

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «магістр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Моніторинг продуктивності працівників»

Виконав студент групи ПЗм-22-1
КУЗЬМУК Вадим Олександрович

Керівник роботи:
МАРЧУК Дмитро Костянтинович

Консультант роботи:
МАРЧУК Галина Вікторівна

Рецензент:
СУГОНЯК Інна Іванівна

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«18» вересня 2023 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: КУЗЬМУК Вадим Олександрович

Керівник роботи: МАРЧУК Дмитро Костянтинович

Тема роботи: «Моніторинг продуктивності працівників»,

затверджена Наказом закладу вищої освіти від «18» вересня 2023 р. № 479/с

Вихідні дані для роботи: **Предметом дослідження є методи та алгоритми моніторингу продуктивності працівників. Об'єктом дослідження є програма моніторингу продуктивності працівників.**

Консультанти з магістерської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Марчук Г.В.	22.09.2023р.	22.09.2023р.
2	Марчук Г.В.	22.10.2023р.	22.10.2023р.
3	Марчук Г.В.	22.10.2023р.	22.10.2023р.

РЕФЕРАТ

Магістерська кваліфікаційна робота присвячена науковому дослідженню методів та технологій моніторингу продуктивності працівників.

Магістерська кваліфікаційна робота включає у себе програмний комплекс, що складається з серверного API, вебінтерфейсу та пояснювальної записки. Пояснювальна записка складається з вступу, 3-х розділів, висновків, списку використаних джерел із 33 найменувань, 3 додатків та 54 ілюстрацій по тексту, з них 6 оформлені на окремій сторінці. Повний обсяг роботи становить 81 сторінку, з них 64 сторінок основного тексту.

Кваліфікаційна робота складається з трьох розділів. В першому розділі було визначено основні вимоги, проведено аналіз задачі, а також досліджено методи та алгоритми для її вирішення. Також було проведено аналіз програмних аналогів, а також здійснено постановку завдання.

В другому було проведено характеристику джерел даних та здійснено практичну реалізацію алгоритму. Розглянуто серверне API з вебінтерфейсом.

В третьому розділі було проведено опис системи та тестування роботи програмного комплексу.

Висновок містить в собі підсумки проведеного дослідження.

Ключові слова: **МОНІТОРИНГ, ПРОДУКТИВНІСТЬ, АНАЛІЗ, ВЕБДОДАТОК.**

ABSTRACT

The qualification work is devoted to a scientific study of employee productivity monitoring.

The master's thesis includes a software package consisting of a server API, a web interface, and an explanatory note. The explanatory note consists of an introduction, 3 chapters, conclusions, a list of 33 references, 3 appendices, and 54 illustrations in the text, 6 of which are on a separate page. The total volume of the work is 81 pages, including 64 pages of the main text.

The qualification work consists of three sections. In the first chapter, the basic requirements were defined, the problem was analyzed, and methods and algorithms for solving it were investigated. The analysis of software analogs was also carried out, as well as the task statement.

The second section characterizes the data sources and implements the algorithm in practice. A server API with a web interface was considered.

In the third section, we tested the software system.

The conclusion contains the results of the study.

Keywords: MONITORING, PERFORMANCE, ANALYSIS, WEB APPLICATION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ ОБЛАСТІ, ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ.....	9
1.1. Теоретичний огляд.....	9
1.2. Використання методів аналізу продуктивності праці.....	12
1.3. Аналіз задачі, засобів та методів її вирішення.....	17
1.4. Огляд існуючих рішень	18
1.5. Постановка задачі дослідження	25
Висновки до розділу 1	26
РОЗДІЛ 2. ПРАКТИЧНА РЕАЛІЗАЦІЯ ВЕБДОДАТКУ	27
2.1 Аналіз вхідних даних	27
2.2 Проектування та розробка вебдодатку	30
2.3 Розробка API частини додатку.....	40
Висновок до розділу 2.....	49
РОЗДІЛ 3. ОЦІНКА ПРАЦЕЗДАТНОСТІ ВЕБДОДАТКУ	50
3.1 Опис та обґрунтування вибору засобів розробки	50
3.2 Опис інтерфейсу системи.....	56
3.3 Тестування вебдодатку	60
Висновок до розділу 3.....	64
ВИСНОВКИ.....	65
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	67
ДОДАТКИ.....	70

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

KPI – Key Performance Indicators

MySQL – My Structured Query Language

JWT – JSON Web Token

JSON – JavaScript Object Notation

API – Application Programming Interface

CRUD – Create, Read, Update, Delete

ВСТУП

Актуальність. Зростання потреби в ефективному моніторингу продуктивності працівників в сучасних підприємствах обумовлене різними факторами. Зміни в робочому середовищі, такі як дистанційна робота та гібридні робочі моделі, роблять важливим відстеження продуктивності незалежно від місця роботи. На сучасних ринках праці ця тенденція стала необхідною відповіддю на зміни вимог та очікувань працівників. В такому контексті, відстеження продуктивності стає ключовим фактором для ефективного управління та забезпечення стійкості підприємства. Зростання конкуренції на глобальному ринку ставить завдання оптимізації використання ресурсів та збереження конкурентоспроможності.

Такі обставини створюють необхідність у вивченні та розвитку методів та інструментів моніторингу, що допоможе підприємствам оптимізувати робочі процеси, збільшити продуктивність та забезпечити більшу ефективність у використанні робочих ресурсів. Тема "Моніторинг продуктивності працівників" відображає актуальні потреби сучасного бізнесу та результати цього дослідження можуть служити основою для впровадження ефективних методів моніторингу продуктивності працівників на підприємствах різних галузей. Шляхом розробки вебдодатку та системи моніторингу, сподіваючись забезпечити підприємства засобами для оптимізації робочих процесів, збільшення продуктивності працівників та забезпечення ефективного використання робочих ресурсів. Такий підхід не лише сприятиме конкурентоспроможності компаній, але й відкриє нові можливості для управління людським капіталом, підвищуючи ефективність бізнесу в сучасному цифровому середовищі.

Метою роботи є розробка рішення задачі моніторингу продуктивності працівників.

