автентифікації за відбитками пальців або лицем.

Якщо ви користуєтеся Slack, то вже могли зіткнутися з безпарольною автентифікацією.

Але в даному методі є деякі нюанси, якщо хтось отримає доступ до даних користувача, він отримає і доступ до додатків і сайтів. Але це не головна задача - турбуватися про безпеку поштових акаунтів користувачів. Крім того, якщо

хтось отримає доступ до чужої пошти, то зможе перехопити акаунти в додатках з безпарольною автентифікацією, скориставшись функцією відновлення пароля. Але ми нічого не можемо вдіяти з поштою користувачів.

Це далеко не весь список, також є інші популярні методи на кшталт авторизації в соціальних мережах (SocialLogin) а також двофакторна авторизація (2FA).

Принципи реалізації наведені нижче. OAuth2 це фреймворк для авторизації, який дозволяє додаткам отримувати обмежений доступ до акаунтів користувачів через HTTP, таким як Fackebook, GitHub і DigitalOcean.

Він працює шляхом делегування ідентифікації користувача сервісу, на якому розміщена обліковий запис користувача і авторизаційні сторонні додатки мають доступ до облікового запису користувача. OAuth2 надає авторизовані потоки для десктопних, веб додатків і мобільних пристроїв.

Ролі авторизації OAuth:

– власник ресурсу;

– клієнт;

– сервер ресурсу;

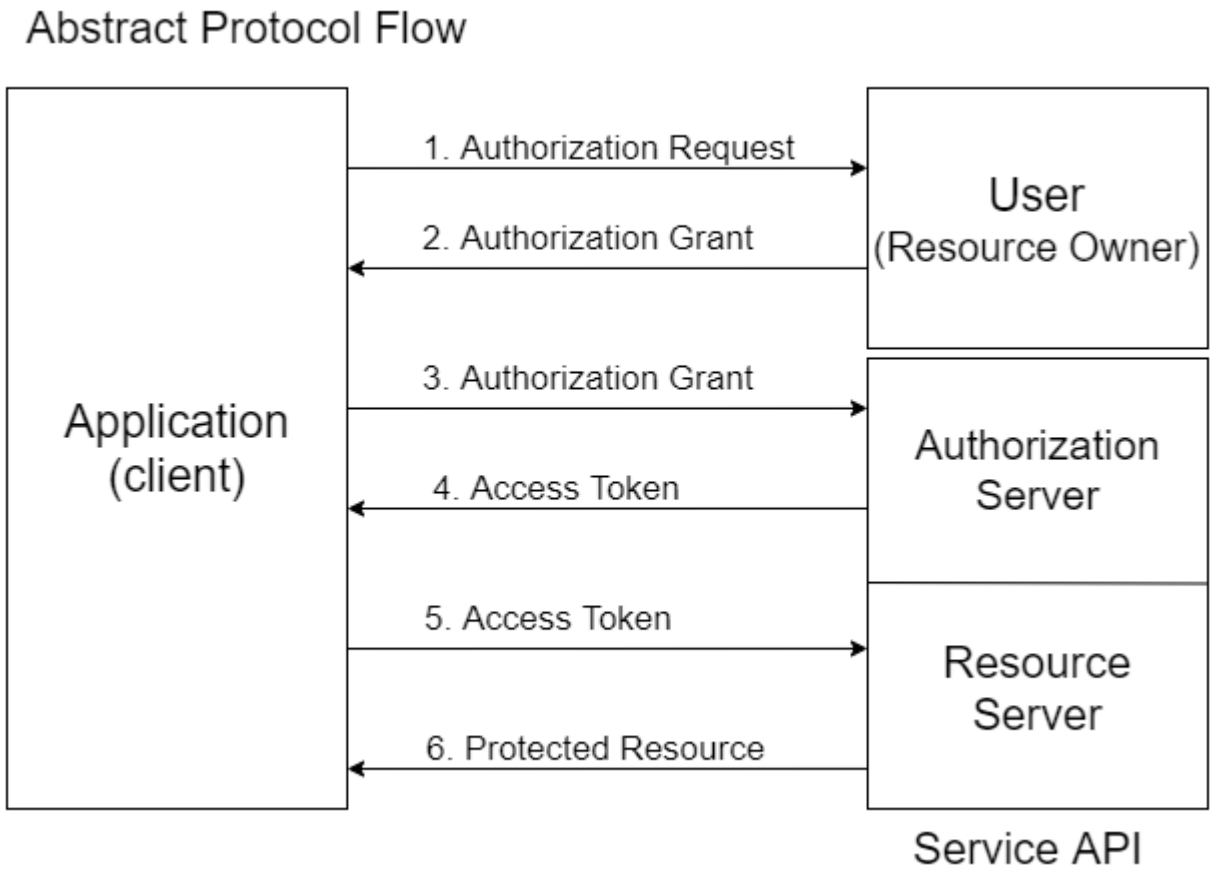
– сервер авторизації.

Власник ресурсу: кристувач. Власник ресурсу це користувач, який використовує додаток для доступу до аккаунту. Доступ до програми до аккаунту користувача обмежений областю авторизаційних дозволів (наприклад читанням або записом).

Сервер ресурсу / авторизації: API [29]. Сервер ресурсу зберігає захищені призначені для користувача аккаунти, а сервер авторизації перевіряє особу користувача і потім видає додатком маркери доступу.

З точки зору розробників додатків, API сервісу виконує обидві ролі, як сервера ресурсу так і сервера авторизації. Ми будемо перенаправляти до обох з цих комбінованих ролей, як роль сервісу так і роль API.

Клієнт: додаток. Клієнт це програма, яка хоче отримати доступ до аккаунту користувача. Перед тим, як воно це зробить, воно повинно бути дозволено користувачем і авторизація повинна бути перевірена API. Використовує абстрактний потік протоколу, який зображений на рисунку 4.5.



Риснок 4.5. – Загальна схема OAuth авторизації

1. Додаток запитує від користувача дозвіл на доступ до ресурсів сервісу
2. Якщо користувач дозволив запит, то додаток отримує надання авторизації
3. Додаток запитує маркер доступу від сервера авторизації (API сервісу) надавши посвідчення його автентичності та надання авторизації

4. Якщо ідентичність додатки справжня і надання авторизації дійсно, то сервер авторизації (API) видає з додатком маркер доступу. Авторизація завершена.

5. Додаток запитує ресурс від сервера ресурсу (API) і надає маркер доступу для підтвердження автентичності

6. Якщо маркер доступу дійсний, сервер ресурсу (API) передає ресурс з додатком.

Фактичний потік цього процесу буде відрізнятися в залежності від типу надання авторизації знаходиться у використанні - це загальна ідея зображена на рисунку 4.5.

Реєстрація додатку відбувається належним чином. Перед використанням OAuth у додатку, Необхідно зареєструвати свій додаток у сервісі авторизації.

Необхідно надавати таку інформацію і ймовірно детальну інформацію про додатку:

– назва додатка;

– сайт додатку;

– URI або URL зворотного виклику.

Перенаправлення на URI використовується там [30], де сервіс буде перенаправляти користувача після того, як заявка буде авторизована або заборонена, отже, частина вашої програми, яка буде обробляти коди авторизації або маркери доступу.

Ідентифікатор клієнта і секретний код клієнта реалізовуться наступним чином.

Як тільки додаток зареєстровано, сервіс буде видавати «облікові дані клієнта» у вигляді ідентифікатора клієнта і секретного коду клієнта.

Ідентифікатор клієнта це публічний рядок, який використовується API сервісу для ідентифікації додатка, і так само використовується для побудови авторизаційних URL'ів, які представляються користувачам.

Секретний код клієнта використовується для установки справжності додатки для API сервісу, коли додаток запитує доступ до аккаунту користувача і повинен бути зберігатись в секреті між додатком і API.

В абстрактному потоці протоколу, перші чотири кроки описують отримання надання авторизації і маркера доступу. Тип надання авторизації залежить від методу використовуваного додатком для запиту авторизації і типами надання.

OAuth 2 визначає 4 типи надання, кожен з яких по своєму корисний в різних ситуаціях:

– код авторизації: використовується разом з додатками на стороні сервера;

– неявність (прихованість): використовується разом з мобільними додатками або веб-додатками (додатки, які запускаються на пристрої користувача);

– власник ресурсу пароля облікових даних: використовується разом з довіреними додатками, такими як ті, що знаходяться у власності самого сервісу;

– облікові дані клієнта: використовується з додатками, що використовують відкрите API.

Код авторизації використовується наступним чином, тип надання коду авторизації є найбільш використовуваним так як він оптимізований під додатки на стороні сервера, де вихідний код не показаний відкрито і конфіденційність секретного коду клієнта може бути збережена.

Цей процес зображений на рисунку 4.6, на основі перенаправлення, що означає що програма має бути здатна взаємодіяти з юзерагентом та іншими узлами даної інформаційно технології, включаючи інтерфейс користувача (тобто з браузером користувача) і отримувати коди авторизації API, які прямують через юзерагент.