Для досягнення поставленої мети необхідно вирішити такі основні завдання:

- дослідити існуючі методи моніторингу продуктивності працівників;
- зробити обґрунтований вибір інструментарію реалізації програми;
- розробка вебдодатку та налаштування моніторингової системи;
- перевірка вебдодатку з використанням отриманих даних.

Об’єкт дослідження є програма моніторингу продуктивності працівників.

Предмет дослідження є методи та алгоритми моніторингу продуктивності працівників.

Наукова новизна. В даній роботі розглядається інтеграція високоефективних інструментів для автоматизації збору та аналізу даних про продуктивність працівників. Застосування автоматизованих засобів спрощує збір інформації та дозволяє в реальному часі відстежувати виконання завдань.

Практичне значення одержаних результатів. Розробка та впровадження системи моніторингу продуктивності працівників має важливе практичне значення для підприємств та організацій, що здійснюють управління персоналом. Ключові аспекти практичного значення включають:

- Розроблена система сприяє оптимізації робочих процесів на підприємстві. Завдяки автоматизації вимірювань та оцінок продуктивності, можливе ефективне розподілення завдань, ресурсів та забезпечення найбільш ефективного використання робочого часу.
- Розроблена система надає стандартизовані критерії оцінювання, що дозволяє об’єктивно визначати та порівнювати продуктивність працівників.

Апробація результатів роботи. Матеріали кваліфікаційної роботи доповідалися на III Всеукраїнської науково-практичної інтернет-конференції: «Сучасні комп’ютерні та інформаційні системи і технології» 12-19 грудня 2022р [33, 34].

РОЗДІЛ 1. АНАЛІЗ ОБЛАСТІ, ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ

1.1. Теоретичний огляд

Моніторинг продуктивності працівників є ключовим елементом сучасного управління та організаційного розвитку [1].

Однією з головних переваг моніторингу продуктивності є здатність підвищити ефективність та продуктивність організації. Шляхом систематичного відстеження та оцінки робочих процесів та виконаної роботи працівників, керівництво може виявити області, де можливі покращення. Приймаючи відповідні заходи на основі аналізу даних моніторингу, організація може підвищити продуктивність, знизити витрати та підвищити ефективність роботи.

Моніторинг продуктивності дозволяє організаціям планувати ресурси більш обґрунтовано. Відстеження робочого часу та завдань працівників допомагає визначити найбільш завантажені та менш завантажені періоди роботи. На основі цієї інформації керівництво може розподілити завдання більш ефективно та розпланувати ресурси так, щоб максимізувати продуктивність.

Моніторинг продуктивності також сприяє забезпеченню високої якості виконаної роботи. Оцінка результатів та якості виконання завдань дозволяє виявити потенційні проблеми та недоліки у процесах. Це допомагає підвищити якість продукції або послуг, а також вчасно доставляти їх клієнтам.

Також моніторинг допомагає керівництву виявляти та розвивати потенціал працівників. Шляхом аналізу даних про робочий процес та результати праці, можна виявити сильні та слабкі сторони працівників і підтримувати їх у розвитку найкращих можливостей.

І нарешті моніторинг допомагає виявляти недосягненні ресурси та неефективні процеси. Це дає можливість оптимізувати робочий потік, зменшити затрати та підвищити вигідність діяльності організації.

Ці аспекти вказують на важливість моніторингу продуктивності для сучасних організацій.

Також у цьому розділі буде розглянуто методи оцінки продуктивності.

1. Ключові показники ефективності (KPI) [20].

KPI (від англійського терміна Key Performance Indicators, ключові показники ефективності) є найбільш поширеним методом оцінки продуктивності в організаціях. Вони представляють собою конкретні числові метрики, які вимірюють результати та продуктивність працівників. Класичні приклади KPI включають обсяг виробленої продукції, прибуток, відсоток завершених проектів у строк тощо.

Основні різновиди KPI такі [18]:

- проектні (результативність проекту, очікування менеджера проекту);
- цільові (індикатори наближення до мети або відповідності поставленим завданням);
- процесні (визначення економічної ефективності роботи).

Крім того, в стратегічному управлінні KPI поділяють на:

- запізнілі — відображають результати діяльності після закінчення звітного періоду. Зазвичай це фінансові та інші ресурсні показники;
- випереджальні — дозволяють управляти роботою всередині звітного періоду, досягаючи заданих результатів до моменту його завершення. Ці показники дозволяють в динаміці аналізувати поточну діяльність підрозділів і компанії загалом, паралельно прогнозуючи, які грошові потоки можуть бути в майбутньому, зміну якості процесів і продукції, ступінь задоволеності замовників тощо.

В управлінні бізнес-процесами (BPM) використовуються такі індикатори [6]:

- KPI результату — дозволяє визначити, який результат і в якому обсязі проведений;
- KPI витрат — дозволяє визначити обсяг витрачених ресурсів;

- КРІ ефективності — характеризує співвідношення отриманого результату до витрат ресурсів (таких, як ROI, цільова вартість одного продажу);
- КРІ продуктивності — характеризує співвідношення отриманих результатів до витраченого часу (наприклад, кількість продажів за місяць);
- КРІ функціонування — дозволяє оцінити відповідність виконуваного бізнес-процесу необхідному алгоритму робіт з його виконання (визначається, наскільки повно і чітко виконані всі вимоги).

Оцінка КРІ дозволяє організації точно вимірювати продуктивність та встановлювати чіткі цілі для працівників. Проте важливо враховувати, що КРІ мають бути вибрані обдуманно і відображати стратегічні цілі організації, а не просто кількісні показники.

2. Збалансована система показників (Balanced Scorecard)

Збалансована система показників включає в себе більш широкий спектр показників, які оцінюють різні аспекти продуктивності та успіху організації. Вона зазвичай включає чотири основні категорії [21]:

- Фінансові показники. Ці показники відображають фінансовий стан організації, такі як прибуток, оборот, рентабельність.
- Клієнтські показники. Ці показники вимірюють задоволеність клієнтів, рівень обслуговування та інші аспекти взаємодії з клієнтами.
- Навчальні показники. Ці показники вказують на розвиток та навчання працівників, їхні навички та здібності.

Збалансована система показників допомагає оцінювати продуктивність з різних перспектив та враховувати більше аспектів організаційного успіху [19].