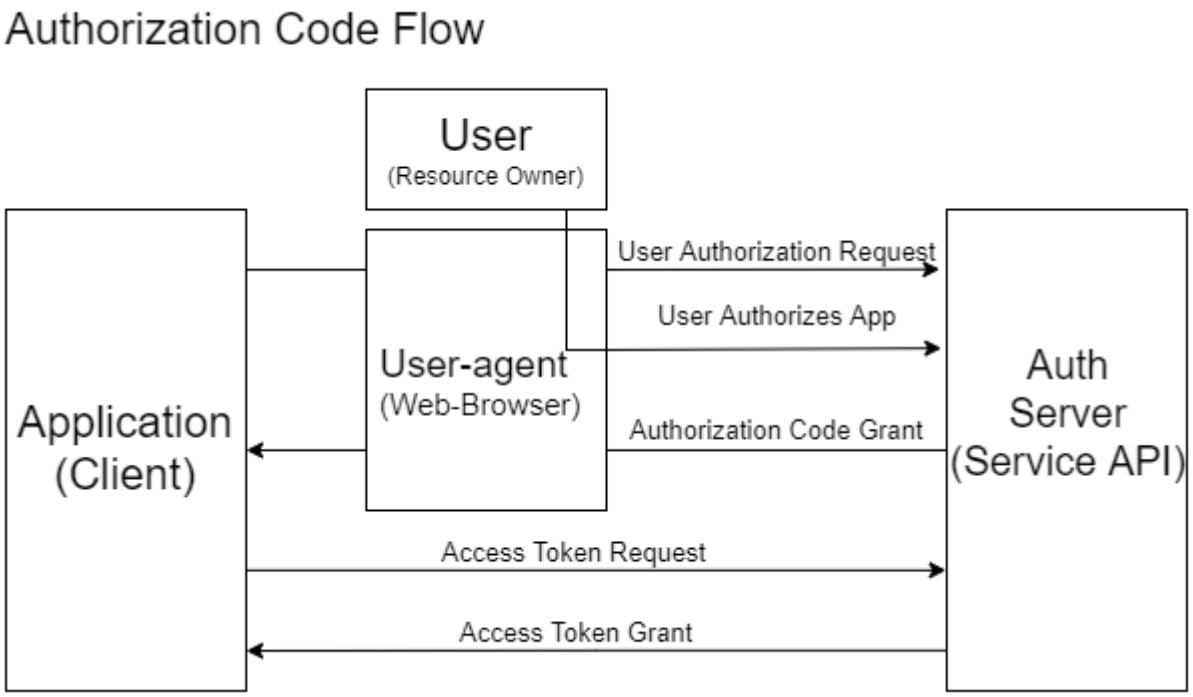


Рисунок 4.6. - Загальна схема OAuth2 авторизації

1. Посилання коду авторизації

По-перше, користувачеві надається посилання коду авторизації, яка виглядає наступним чином:

https://cloud.digitalocean.com/v1/oauth/authorize?response_type=code&client_id=CLIENT_ID&redirect_uri=CALLBACK_URL&scope=read

https://cloud.digitalocean.com/v1/oauth/authorize - кінцева точка авторизації API.

response_type = code - передбачає що ваш додаток запитує надання коду авторизації.

client_id = CLIENT_ID - ідентифікатор клієнта додатка (значення за яким API визначає додаток).

redirect_uri = CALLBACK_URL - місце куди сервіс перенаправляє браузер після того як код авторизації надано.

scope = read - визначає рівень доступу, який запитує додаток

2. Користувач авторизує додаток

Коли користувач натискає на посилання, він повинен спочатку увійти в сервісі, для посвідчення його особи. Потім сервіс запросить у нього підтвердження на дозвіл або заборону доступу додатку до їх аккаунту.

3. Додаток отримує код авторизації

Якщо користувач натискає на «Авторизувати додаток», то сервіс перенаправляє браузер на URI переадресації додатку, яка була вказана при реєстрації клієнта, разом з кодом авторизації. Переадресація буде виглядати так: https://dropletbook.com/callback?code=AUTHORIZATION_CODE

4. Додаток отримує маркер доступу

Додаток запитує маркер доступу від API передаючи код авторизації кінцевій точці маркера API, разом з докладною інформацією про ідентифікацію, включаючи секретний код клієнта. Ось приклад запиту через POST до кінцевої точки маркера DigitalOcean: https://cloud.digitalocean.com/v1/oauth/token?client_id=CLIENT_ID&client_secret=CLIENT_SECRET&grant_type=authorization_code&code=AUTHORIZATION_CODE&redirect_uri=CALLBACK_URL

5. Додаток отримує маркер доступу

Якщо авторизація пройшла успішно, API відправить відповідь з додатком містить маркер доступу і опціонально маркер поновлення.

Тепер додаток авторизовано. Він може використовувати маркер для доступу до аккаунту користувача через АПІ сервісу, з обмеженнями по доступу, поки маркер не закінчиться, чи не буде скасований.

Якщо маркер поновлення був переданий, то він може бути використаний для запиту нових маркерів доступу, якщо термін дії вихідного маркера закінчився.

Також тут присутній тип неявність або прихованість.

Тип неявного надання використовується для мобільних додатків і веб-додатків (тобто додатків, які запускаються в браузері), де конфіденційність секретного коду клієнта не гарантується.

Тип неявного надання також є процесом на основі переадресації, але маркер доступу дається браузеру для передачі його додатку, так що він може бути доступний як користувачу, так і іншим додаткам на пристрої користувача.

Крім того, цей процес не перевіряє справжність ідентичності додатка і годиться на перенаправлення на URI (який був зареєстрований з сервісом), щоб служити цій меті. Тип неявного надання не підтримує маркери поновлення.

Процес неявного надання в основному працює наступним чином: користувачеві пропонується авторизувати додаток, потім сервер авторизації передає маркер доступу назад браузеру, який в свою чергу передає його додатку. Зображено на рисунку 4.7.

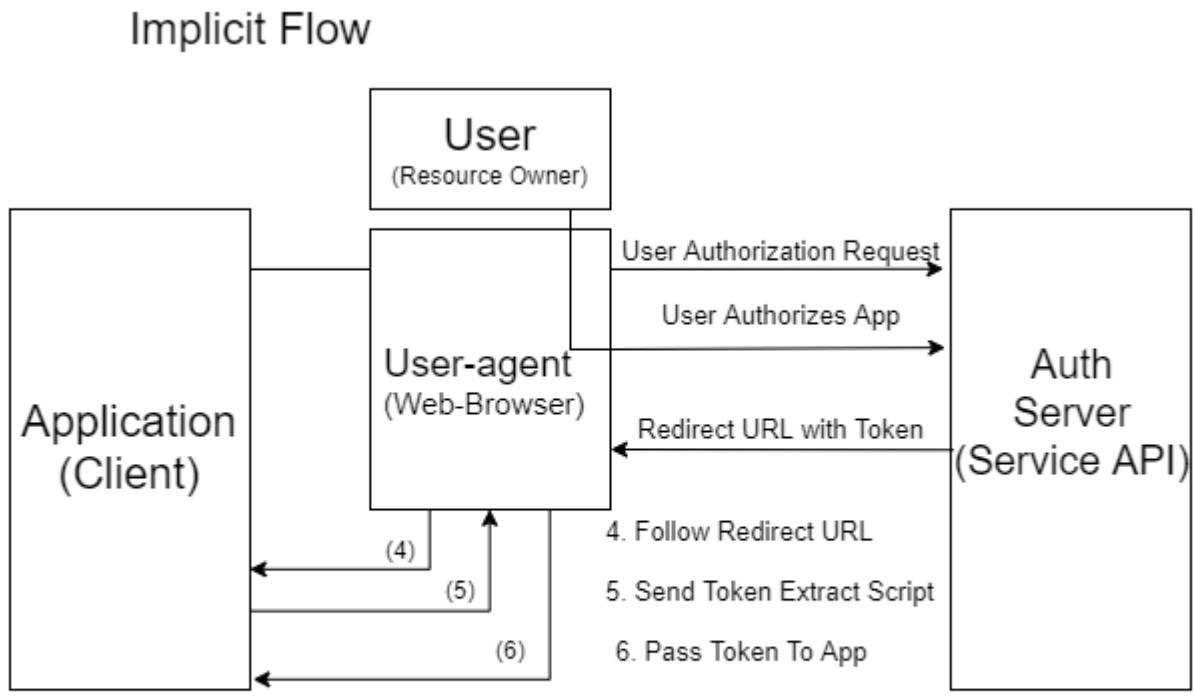


Рисунок 4.7. - Загальна схема OAuth2 авторизації для клієнтських додатків

Тип надання облікових даних клієнта надає додатку спосіб доступу до свого особистого облікового запису сервісу. Приклади того, коли це може бути корисним, якщо додаток хоче оновити свою зареєстровану інформацію або URI переадресації, або звернутися до інших даних збережених в акаунті сервісу через API.

Додаток запитує маркер доступу відправляючи свої облікові дані, свої ідентифікатори клієнта і секретний код клієнта, серверу авторизації [31-32].

Якщо облікові дані клієнта пройшли перевірку, сервер авторизації повертає додатку маркер доступу. З цього моменту додатку дозволено використовувати свій власний аккаунт!

Власник ресурсу облікових даних, тобто, користувач надає свої облікові дані сервісу (ім'я користувача і пароль) безпосередньо додатку, який використовує облікові дані для отримання маркера доступу від сервісу.

Цей тип надання повинен бути активізованим на сервері авторизації, тільки якщо інші потоки не є життєздатними.

Також він повинен використовуватися, якщо додаток є довірений для сервісу (наприклад, якщо він належить сервісу або операційній системі комп'ютера).

Після того як користувач передав свої облікові дані додатку, після цього він повинно запросити маркер доступу у сервера авторизації. POST запит виглядає так:

https://oauth.example.com/token?grant_type=password&username=USERNAME&password=PASSWORD&client_id=CLIENT_ID

Якщо облікові дані користувача пройшли перевірку, сервер авторизації повертає маркер доступу з додатком. Тепер додаток авторизовано!

Бізнес-логіка сервісу винесена в окремий простір імен, цей модуль є framework-agnostic, що означає незалежність від вибраної технології, тільки від мови програмування. Це одночасно і є нашою доменною-моделлю. Зображено на рисунку 4.8.

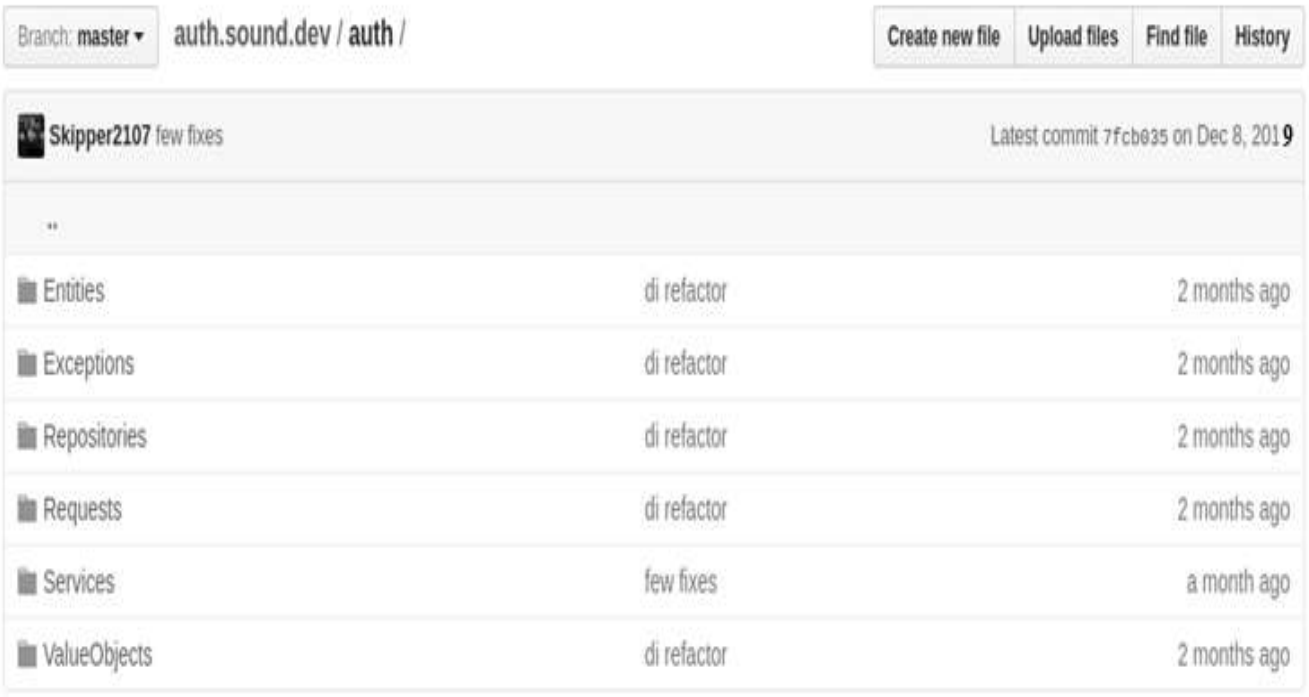


Рисунок 4.8 - Початковий код інформаційної технології моніторингу виконання завдань авторизації на github

Головним сервіс-класом сервісу є однозначно LoginService зображено на рисунку 4.9, адже в процесі роботи, цей клас буде найчастіше буде проходити процес бутстрапа. Він реалізує такі патерни як DependencyInjection та AgnosticService.

Даний клас дозволяє авторизувати користувачів інформаційної технології моніторингу виконання завдань у веб-ресурсі за допомогою ім’я користувача, тобто електронної адреси та номеру телефона, а згодом користувач додатку придумує пароль доступу до ресурсу. Також патерн DependencyInjection запобігає виявленню помилок на стороні клієнта і допомогає користувача, який уже зареєстрований, але забувся пароль до системи, відновити його без втрати інформації на своєму акаунті.

```php

use Skipper\Auth\Entities\User;
use Skipper\Auth\Exceptions\WrongCredentialsAuthException;
use Skipper\Auth\Repositories\UserRepository;
use Skipper\Auth\Requests\Login;

/**
 * Class LoginService
 * @package Skipper\Auth\Services
 */
class LoginService
{

    /** @var UserRepository $users */
    protected $users;
    /** @var AppService $apps */
    protected $apps;
    /** @var CipherService $cipher */
    protected $cipher;

    /**
     * LoginService constructor.
     * @param UserRepository $userRepository
     * @param AppService $appService
     * @param CipherService $cipherService
     */
    public function __construct(UserRepository $userRepository, AppService $appService, CipherService $cipherService)
    {
        $this->users = $userRepository;
        $this->apps = $appService;
        $this->cipher = $cipherService;
    }

    /**
     * @param Login $request
     * @return User
     * @throws WrongCredentialsAuthException
     * @throws \Skipper\Exceptions\DomainException
     */
    public function login(Login $request): User
    {
        $user = $this->users->getUserByLogin($request->getLogin());
        if (!$this->cipher->check($request->getPassword(), $user->getPassword())) {
            throw new WrongCredentialsAuthException();
        }
        return $user;
    }
}
```

Рисунок 4.9. - Початковий код класу LoginService

Проаналізувавши найпопулярніші процеси авторизації, було прийняте рішення реалізувати сервіс авторизації на основі токенів, з основою OAuth2 технології.

Оскільки очікується, що найбільше навантаження на сервіс (як основну точку входу) складе веб-версія, то має сенс виконати сервіс авторизації з декількома активними методами входу.

Веб-версія буде використовувати процес облікових даних, решта сервісів - чистий OAuth2.

4.4 Реалізація складових у інформаційній технології моніторингу виконання завдань

Для реалізації інформаційної технології моніторингу виконання завдань були реалізовані такі складові, як інтерфейс користувача, удосконалення оповіщення до завдань, формування завдань керівником проекту та моніторинг виконання завдань з урахуванням дедлайну.

У магістерській кваліфікаційній дипломній роботі виконується реалізація інформаційної технології моніторинга виконаня завдань у середовищі Visual Studio з використанням мови розробки – C# та бібліотеки для інтерфейсу користувача React. Для роботи основного компонента реалізовано групу класів, які знаходяться в бібліотеці NotificationBot.Core.

Основі класи та методи інформаійної технології моніторингу виконання завдань Message, Companion, MessageExclodes, Tokens, Users, TimeOfMessage.

Представлення класів та методів буде представлано нижче з їх коротким описом.

Message – клас, якиий дозволяє відправляти та відміняти повідомлення виконавцям, а також дозволяє створювати дату та час оповіщення у певний деньголовну форму програми і який містить методи обробки подій від елементів керування. Містить елементи управління відправки оповіщень користувачам.

Companion – клас, який дозволяє добавляти користувачів у список важих груп.

MessageExclodes – клас, який реалізує вибір точної дати та часу відправки повідомлення, а також відмінти оповіщення, якщо воно відправлено помилково.

Tokens – клас, який призначений для зміни пароля в обліковому записі програмного засобу, а також облікових записів Telegram та Viber.

Users– клас, який дозволяє створювати нового користувача у даній системі моніторингу виконання завдань, показує дату та час створення, зберігає паролі у базі даних у вигляді хеш-таблиці(зашифрованих).

TimeOfMessage – клас, у якому реалізовано точний час у певний день, коли було відправлено оповіщення виконавцю.

public static List<Bitmap> GetStrings(Bitmap text) – метод для виділення символів оповіщення.

public static List<Bitmap> GetStringNotification (Bitmap str) – метод самого оповіщеня.

public static List <Bitmap> GetNotJs(Bitmap text) – метод передаваня символів тексту.

public static Bitmap TrimBitmap(Bitmap bmp) – метод обрізання білих полів навколо оповіщення.

public void Notification (Bitmap b, int classindex) – метод відправки оповіщення.

public string Users(Bitmap b) – метод створення користувачів, тобто їхнього облікового запису.

public void Friends() – метод добавлення користувачів у список друзів.

public void DeserializeParams() – метод читання параметрів завдання з оповіщення.

public static SchedulerExclodes (Bitmap b, Size sz) – метод, який задає певний час та дату закінчення оповіщення виконавцю.

public static Tokens Inverse Tokens (Bitmap b) – метод зміни пароля облікового запису.

public static Give Comment Status (Bitmap b) – метод додавання коментарів до заданого завдання.

Для реалізації інформаційної технології моніторингу виконання завдань використовуються також інші вузли, реалізація яких наведена у додатку Б.

Інтерфейс користувача – графічний інтерфейс, який надає змогу керувати системною за допомогою графічних об’єктів і дій над ними.

Для проектування інформаційної системи необхідним є створення графічних схем інтерфейсу. Структурна схема інтерфейсу інформаційної системи моніторингу виконання завдань зображено на рисунку 4.10.

Риснок 4.10. – Структурна схема інтерфейсу інформаційної технології моніторингу виконання завдань

Інтерфейс користувача інформаційної системи моніторингу виконання завдань складається з таких сторінок:

– сторінка для авторизації користувача;
– сторінка для реєстрації користувача;
– сторінка для підтвердження реєстрації користувача;
– сторінка для управління завданнями;
– сторінка «користувачі».

Графічна схема інтерфейсу сторінки для авторизації зображено на рисунку Сторінка для авторизації складається з таких елементів:

1. текст «Вхід в систему»;
2. поле для ведення номера телефону;
3. поле для ведення паролю користувача;
4. кнопка «Вхід».