3. Методика "360 градусів" (360-Degree Feedback) [22].

Методика "360 градусів" передбачає отримання оцінок та відгуків від різних джерел щодо продуктивності працівника. Зазвичай ці джерела включають самого працівника, його керівника, колег, підлеглих та інших, які працюють з ним.

Отримані оцінки допомагають створити більш повний образ продуктивності працівника і виявити його сильні та слабкі сторони. Використання цієї методики може покращити комунікацію та співпрацю в організації.

4. Спрощені методи оцінки продуктивності.

Деякі організації можуть використовувати менш складені методи оцінки продуктивності. Наприклад, метод "Менеджер-підлеглий" включає щорічні розмови між керівником і підлеглим, під час яких обговорюються досягнення, цілі та плани на майбутнє. Цей метод може бути ефективним для менших організацій або тих, де складні програмні засоби не є необхідними.

1.2. Використання методів аналізу продуктивності праці

1. Метод кластеризації

Метод кластеризації є потужним інструментом для аналізу продуктивності працівників, оскільки він дозволяє систематизувати та групувати працівників відповідно до різноманітних характеристик. Це полегшує процес аналізу та допомагає виявити ключові взаємозв'язки між характеристиками та рівнем продуктивності [2].

Можливим способом використання цього методу під час аналізу продуктивності працівників може бути:

- Персоналізований підхід до управління.

Кластеризація дозволяє створювати персоналізовані стратегії управління для різних груп працівників. Наприклад, для одних можуть бути ефективні короткі перерви між робочими задачами, тоді як іншим може бути корисною довша концентрація на одній задачі [3].

- Збільшення мотивації працівників.

Групування працівників за спільними характеристиками може стимулювати змагальність та спільну роботу. Продуктивність може підвищитися, коли працівники бачать, що їхні результати порівнюються зі стандартами та іншими групами працівників.

- Оптимізація навчання та розвитку.

Кластеризація допомагає ідентифікувати, які групи працівників можуть вигідно користуватися спеціалізованими навчальними програмами. Наприклад, для працівників зі схожими навичками може бути ефективним спеціалізоване навчання, спрямоване на покращення їх продуктивності.

- Аналіз ризиків та управління працівниками.

За допомогою кластеризації можна виявити потенційні ризики, пов'язані з продуктивністю працівників. Наприклад, якщо певна група має високий ризик виснаження або низьку мотивацію, можна прийняти заходи для їх попередження та покращення.

- Сегментація працівників.

У випадку вебсервісів чи програм для відстеження продуктивності, метод кластеризації може допомогти визначити сегменти працівників з подібними потребами та стилями роботи.

Також можемо розглянути ситуацію, коли у великій компанії є різні відділи, і хочете дізнатися, які групи працівників найбільш продуктивні. За допомогою методу кластеризації, можна групувати працівників відповідно до таких характеристик, як роль, відділ, досвід тощо. Потім визначити середню продуктивність для кожного кластеру. Це допоможе ідентифікувати, які групи працівників є найбільш продуктивними. І саме така інформація може допомогти керівництву компанії зробити певні висновки, щодо своїх працівників.

Отже, метод кластеризації в контексті аналізу продуктивності працівників відкриває широкі можливості для управління та оптимізації робочих процесів в організації. Він дозволяє створювати індивідуальні підходи до кожної групи працівників та забезпечує більшу ефективність та задоволеність на робочому місці.

2. Метод обробки часових рядів

Метод обробки часових рядів є іншим потужним інструментом для аналізу продуктивності працівників, особливо в контексті вимірювання

робочого часу та розподілу завдань. Цей метод полягає у зборі та аналізі даних про робочий час працівників з метою виявлення шаблонів, трендів та аномалій [4].

Такий метод можна використати наступним чином:

1) Збір і агрегація даних про робочий час.

Спершу потрібно зібрати дані про робочий час працівників. Це може включати в себе інформацію про час початку та закінчення робочого дня, перерви, робочі години, а також додаткові дані, як-то проєкти, завдання, чи категорії робіт.

2) Побудова часового ряду.

Отримані дані можна використовувати для створення часового ряду, де час є незалежною змінною, а обсяг виконаної роботи (наприклад, кількість завдань, витрачений час) - залежною змінною. Цей часовий ряд може відображати, як змінюється продуктивність працівника протягом робочого дня, тижня або місяця [5].

3) Аналіз часових рядів.

За допомогою методів обробки часових рядів можна визначити різні аспекти продуктивності, такі як:

- Тренди. Можна виявити загальний тренд продуктивності працівника - чи є зростання або спад у виконанні завдань протягом певного часу.

- Сезонність. Деякі працівники можуть мати сезонні зміни в продуктивності, які пов'язані з певними періодами року або подіями.

- Аномалії. Можна виявити аномалії, такі як дуже ефективний день, коли працівник виконав надзвичайно багато завдань, або день з низькою продуктивністю, коли нічого не виконано.

4) Прогнозування продуктивності.

На основі аналізу часового ряду можна розробити моделі прогнозування, які допоможуть передбачити майбутню продуктивність працівника. Наприклад, можна передбачити, скільки завдань працівник

зможє виконати на наступному робочому дні, враховуючи його історичну продуктивність та інші фактори.

5) Оптимізація робочих процесів.

Аналіз часових рядів дозволяє ідентифікувати слабкі місця та можливості для оптимізації робочих процесів. Наприклад, якщо аналіз показує, що багато часу втрачається на перерви між завданнями, це може свідчити про неефективність та створювати підстави для змін.

Прикладом використання методу обробки часових рядів може бути вебсервіс або додаток, які дозволяють працівникам відстежувати свій робочий час, реєструвати виконані завдання та аналізувати свою продуктивність протягом робочого дня. Такий інструмент надає можливість краще розуміти та оптимізувати власний робочий процес. Саме такий вебсервіс буде розроблено під час дипломної роботи.

3. Метод визначення ключових показників ефективності

Більше детально розглянемо метрики, на основі яких можна визначити рівень продуктивності [6].

1) Вибір важливих метрик.

Перший крок у визначенні KPI - це вибір ключових метрик, які найкраще відображають продуктивність працівників. Наприклад, для робітників у виробничому секторі KPI може включати в себе кількість виготовлених одиниць товару за годину. Для робітників у IT-сфері - швидкість вирішення завдань або якість коду.

2) Визначення цілей та стандартів.

Для кожної вибраної KPI потрібно визначити цілі та стандарти. Наприклад, ціль може бути такою: "Кількість виготовлених одиниць товару повинна збільшитися на 10% протягом наступного кварталу." Стандарт - це показник, який служить порівняльною точкою для вимірювання прогресу.

3) Моніторинг та звітність.

Важливим етапом є постійний моніторинг KPI та створення звітів на основі них. Це допомагає відстежувати, чи досягнуті поставлені цілі та допомагає вчасно виявляти відхилення.

4) Внесення коректив.

Якщо KPI не виконуються, необхідно вжити заходів для виправлення ситуації. Наприклад, якщо кількість виготовлених одиниць товару не досягає цілі, можна розглянути оптимізацію виробничих процесів, навчання працівників або впровадження нового обладнання.

Як приклад використання такого методу може бути наступне:

- KPI: Кількість виконаних завдань за один день.
- Ціль: Збільшити продуктивність на 10% за квартал.
- Стандарт: Середній обсяг виконання завдань минулого кварталу.

4. Використання методів.

В рамках дослідження були розглянуті та проаналізовані різні методи, які допомагають визначити та підвищити продуктивність працівників. До них належать метод кластеризації, метод обробки часових рядів і метод визначення ключових показників ефективності (KPI).

Метод кластеризації надає можливість групувати працівників в класи на основі подібних характеристик, що допомагає створювати персоналізовані стратегії управління, збільшувати мотивацію, оптимізовувати навчання та розвиток, а також аналізувати ризики та керувати працівниками.

Метод обробки часових рядів дозволяє аналізувати робочий час працівників для виявлення шаблонів, трендів та аномалій. Він може бути використаний для побудови часових рядів, прогнозування продуктивності та оптимізації робочих процесів.

Метод визначення KPI дозволяє вибрати важливі метрики, визначити цілі та стандарти, моніторити та звітувати про них, а також внести корективи у випадку не досягнення цілей.

У проєкті планується використовувати всі методи для створення вебсервісу, який дозволить працівникам відстежувати та аналізувати свою

продуктивність, з метою покращення робочих процесів та досягнення високих результатів. Кластеризація допоможе групувати користувачів на основі їхніх робочих характеристик, метод обробки часових рядів надасть можливість аналізувати робочий час та шукати оптимальні рішення, а метод визначення KPI допоможе створити систему метрик для вимірювання та контролю продуктивності.

Такий комплексний підхід до аналізу та управління продуктивністю працівників дозволить підвищити ефективність робочих процесів та сприяти досягненню поставлених цілей.

1.3. Аналіз задачі, засобів та методів її вирішення

В даному аналізі розглянемо проблематику, методи та засоби вирішення завдань, пов'язаних з аналізом продуктивності праці. Для прикладу у роботі буде використовуватися компанія, яка займається створеннями анімацій та відео.

Проблематика.

Планування виробництва: Анімаційна студія повинна планувати свою роботу, враховуючи ресурси, строкові рамки та вимоги замовників. Прогнозування попиту на анімаційні фільми, серіали або рекламні ролики є важливим етапом планування виробництва.

Бюджетування: Анімаційна студія повинна ефективно використовувати свій бюджет, який включає витрати на розробку, виробництво та маркетинг. Прогнозування витрат та доходів допомагає встановити бюджетні обмеження та забезпечити фінансову стабільність студії.

Оптимізація процесу розробки: Анімаційний процес складається з багатьох етапів, включаючи сценарій, розробку персонажів, анімацію та постпродакшн. Аналіз та моделювання даних допомагає ідентифікувати ефективні методи та засоби для покращення робочих процесів та зниження часу виробництва.

Методи вирішення задачі.

Аналіз історичних даних: Для прогнозування майбутнього попиту та доходів студії можна використовувати дані про попередні проекти. Це дозволить виявити тенденції та залежності між різними факторами, які впливають на успішність проектів.

Математичне моделювання: Використання математичних моделей дозволяє прогнозувати попит на анімаційні продукти та визначати оптимальні стратегії виробництва. Наприклад, моделі часових рядів можуть допомогти передбачити темпи зростання попиту на ринку анімації.

Методи машинного навчання: Застосування алгоритмів машинного навчання, таких як нейронні мережі, може допомогти анімаційній студії аналізувати великі обсяги даних, включаючи відгуки глядачів, соціальні медіа та маркетингові дані. Це дозволяє прогнозувати попит, розуміти вподобання аудиторії та адаптувати стратегії виробництва.

Засоби вирішення задачі.

Системи аналітики даних. Використання спеціалізованих програмних засобів для аналізу та візуалізації даних, таких як Python-бібліотеки (наприклад, `pandas` і `matplotlib`), дозволяє анімаційній студії виконувати статистичний аналіз даних та будувати моделі прогнозування.

1.4. Огляд існуючих рішень

Розглянемо деякі програмні засоби та вебсервіси, які вже використовуються для моніторингу продуктивності працівників. Цей огляд допоможе нам краще зрозуміти, які можливості вже доступні та як проєкт може відрізнятись від існуючих рішень.

1) Workpuls [\[офіційний сайт\]](#)

Workpuls - це програма для моніторингу продуктивності працівників у сфері ІТ та не тільки. Вона надає інструменти для відстеження робочого часу, моніторингу використання програм, вебсайтів та інших аспектів робочої активності [23].

Переваги: Workpuls дозволяє детально вивчати, як працівники витрачають робочий час, та надає можливість оптимізації процесів у сфері ІТ. Це допомагає підвищити продуктивність та забезпечує відповідність стандартам безпеки.

Недоліки: Важливо бути обережним у використанні таких програм, оскільки вони можуть викликати питання конфіденційності та приватності працівників.

Приклад використання Workpuls може бути таким: Компанія, яка розробляє програмне забезпечення, використовує Workpuls для моніторингу часу, витраченого розробниками на різні завдання та проекти. Це допомагає керівництву отримувати інформацію щодо ефективності розробки, вчасності завершення проектів та інших аспектів продуктивності. Подивитися, як виглядає цей вебдодаток, можна на рисунку 1.1.