Графічна схема інтерфейсу сторінки користувача для управління завданнями зображена на рисунку 4.11. На даній сторінці використовується панель для навігації між сторінками. Сторінка для управління завданнями складається з таких елементів:

1. навігаційне меню;
2. текст «Менеджер завдань»;
3. таблиця із завданнями;
4. номера телефонів та ПІБ користувачів;
5. опис завдань;
6. дата створення завдань;
7. кнопка редагування завдань;
8. пагінація для таблиці;
9. кнопка для створення завдання.

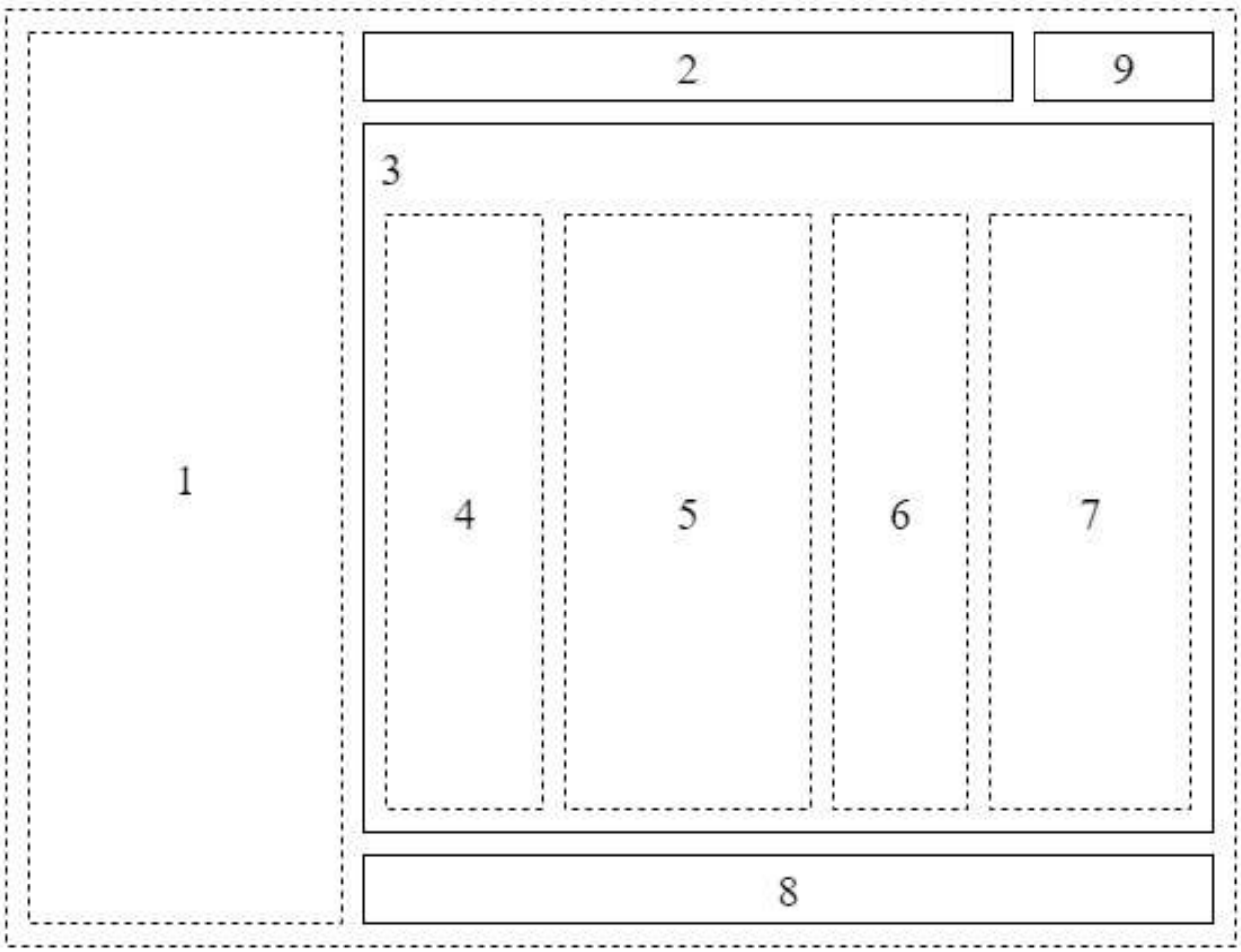


Рисунок 4.11 – Графічна схема інтерфейсу сторінки користувача для управління завданнями

Ключовим кроком інформаційної технології моніторингу виконання завдань є проектування загального алгоритму роботи. UML-діаграма алгоритму роботи інформаційної системи моніторингу виконання завдань зображено на рисунку 4.12. Основними кроками алгоритму є:

1. початок роботи інформаційної системи моніторингу (ініціалізація всіх вузлів);
2. реєстрація або авторизація користувача;
3. Navigate – перехід по сторінкам;
4. якщо вибрано «Менеджер завдань», то викликаємо метод ShowTasks;
   4.1. якщо вибрано «Видалити», то викликаємо метод DeleteTask;
   4.2. якщо вибрано «Створити», то викликаємо метод AddTask;
   4.3. якщо вибрано «Відмінити», то викликаємо метод CancelTask;

5. якщо вибрано «Контактни», то викликаємо метод ShowContacts;
   5.1. якщо вибрано «Видалити», то викликаємо метод DeleteContact;
   5.2. якщо вибрано «Створити», то викликаємо метод AddContact;
   5.3. якщо вибрано «Підтвердити», то викликаємо метод ConfirmContact;
6. якщо вибрано «Налаштування», то викликаємо метод ShowSettings;
   6.1. SaveSettings – збереження змінених параметрів;
7. Якщо вибрано «Вихід», то завершуємо роботу.

У результаті, створена UML-діаграма роботи інформаційної технології моніторингу виконання завдань, яка зображена на рисунку 4.12.

4.5 Аналіз роботи інформаційної технології моніторингу виконання завдань

У широкому сенсі, тестування – це одна з технік контролю якості (Quality Control), яка включає планування, складання тестів, безпосередньо виконання тестування і аналіз отриманих результатів.

Важливо розуміти, що тестування ПЗ включає в себе не тільки проведення тестів, але і багато інших дій, пов’язаних з процесом забезпечення якості:

1. Аналіз і планування.
2. Розробку тестових сценаріїв.
3. Оцінку критеріїв закінчення тестування.
4. Написання звітів.
5. Рецензування документації (в тому числі і вихідного коду).
6. Проведення статичного аналізу.

Розглянемо наступні методи тестування: «біла скринька» і «чорна скринька».

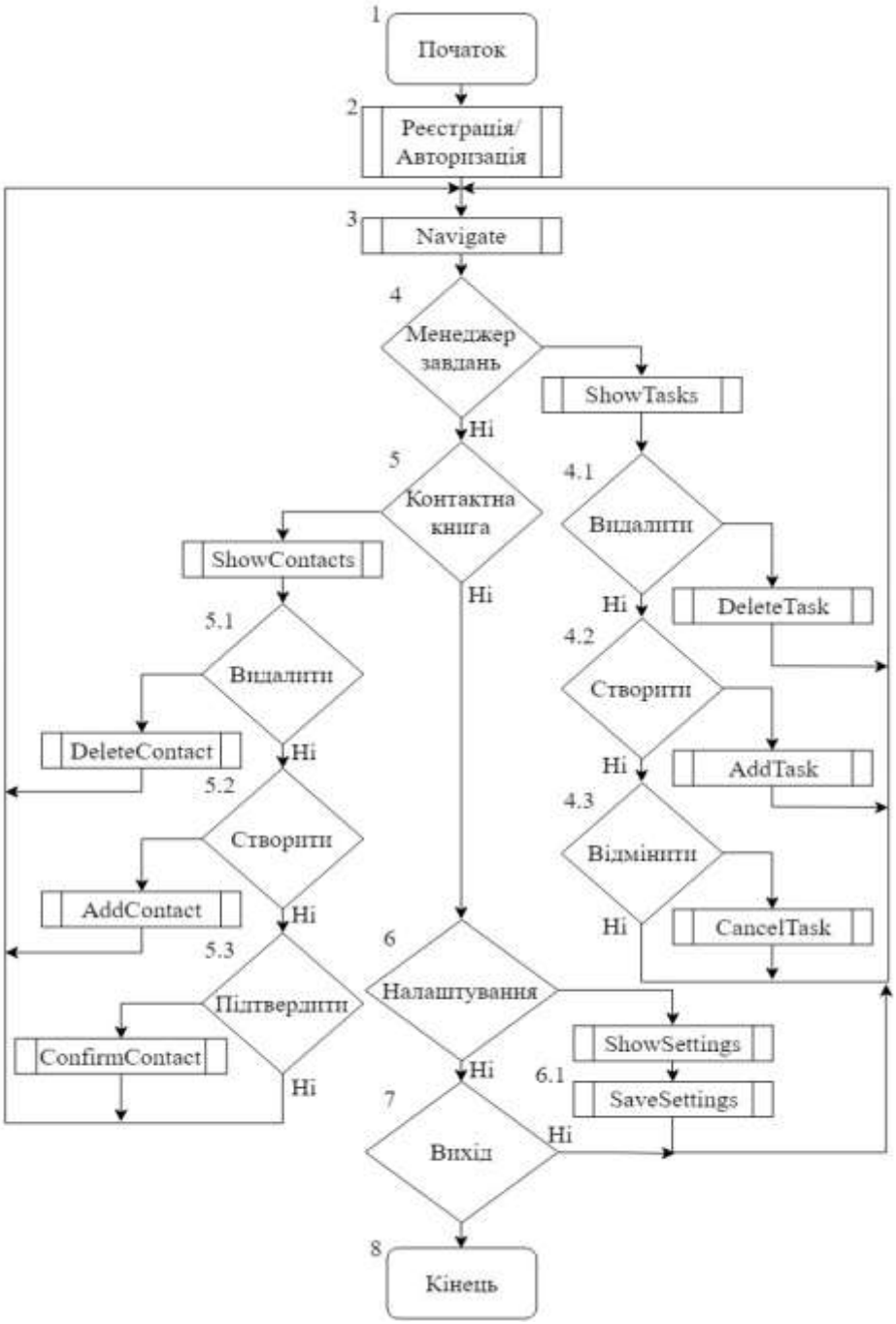


Рисунок 4.12. – UML-діаграма роботи оповіщень інформаційної технології моніторингу виконання завдань