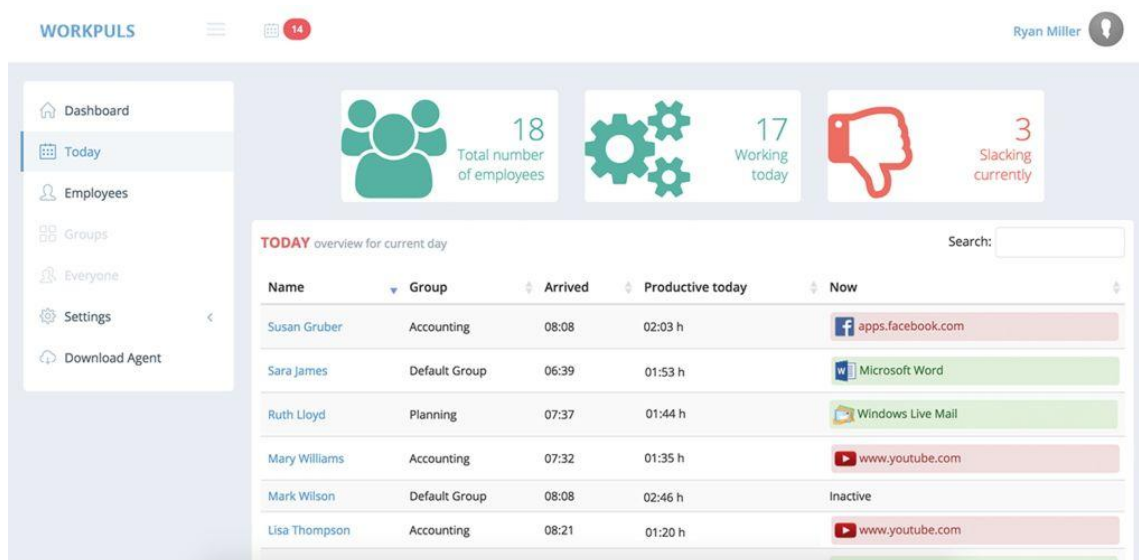


Рис. 1.1. Workpuls

2) HubStaff [\[офіційний сайт\]](#)

HubStaff - це програма та вебсервіс для моніторингу робочого часу та продуктивності співробітників. Вона спеціалізується на наданні інструментів для відстеження часу та контролю продуктивності у реальному часі [24].

Переваги: HubStaff надає численні можливості для кращого аналізу продуктивності співробітників. Основні переваги включають:

- Відстеження часу:

Програма автоматично реєструє час, який працівники витрачають на проекти та завдання.

- Моніторинг активності:

HubStaff відстежує робочу активність на комп'ютерах співробітників, включаючи використання програм та вебсайтів.

- Прив'язка до проектів:

Проекти та завдання можуть бути докладно відстежені, і час може бути прив'язаний до конкретних проектів або завдань.

Недоліки: Незважаючи на свої переваги, HubStaff має деякі недоліки:

- Конфіденційність та приватність:

Система моніторингу може порушити конфіденційність співробітників та викликати питання приватності.

- Не відповідає для всіх видів робіт:

Цей тип програми може бути непридатним для креативних або некомерційних робіт, де продуктивність не завжди вимірюється часом та активністю на комп'ютері.

Приклад використання: Компанія в сфері веброзробки використовує HubStaff для відстеження часу та продуктивності своїх розробників. Програма допомагає визначити, скільки часу витрачається на окремі проекти, і допомагає виробляти звіти для клієнтів щодо витрат робочого часу та вартості розробки. Активний моніторинг також допомагає виявити прості та невикористаний час, що сприяє оптимізації робочих процесів. На рисунку 1.2 зображено вебдодаток.

Job site reports

Organization: Bolivar G

ME TEAM

Thu, Nov 22, 2018

Filters

Export

Bolivar Group America - New York

Group by: Date

Member	Job site	Enter	Exit	Time
Thu, Nov 22, 2018				
Brandon Vargas	New York #1	4:54 AM	9:00 AM	4:06
		10:00 AM	12:00 PM	2:00
	New York #3	1:00 PM	2:00 PM	1:00
		2:20 PM	3:20 PM	1:00
	New York #7	4:00 PM	4:30 PM	0:30
Total				8:36

Рис. 1.2. HubStaff

3) Teramind [\[офіційний сайт\]](#)

Teramind - це інтегроване рішення для моніторингу активності та продуктивності працівників на комп'ютерах. Воно надає детальні інструменти для відстеження та аналізу активності на комп'ютерах, включаючи моніторинг використання програм, вебсайтів, інтернет-пошти та інших додатків. Teramind також вміє відслідковувати клавіші, які натискаються, та дії на екрані, що робить його потужним інструментом для контролю кібербезпеки та виявлення внутрішніх загроз [25].

Переваги: Teramind має декілька значущих переваг:

- Деталізований моніторинг:

Великий обсяг даних та деталізація дозволяють виявляти відхилення та порушення політик безпеки.

- Контроль доступу:

Teramind надає можливість обмежити доступ до конфіденційної інформації, що підвищує рівень кібербезпеки.

- Аналітика та звіти:

Засіб надає інструменти для аналізу даних та створення звітів, що корисно для управління та прийняття рішень.

Недоліки: Існують деякі недоліки Teramind:

- Складність налаштування:

Вимагає складного налаштування, що може бути витратним за часом.

- Опір співробітників:

Інтенсивний моніторинг активності може викликати опір серед працівників і порушити конфіденційність.

Прикладом використання є: Компанія в сфері фінансів використовує Teramind для контролю за використанням фінансових додатків та доступу до конфіденційних фінансових даних. Програма дозволяє виявити незвичайну активність, таку як спроби незаконного доступу, і надає адміністраторам можливість реагувати негайно на потенційні загрози для кібербезпеки. Інтерфейс додатку можна побачити на рисунку 1.3.

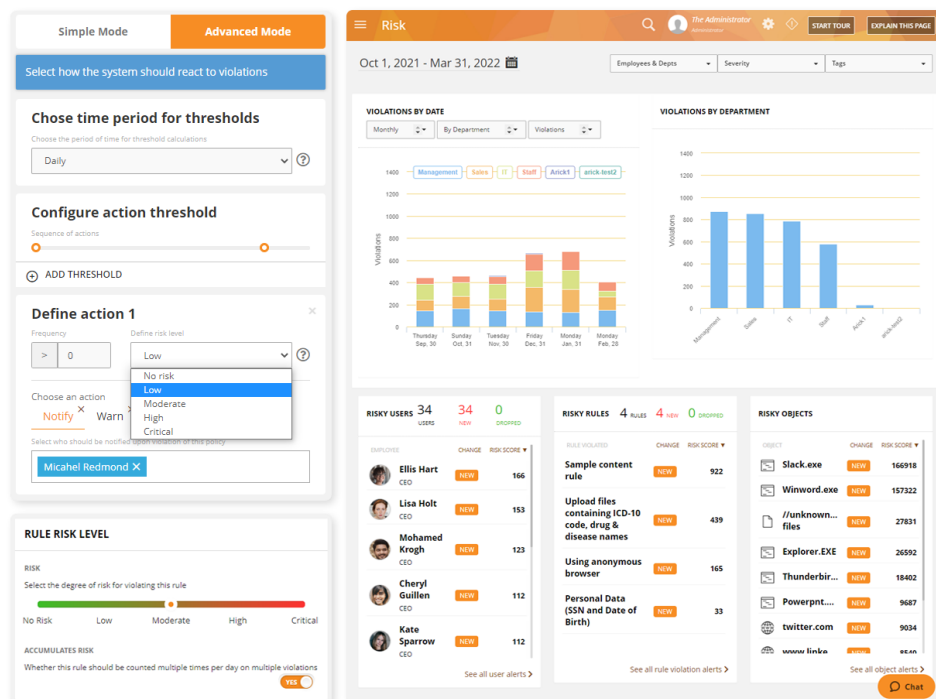


Рис. 1.3. Teramind

- 4) Yaware [\[офіційний сайт\]](#)

Yaware - це програма для моніторингу продуктивності та обліку робочого часу співробітників. Вона дозволяє відстежувати, як працівники витрачають свій робочий час, які додатки та вебсайти вони використовують, і генерує звіти для аналізу продуктивності [26].

Переваги: Yaware має декілька переваг:

- Активний моніторинг.

Програма активно відстежує використання комп'ютера, фіксуючи час, витрачений на різні завдання.

- Аналітика продуктивності.

Yaware генерує деталізовані звіти про продуктивність, що дозволяє керівництву аналізувати та покращувати ефективність працівників.

- Контроль вебсайтів та додатків.

Програма дозволяє обмежити доступ до небажаних вебсайтів та додатків, що поліпшує фокус та знижує ризик відволікання.

Недоліки: Декілька недоліків Yaware включають:

- Питання конфіденційності.

Інтенсивний моніторинг може викликати опір серед працівників та порушити конфіденційність.

- Сумісність з мобільними пристроями.

Деякі функції Yaware можуть бути обмежені на мобільних пристроях.

Приклад такого додатку можна побачити на рисунку 1.4.

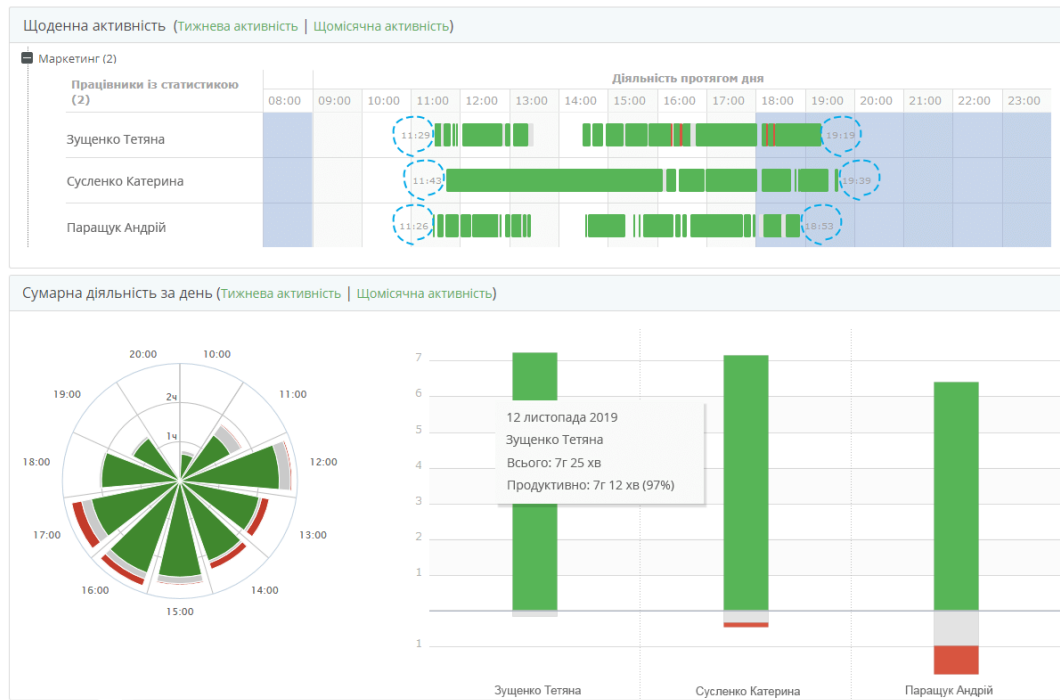


Рис. 1.4. Yaware

Приклад використання може бути наступне: ІТ-компанія використовує Yaware для відстеження часу, витраченого розробниками на різні завдання та програмування. Програма допомагає виявити, які завдання вимагають більше часу, та оптимізувати робочий процес для підвищення продуктивності.