Метод тестування «біла скринька» виконується розробником (так як необхідно знати внутрішні принципи роботи програми) для перевірки внутрішньої структури програмного засобу. Об’єктом тестування є дані,

отримані шляхом аналізу логіки програми. Перевіряється коректність побудови всіх елементів програми та правильність їхньої взаємодії один з одним. Зазвичай аналізуються керуючі зв'язки елементів, рідше – інформаційні зв'язки. Тестування за принципом «білої скриньки» характеризується ступенем, в якому тести виконують або покривають логіку (вихідний текст) програми. Зазвичай тестування «білої скриньки» засноване на аналізі керуючої структури програми. Програма вважається повністю перевіреною, якщо проведено вичерпне тестування маршрутів (шляхів) її графа управління.

Принцип «білої скриньки» дозволяє врахувати особливості програмних помилок:

1. Попередні припущення про ймовірність потоку керування або даних у програмі часто бувають некоректними. У результаті типовим може стати маршрут, модель обчислень за яким опрацьована слабо.

2. Кількість помилок мінімально в «центрі» і максимально на «периферії» програми.

3. Деякі результати в програмі залежать не від вихідних даних, а від внутрішніх станів програми.

4. При записі алгоритму програмного забезпечення у вигляді тексту на мові програмування можливе внесення типових помилок трансляції (синтаксичних та семантичних).

Метод тестування «чорна скринька» використовується, коли тестувальнику не потрібно знати внутрішні властивості програми. При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе. Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок.

Тести демонструють:

1. Як виконуються функції програми.
2. Як приймаються вихідні дані.
3. Як виробляються результати.

4. Як зберігається цілісність зовнішньої інформації.

Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми. Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. Принцип «чорної скриньки» – не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

1. Некоректних чи відсутніх функцій.
2. Помилок інтерфейсу.
3. Помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних.
4. Помилок характеристик (необхідна ємність пам'яті і т. д.).
5. Помилок ініціалізації та завершення.

Існує метод тестування «сіра скринька», яка створена на основі розглянутих методів. При роботі з даним методом тестувальник має доступ до коду програми, проте тестування проводить з точки зору кінцевого користувача. Суть даних методів не є складною, проте ефективність тестування за допомогою кожного з них вимагає хороших знань та навичок. В результаті, обрано метод тестування «чорна скринька», оскільки за даним методом тестуються лише вхідні/вихідні дані.

Для перевірки працездатності розробленого програмного засобу було проведено низка досліджень.

Нехай потужність множини задач, що сформовані на етапі інфологічного моделювання дорівнює 3, дедлайн виконання завдання задається керівником, а потужність множини облікових записів месенджерів дорівнює 2. Процес введення такої вхідної інформації у моніторинг виконання завдань поданий на рисунку 4.13, а запропонований програмний засіб – на рисунку 4.14.

Дослідження були проведені при зміні множини задач на множині завдань потужністю 1000, що сформовані на етапі інфологічного моделювання.

Завдання задано керівником, а потужність множини завдань на один обліковий запис дорівнювала 5. Кожен з результатів, представлених у таблиці, є середнім результатом 1000 дослідів з таких пареметрів як кількість завдань, дедлайн виконання завдання, формування завданя керівником проекту групі виконавців, ствrення груп користувачів, відправка завдань на облікові записи користувачів.

Створена інформаційна технологія моніторингу виконання завдань порівнювалась з програмним продуктом Hubstaff. Дане дослідження показало, що програмний засіб Hubstaff негативно справився з моніторингом виконання завдань при зміні потужностей множини задач та швидкодії відправки оповіщень до виконавців, а також програмний засіб Hubstaff, не дає можливості створювати багато оповіщень для груп користувачів, тому ця задача не є актуальною. Результати моніторингу виконання завдань наведені у таблиці 4.3.

Таблиця 4.3 – Результати моніторингу виконання завдань

Кількість завдань заданих виконавцю або групі виконавців	Виконаних завдань	Невиконаних завдань	Порівняльний аналіз виконання завдань через облікові записи запропонованою IT /Hubstaff (%)
1	1	0	100/100
3	2	1	70/30
5	5	0	100/100
10	9	1	90/10
14	12	2	85/15
5	5	0	100/100
1	1	0	100/100
8	6	2	75/25
11	11	0	100/100

Продовження Таблиці 4.3.

18	16	2	88/12
21	18	3	85/15
4	3	1	75/25
1	1	0	100/100
12	11	1	91/9
25	21	4	84/16
2	2	0	100/100
14	2	2	85/15

Як видно з таблиці 4.3, швидкодія моніторингу виконання завдань з використанням запропонованих технологій збільшиласть на 15%.

На рисунку 4.13. зображена початкова активність при вході до інформаційної технології моніторингу виконання завдань. Після авторизації система перевіряє чи користувач має права головного юзера. Якщо користувач відсутній, то користувач не зможе ввійти у систему, якщо користувач забув свій пароль до облікового запису, то він має можливість його відновити через електронну адресу чи обліковий запис телеграму, процес відновлення паролю зображений на рисунку 4.14. Реєстрація нового користувача представлена на рисунку 4.15.

Головний користувач має можливість додавати виконавців до системи автоматично, чи кожен користувач може зареєструватися окремо у системі через заповнення усіх необхідних параметрів у полі реєстрації.

Для створення оповіщення користувачеві необхідно перейти у пункт завдання, вибрати виконавців чи сформувати групу виконавці, які будуть виконувати певне завдання, вказати текст повідомлення, дату закінчення виконання завдання, інформаційна технологія сама визначить статус завдання та відсоток виконаних робіт.

Процес додавання оповіщень користувачеві зображений на рисунку 4.16.