Під час аналізу існуючих рішень для моніторингу продуктивності працівників, було виявлено різноманітні програми та вебсервіси, кожен із яких має свої переваги та недоліки. У порівнянні із проектом, ціль полягатиме в поєднанні кращих аспектів цих існуючих рішень. Буде спрямовано фокус на розробку засобу моніторингу, який надає можливість ведення робочого часу, відстеження активності та аналізу продуктивності, але при цьому залишається простим у використанні. Мета - включити функції, які дозволять пристосовувати рівень моніторингу до потреб кожного користувача та забезпечити конфіденційність даних. Отже, цей проєкт спрямований на поєднання кращих практик з існуючих рішень і розробку інноваційного засобу моніторингу продуктивності працівників, який буде відповідати сучасним вимогам та потребам користувачів.

1.5. Постановка задачі дослідження

Для досягнення поставленої мети розробки системи моніторингу продуктивності працівників та впровадження інноваційних підходів до управління персоналом, в рамках роботи формалізується модель та постановка завдань, які необхідно вирішити:

Модель системи моніторингу продуктивності:

- 1) Виявлення ключових показників продуктивності.

Визначення переліку ключових показників продуктивності, які будуть враховуватися системою. Це включає фактори, такі як виконані завдання, тривалість роботи, ефективність розв'язання завдань тощо.

- 2) Створення метрик та критеріїв оцінювання:

Розробка критеріїв оцінювання продуктивності, що враховуватимуться при автоматизованому процесі визначення ефективності працівників. Метрики повинні бути об'єктивними та відображати реальність роботи.

- 3) Аналіз факторів впливу на продуктивність.

Вивчення та ідентифікація факторів, які можуть впливати на продуктивність працівників. Це може включати аналіз внутрішніх та зовнішніх факторів, таких як тип завдань, робоче середовище, індивідуальні навички тощо.

Завдання для вирішення:

- 4) Розробка алгоритмів моніторингу.

Створення алгоритмів, які забезпечать ефективний та точний збір та аналіз даних про продуктивність працівників. Реалізація алгоритмів повинна враховувати різні види завдань та їх особливості.

- 5) Розробка модулю оцінки продуктивності.

Реалізація програмного модулю, який автоматично оцінюватиме продуктивність працівників на основі визначених метрик та критеріїв. Модуль повинен бути легко інтегрований у вже існуючі системи управління.

- 6) Забезпечення безпеки та конфіденційності.

Розробка заходів забезпечення безпеки та конфіденційності даних, зібраних системою. Це включає в себе заходи для запобігання несанкціонованому доступу до інформації та збереження приватності працівників.

Висновки до розділу 1

Перший розділ даної роботи присвячений аналізу області, проблематики, методів та засобів вирішення поставленої задачі. Здійснений теоретичний огляд дозволяє отримати уявлення про основні аспекти досліджуваної теми. Використання методів аналізу продуктивності праці сприяло глибшому розумінню факторів, що впливають на вирішення поставленої проблеми.

Огляд існуючих рішень у сфері досліджуваної проблеми дозволяє виявити прогалини та напрямки подальших досліджень.

Отримані результати аналізу вказують на актуальність теми та підтверджують необхідність подальших наукових досліджень у даній області. Постановка задачі дослідження визначає конкретні цілі та завдання, спрямовані на вирішення виявлених проблем.

Розділ 1 є важливим етапом у структурованні дослідження, оскільки він визначає основні параметри, на яких буде зосереджено увагу у подальших розділах.

РОЗДІЛ 2. ПРАКТИЧНА РЕАЛІЗАЦІЯ ВЕБДОДАТКУ

2.1 Аналіз вхідних даних

Дані для даного проекту були запозичені з відкритих джерел інтернету, у компанії, що займається анімаціями. В цілях збереження особистої інформації, було змінено усі імена, фамілії, назви проектів та дати.

Цей набір даних включає в себе більше ніж 1500 записів, що збиралися кожного дня впродовж року. Але в роботі буде використовуватися набагато менше записів. Усі дані проекту зберігаються у СУБД MySQL.

MySQL є однією з найпопулярніших відкритих систем управління базами даних (СУБД) [29]. Вибір MySQL для використання в проекті вебсервісу моніторингу продуктивності працівників обґрунтований з ряду факторів:

Відкритий джерела: MySQL є відкритою системою, що означає, що вона є безкоштовною та доступною для використання, а також має активну спільноту розробників, яка надає підтримку.

Широкі можливості: MySQL має багато можливостей для роботи з даними, включаючи підтримку структурованих та неструктурованих даних, можливість використовувати SQL-запити та інші мови програмування.

Висока швидкість: MySQL володіє високою швидкістю операцій з базами даних, що важливо для збереження та обробки великої кількості даних про працівників та їх продуктивність.

Масштабованість: MySQL легко масштабується вгору, що означає, що можна почати з невеликої бази даних і розширювати її з часом зростанням обсягу даних.

Застосування СУБД MySQL в проекті:

MySQL використовуватиметься для зберігання та керування даними про працівників, їх робочий час, продуктивність та іншою інформацією. Ця система допоможе структуровано зберігати дані та забезпечити швидкий доступ до них.

Маючи на увазі всі ці переваги, вибір MySQL допоможе побудувати надійну та ефективну систему для зберігання даних та обробки інформації про продуктивність працівників. На рисунку 2.1 представлені вхідні дані, у Додатку Б будуть наведені усі дані.