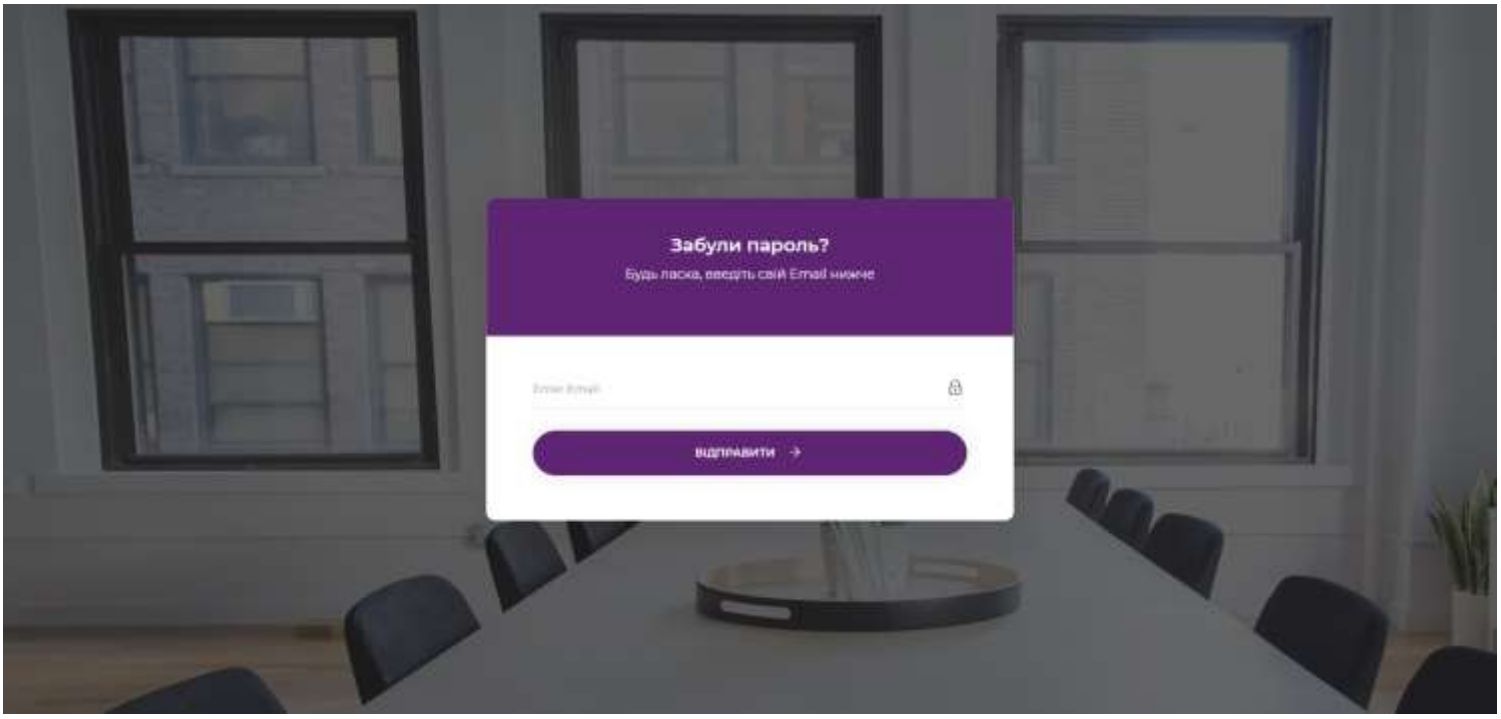


Рисунок 4.13 – Початкова активність

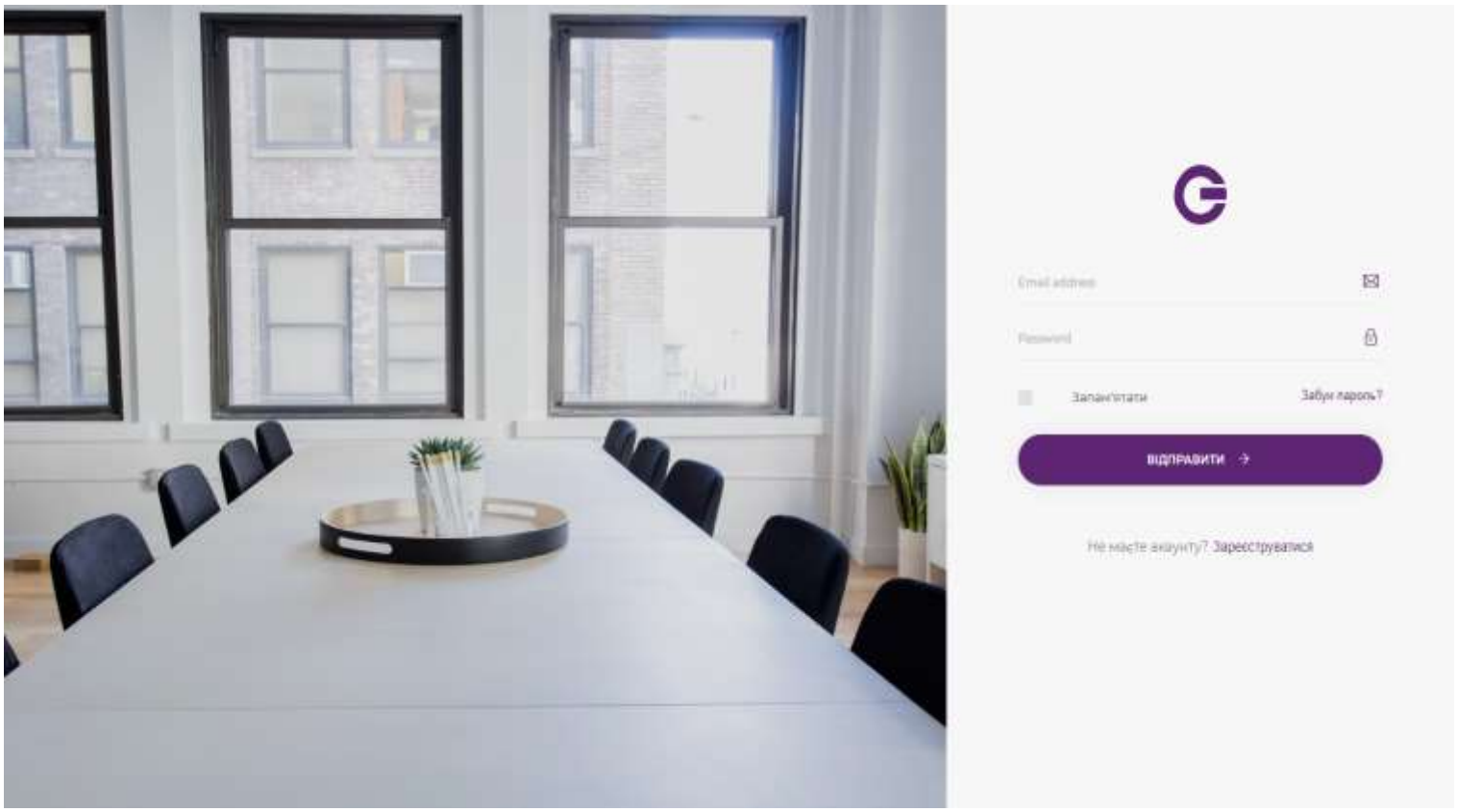


Рисунок 4.14 – Відновлення паролю

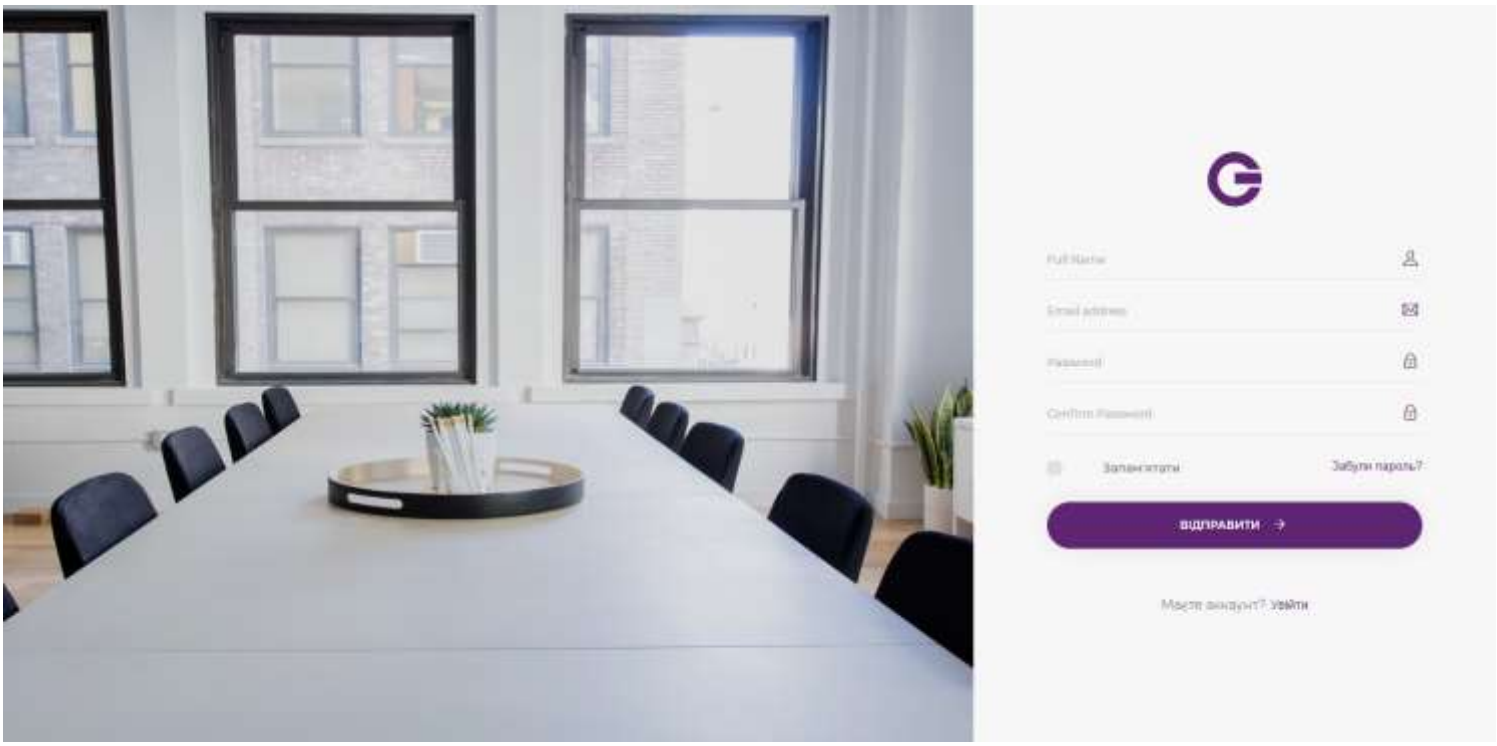


Рисунок 4.15 – Реєстрація користувача

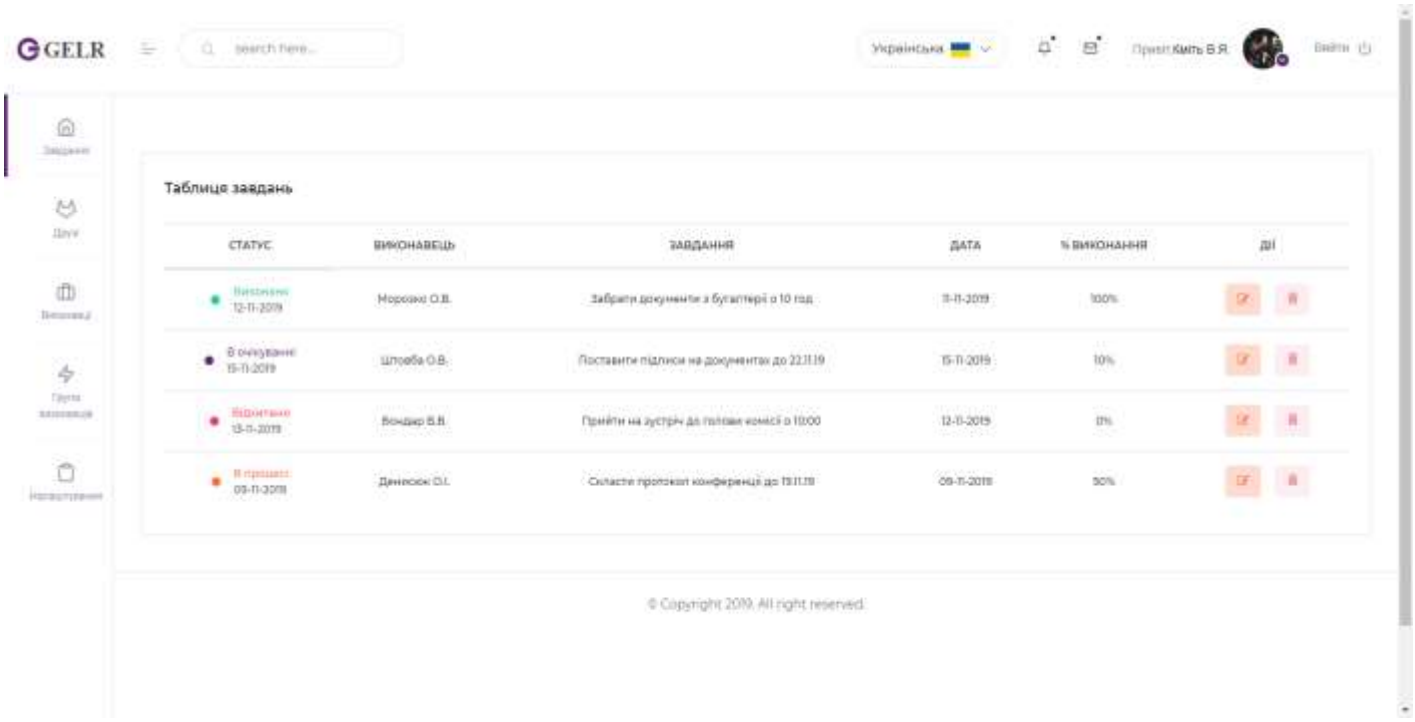


Рисунок 4.16 – Процес добавлення оповіщень

5 ЕКОНОМІЧНИЙ РОЗДІЛ

4.1 Оцінювання комерційного потенціалу розробки

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

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

Результати оцінювання комерційного потенціалу розробки наведено в таблиці 5.1.

Таблиця 5.1 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта	
	1. Експерт 1	2. Експерт 2
	Бали, виставлені експертами:	
1	4	4
2	4	3
3	3	4
4	4	3
5	3	4
6	4	4
7	3	3
8	4	4
9	4	4
10	4	3
11	3	4
12	3	4
Сума балів	$СБ_1 = 44$	$СБ_2 = 44$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{2} = 44$	

Отже, з отриманих даних таблиці 5.1 видно, що нова розробка має високий рівень комерційного потенціалу.

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

Для розробки нового програмного продукту необхідні такі витрати.

Основна заробітна плата для розробників визначається за формулою (5.1):

$$З_о = \frac{М}{Т_р} \cdot t, \quad (5.1)$$

де М- місячний посадовий оклад конкретного розробника;

$Т_р$ - кількість робочих днів у місяці, $Т_р$ = 21 день;

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

Розрахунки заробітних плат для керівника і програміста наведені в таблиці 5.2.

Таблиця 5.2 – Розрахунки основної заробітної плати

Працівник	Оклад М, грн.	Оплата за робочий день, грн.	Число днів роботи, t	Витрати на оплату праці, грн.
Науковий керівник	5500	261,90	5	1309,5
Інженер-програміст	4000	190,47	50	9523,5
Всього:				10833

Розрахуємо додаткову заробітну плату:

$$З_{дод} = 0,1 \cdot 10833 = 1083,3 \text{ (грн.)}$$

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

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

$$Н_{зп}= (10833+1083,3)\cdot \frac{36,3}{100} = 4325,5 \text{ (грн.)}.$$

Розрахунок амортизаційних витрат для програмного забезпечення виконується за такою формулою:

$$А=\frac{Ц\cdot Н_а}{100}\cdot \frac{Т}{12}, \quad (5.3)$$

де Ц – балансова вартість обладнання, грн;

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

Т – Термін використання (Т=3 міс.).

Таблиця 5.3 – Розрахунок амортизаційних відрахувань

Найменування програмного забезпечення	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань,грн
Персональний комп'ютер	9000	25	3	562,5
Всього:				562,5

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

$$K = \sum_1^n Н_і \cdot Ц_і \cdot К_і, \quad (5.4)$$

де n – кількість комплектуючих;

Ні - кількість комплектуючих і-го виду;

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

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

Таблиця 5.4 - Витрати на комплектуючі, що були використані для розробки ПЗ.

Найменування матеріалу	Одиниці виміру	Ціна, грн.	Витрачено	Вартість витрачених матеріалів, грн.
Флешка	шт.	180	1	180
Пачка паперу	уп.	120	1	120
Ручка	шт.	10	1	10
Всього з урахуванням транспортних витрат				341

Витрати на силову електроенергію розраховуються за формулою:

$$В_е=В\cdot П\cdot Ф\cdot К_п\ ; \quad (5.5)$$

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

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

Ф – фактична кількість годин роботи комп'ютера (Ф=190 год.);

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

$$В_е= 1,7 \cdot 0,6\cdot 190\cdot 0,7 = 135,66 \text{ (грн.)}$$

Розрахуємо інші витрати $В_{ін}$.

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

$$В_{ін} = (1..3) \cdot (З_о + З_р). \quad (5.6)$$

Отже, розрахуємо інші витрати:

$$В_{ін} = 1 * (10833 + 1083,3) = 11916,3 \text{ (грн).}$$

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

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

В = 10833+1083,3+4325,5+562,5+341+135,66+11916,3= 29197,26 (грн.)

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

$$В_{заг} = \frac{В_{ін}}{\alpha} \quad (5.7)$$

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

$$В_{заг} = \frac{29197,26}{1} = 29197,26$$

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

$$ЗВ = \frac{В_{заг}}{\beta} \quad (5.8)$$

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

Отже, розрахуємо загальні витрати:

$$ЗВ = \frac{29197,26}{0,9} = 32441,4 \text{ (грн.)}$$

5.3 Прогнозування комерційних ефектів від реалізації результатів розробки.

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

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

$$\Delta П_i = \sum_1^n (\Delta П_я \cdot N + П_я \Delta N)_i \quad (5.9)$$

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

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

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

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

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

В результаті впровадження результатів наукової розробки витрати на виготовлення інформаційної технології зменшаться на 20 грн (що автоматично спричинить збільшення чистого прибутку підприємства на 20 грн), а кількість користувачів, які будуть користуватись збільшиться: протягом першого року – на 200 користувачів, протягом другого року – на 150 користувачів, протягом

третього року – 100 користувачів. Реалізація інформаційної технології до впровадження результатів наукової розробки складала 700 користувачів, а прибуток, що отримував розробник до впровадження результатів наукової розробки – 200 грн.

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

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

$$\Delta П_1 = 20 \cdot 700 + (200 + 20) \cdot 200 = 58000 \text{ грн.}$$

Протягом другого року:

$$\Delta П_2 = 20 \cdot 700 + (200 + 20) \cdot (200 + 150) = 91000 \text{ грн.}$$

Протягом третього року:

$$\Delta П_3 = 20 \cdot 700 + (200 + 20) \cdot (200 + 150 + 100) = 113000 \text{ грн.}$$

5.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

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

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

$$Е_{абс} = (ПП - PV), \qquad (5.10)$$

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

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

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

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

Рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, рисунок 5.1.

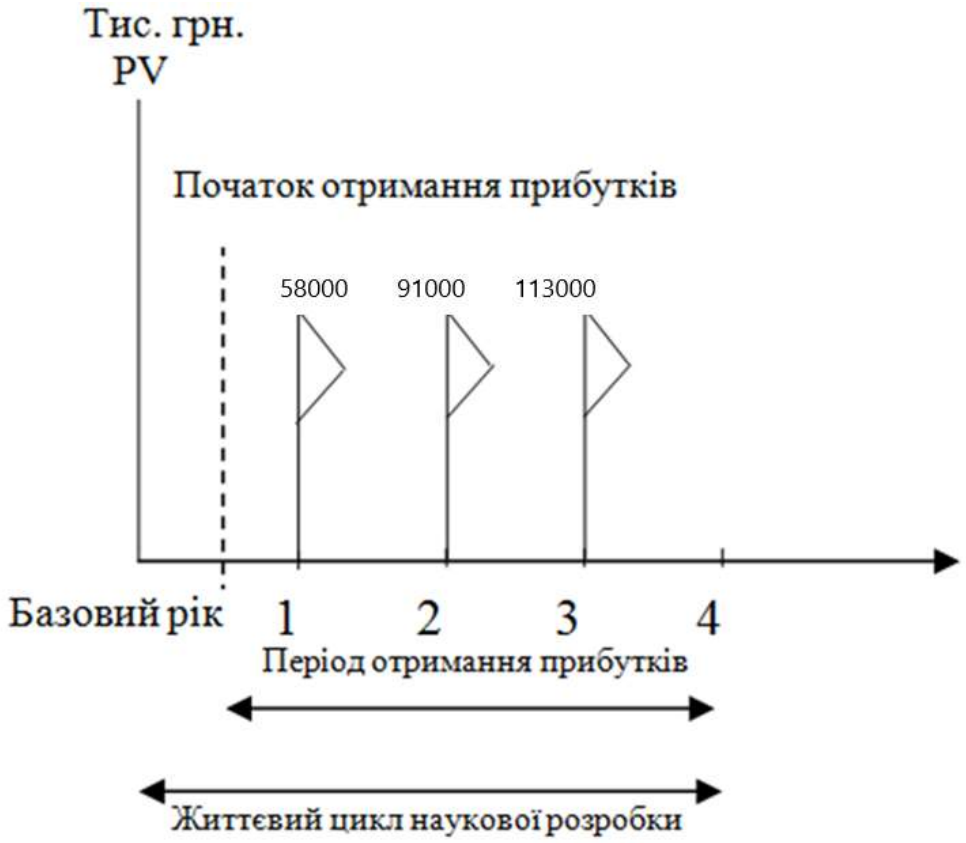


Рисунок 5.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Розрахуємо вартість чистих прибутків за формулою:

$$ПП = \sum_1^m \frac{\Delta П_i}{(1+\tau)^t} \quad (5.11)$$

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

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

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

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

Отже, розрахуємо вартість чистого прибутку:

$$ПП = \frac{32441,4}{(1+0,1)^0} + \frac{58000}{(1+0,1)^2} + \frac{91000}{(1+0,1)^3} + \frac{113000}{(1+0,1)^4} = 225925,44 \text{ (грн.)}$$

Тоді розрахуємо $E_{абс}$:

$$Е_{абс} = 225925,44 - 32441,4 = 193484,04 \text{ грн.}$$

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

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$ за формулою:

$$Е_в = \sqrt[Т]{1 + \frac{Е_{абс}}{PV}} - 1 \qquad (5.12)$$

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

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$Т_ж$ – життєвий цикл наукової розробки, роки.

Тоді будемо мати:

$$Е_в = \sqrt[3]{1 + \frac{193484,04}{32441,4}} - 1 = 0,90 \text{ або } 90\ \%$$

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

$$\tau = d + f,$$

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

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

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

Оскільки $Е_в$ = 90% > $\tau_{мін}$ = 0,3 = 30%, то у інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

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

$$Т_{ок} = \frac{1}{Е_в}$$

$$Т_{ок} = \frac{1}{0,9} = 1,11 \text{ року}$$

Обрахувавши термін окупності даної наукової розробки, можна зробити висновок, що фінансування даної наукової розробки буде доцільним.

ВИСНОВКИ

При виконанні досліджень за визначеною тематикою магістерської кваліфікаційної роботи проведено аналіз моделей, методів моніторингу виконання завдань та доведено доцільність розробки інформаційної технології моніторингу виконання завдань.

Для реалізації інформаційної технології моніторингу виконання завдань було обрано мову програмування C# та середовище розробки Visual Studio. Для створення web-інтерфейсу використовувалася бібліотека React та середовище розробки Visual Studio Code.

Створено архітектуру інформаційно технології за допомогою діаграм варіантів використання, послідовності, класів та структурної схеми компонентів. Розроблено графічні схеми інтерфейсів головного вікна та основних компонентів. Проведено тестування графічного інтерфейсу інформаційної технології моніторингу виконання завдань з використанням методу тестування – «чорна скринька».

Під час дослідження було удосконалено модель моніторингу виконання завдань за рахунок впровадження додаткового інструментарію, а також удосконалено метод моніторингу виконання завдань за рахунок інтеграції облікових записів Telegram, Email користувачів, що дозволить ефективніше контролювати виконування завдання від керівника організації та швидше отримувати оповіщення про завдання на мобільний телефон чи інші девайси з підтримкою облікових записів. Внаслідок цього відбулось покращення швидкості на 15% та розширення функціоналу моніторингу виконання завдань.

За результатами досліджень подано заявку на реєстрацію авторського права на твір (комп'ютерну програму) “Інформаційна технологія моніторингу виконання завдань” (номер реєстрації заявки АПС/10094-19), а також опубліковано 2 тез доповіді з науково-технічних конференцій.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Савчук Т.О. Використання сучасних підходів підвищення ефективності процесу моніторингу виконання завдань/ Савчук Т., Кміть В. /Всеукраїнська науково-практична Інтернет-конференція «Молодь в науці: дослідження, проблеми, перспективи - 2019», URL: https://conferences.vntu.edu.ua/index.php/mn/mn2019/paper/view/8188/6821 Збірник праць. – Вінниця : ВНТУ, 2019.

2. Савчук Т.О. Підвищення ефективності моніторингу виконання завдань/ Савчук Т., Кміть В. /« Науково-технічна конференція підрозділів Вінницького національного технічного університету - 2019», URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2019/paper/view/7183/6255 Збірник праць. – Вінниця : ВНТУ, 2019.

3. Monitoring in science and technology [Електронний ресурс]. – Режим доступу: – https://en.wikipedia.org/wiki/Monitoring.

4. API Telegram [Електронний ресурс]. – Режим доступу: – https://core.telegram.org/.

5. ASP.NET — Вікіпедія – [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/ASP.NET.

6. Структурные карты Константайна – [Електронний ресурс]. – Режим доступу: http://www.interface.ru/fset.asp?Url=/case/defs71.htm.

7. Терещенко П. В. Інтерфейси інформаційних систем. / П. В. Терещенко. – Київ, 2012. – 650 с.

8. Рихтер Д. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. / Д. Рихтер. – 4–е изд. – Москва, 2017. – 896 с.

9. Bura J. Construct 2 Game Development by Example/ John Bura. – England, UK, 2014. – 230 pages.

10. Єлісеєнко О. Є. Порівняльний аналіз сучасних гральних рушіїв / О.Є. Єлісеєнко – Вінниця: ВНТУ, 2016. – 55 с.

11. Кроністер Дж. Blender – Базовий матеріал. 4–те видання : пер. з англ. / Дж. Кроністер. – Київ, 2015. – 588 с.

12. Habgood J. The Game Maker’s Apprentice. /J. Habgood. – USA, 2006. – 334 pages.

13. Романюк О. Н. Організація баз даних і знань. / О.Н. Романюк, Т.О. Савчук // Навчальний посібник. – Вінниця: УНІВЕРСУМ – Вінниця. – 2003. – 123 с.

14. Яворович О.Ю. Unity3d як універсальний інструмент розробки ігрових додатків / О.Ю. Яворович – Вінниця: ВНТУ, 2016. – 120 с.

15. Goldstone W. Unity Game Development Essentials. / W. Goldstone. – Birmingham, UK, 2009. – 894 pages.

16. Степанченко И.В. Методы тестирования программного обеспечения: Учеб. пособие / И.В. Степанчеко. – ВолгГТУ. – Волгоград, 2006. – 74 с.

17. Бейзер Б. Тестирование чёрного ящика. Технологии функционального тестирования программного обеспечения и систем. / Б. Бейзер – СПб.: Питер, 2004. – 320 с.

18. Microsoft. Центр загрузки. Microsoft Visual Studio Learning Pack 2.0 [Електронний ресурс] – Режим доступу: http://www.microsoft.com/downloads/details.aspx?displaylang=ru&FamilyID=0ce3cbbd-7fc7-410b-8c2c-e18d1c60a6cd

19. Башле, П. Н. Інформаційна безпека та захист інформації [Електронний ресурс]: Підручник / П. Н. Башле, А. В. Бабаш, Е. К. Баранова. - М .: РІОР, 2013. - 222 с.

20. Проектування інформаційних систем: навч. посібник / Н. з.Ємельянова, Т. л. Партика, І. і. Попов. - М .: Форум, 2009. - 432 с.

21. Проектування інформаційних систем: Навчальний посібник / Н. н. Заботіна. - М .: НДЦ Инфра-М, 2013. - 331 с.

22. Савчук Т.О., В.Б.Мокін, М.П.Боцула та ін. Розробка технічних вимог до системи передачі й обробки інформації та підтримки прийняття рішень за даними автоматичної інформаційно-вимірювальної системи «Прикарпаття» Звіт

про НДР, ВНТУ. – 2822 (№ ДР 0109U006542) – Інв. № 0209U010308. – К., 2009. – 105с.

23. Проектування інформаційних систем: навч. посібник / Н. з. Ємельянова, Т. л. Партика, І. і. Попов. - М .: Форум, 2009. - 432 с.

24. Скотт б., Нейл т. Проектування веб-інтерфейсів. - Пров. З англ. - Спб .: символ-плюс, 2010. - 352 с.

25. React [Електронний ресурс] – Режим доступу: https://reactjs.org/

26. Назаров, С. в. Операційні системи. Практикум: навч. посібник.-М. КноРус, 2012-376с

27. Жук А. п. І ін. Захист інформації: навч. посібник.- М .: ИНФРА-М, 2013.-392 с.

28. Костюк, Ю. Л. Основи розробки алгоритмів [Електронний ресурс]: навчальний посібник / Ю. Л. Костюк, І. Л. Фукс. - М.: БИНОМ. Лабораторія знань, 2010. - 286 с. : Ил. ; 60x90 / 16. - (Елективний курс. Інформатика). - Режим доступу: http: // www. znanium.com

29. Soumya Krishnan M. Software Development Risk Aspects and Success Frequency on Spiral and Agile Model. / M. Soumya Krishnan // International Journal of Innovative Research in Computer and Communication Engineering (An ISO 3297: 2007 Certified Organization) Vol. 3, Issue 1, January 2015 pp.301-310

30. Sherman M. Building Secure Software for Mission Critical Systems [Электронный ресурс] – Режим доступа: http://resources.sei.cmu.edu/asset_files/Presentation/ 2017_017_001_495865.pdf

31. Полицын С. А. Подходы к вычислению временных затрат на проекты в сфере разработки программного обеспечения на основе использования прецедентов / С.А. Полицын // Программная инженерия №7 2011 С.9-14

32. Лужецький В.А. Організаційно-правові питання безпеки інформації Концептуальна модель системи інформаційного впливу / В.А. Лужецький // Безпека інформації Ukrainian Scientific Journal of Information Security Том 23, № 1 (2017)