id (DC2Type:uuid)	title	seconds	user_id (DC2Type:uuid)	hours	created
0x162289fb96a311ee8e5c0242ac164b08	Project D	19	0x1ee5f7a73d6a647e9eae71d3f128f71	2.00	2023-12-07 00:27:46
0x161e9d6b96a311ee8e5c0242ac164b08	Project D	24	0x1ee5f79b7ef269848a5e4bf8d0869cd1	5.00	2023-12-07 01:04:04
0x1a2345af96a311ee8e5c0242ac164b08	Project A	12	0x1ee5f76d6814606a9a12f76f0a9b1d9f	2.50	2023-12-07 02:59:45
0x149cf55796a311ee8e5c0242ac164b08	Project D	21	0x1ee5f7a73d6a647e9eae71d3f128f71	4.00	2023-12-07 03:25:45
0x12cc3fea96a311ee8e5c0242ac164b08	Project C	39	0x1ee56507b58f6bf8be34e1d42bcd65f	3.00	2023-12-07 05:20:52
0x97c4b07e96a211ee8e5c0242ac164b08	Project D	29	0x1ee5f7808d296c5ca54bcb02cf35e170	4.00	2023-12-07 08:16:45
0x97ca9dbd96a211ee8e5c0242ac164b08	Project B	7	0x1ee565091ea0679abdead53a21dda506	5.00	2023-12-07 11:08:09
0x12db479796a311ee8e5c0242ac164b08	Project C	13	0x1ee56507b58f6bf8be34e1d42bcd65f	2.50	2023-12-07 13:13:37
0x161663d996a311ee8e5c0242ac164b08	Project A	12	0x1ee564f533d568b8ba9d670bb1ff19b5	2.50	2023-12-07 14:09:16
0x1a2345f996a311ee8e5c0242ac164b08	Project C	12	0x1ee5650966c360cc993661cc32f3adec	3.00	2023-12-07 17:29:15
0x1a2345d996a311ee8e5c0242ac164b08	Project C	30	0x1ee5650966c360cc993661cc32f3adec	6.00	2023-12-07 18:04:15
0x1a23459d96a311ee8e5c0242ac164b08	Project C	13	0x1ee5650966c360cc993661cc32f3adec	3.00	2023-12-07 20:49:15
0x1a2345c496a311ee8e5c0242ac164b08	Project D	21	0x1ee5f79b7ef269848a5e4bf8d0869cd1	4.00	2023-12-07 22:32:00
0x161e9b5396a311ee8e5c0242ac164b08	Project B	28	0x1ee564f768c06b48939f377d2f205189	4.00	2023-12-08 00:47:31
0x162285b796a311ee8e5c0242ac164b08	Project A	12	0x1ee56508ea066b74aa347d4680b7876d	4.00	2023-12-08 03:16:23

Рис. 2.1. Перегляд даних таблиці work

Розглянемо більш детально дані наведені у таблиці:

Стовпець «id» - зашифрована версія ідентифікаційного номеру запису.

Стовпець «title» - в ній зберігається назва проекту.

Стовпець «hours» - кількість часу, який робочий витратив за сьогодні.

Стовпець «seconds» - кількість секунд, які робочий виробив за певний період часу.

Стовпець «user_id» - зашифрована версія ідентифікатора користувача, що посилається на іншу таблицю, в якій вже зберігається уся інформація про робочого.

Стовпець «created» - дата та час створення запису.

На рисунку 2.2 можна подивитися тип даних кожного стовпчику.

#	Name	Type
1	<u>id</u>	binary(16)
2	title	varchar(255)
3	seconds	int
4	<u>user_id</u>	binary(16)
5	hours	decimal(7,2)
6	created	varchar(255)

Рис 2.2. Перегляд типу змінних

Так як дана таблиця має зв'язок з іншою таблицею, в якій зберігається уся інформація про користувача (рис.2.3).

id (DC2Type=uuid)	email	roles	password	phone	first_name	last_name	sex	birthday
0x1ee564f533d568b8ba9d670bb1f19b5	doc1@example.com	["ROLE_USER"]	\$2y\$13\$SecTnOgEOrIFQ4lr21yzZe4efDqGAYf6wgFYE2HLgX...	+38(012)-345-67-89	Павло	Дятлов	m	1693515600
0x1ee5f76d6814606a9a12f76f0a9b1d9f	doc10@example.com	["ROLE_USER"]	\$2y\$13\$ESmOj7LKejgstm0cbLxUOWqY3peTKQ39Zfo/RuHFCP...	+38(085)-527-52-55	Мілан	Малець	m	499986000
0x1ee5f780d296c5ca54bcb02cf35e170	doc11@example.com	["ROLE_USER"]	\$2y\$13\$opL6hRvmvWkJOm.1rt10xQCQY8GVj1uMucHcSmCJH...	+38(012)-678-54-52	Юлія	Єфімова	f	644529600
0x1ee5f79b7ef269848a5e4bf8d0869cd1	doc12@example.com	["ROLE_USER"]	\$2y\$13\$SoUVIHPS.uBQUagtbKxUjenCuV.Tuzoq3B1pZDWBe89...	+38(012)-686-45-45	Сергій	Кулик	m	581716800
0x1ee57a73d6a647e9eaed71d3f128f71	doc13@example.com	["ROLE_USER"]	\$2y\$13\$YENNDIzibhsrHUjHBEICgek7v4F8a6uV4gB7C4q9M4o...	+38(012)-876-54-54	Олександр	Коваленко	m	266446800
0x1ee5650966c360cc993661cc32f3adec	doc9@example.com	["ROLE_USER"]	\$2y\$13\$gUjhk2k45XcYR8e6co2QJMqzPaZQsnyM32NzyBngy...	+38(012)-646-82-64	Юлія	Швець	m	123456789
0x1ee565091ea0679abdead53a21dda506	doc8@example.com	["ROLE_USER"]	\$2y\$13\$W4FqYxhvLnzW4jHnxo5au9JkL235hz.7ErjSPcexE...	+38(012)-364-34-62	Сергій	Войцільський	m	123456789
0x1ee56508ea066b74aa347d4680b7876d	doc7@example.com	["ROLE_USER"]	\$2y\$13\$E4zwLEbgWd58AHKkbzCB1OJ0l3bcyULAXmdKH7ywd7...	+38(012)-646-82-49	Роман	Левицький	m	123456789
0x1ee5650886c46fd4a79a9585812ef5d4	doc6@example.com	["ROLE_USER"]	\$2y\$13\$couFTQWCj4o95oKRfTLMOy808rf2jPdibc14csyju...	+38(012)-752-07-43	Олена	Бачинська	m	123456789
0x1ee5650848fe642a9a782f031f27eb16	doc5@example.com	["ROLE_USER"]	\$2y\$13\$Eo2Zyu7ZqdtSjWx5vJkDBOj2hV9sdf.TCjDlNwLf...	+38(012)-782-75-14	Сергій	Одарченко	m	123456789
0x1ee565081bcb67828265f3e6045f1e9	doc4@example.com	["ROLE_USER"]	\$2y\$13\$RrNsrlt6E5t73snQfV.mfHk3cz7EcZDn6QRzf/ZV...	+38(012)-676-42-45	Оксана	Баширова	m	123456789
0x1ee56507b58fbf8be34e1d42bcd65f	doc3@example.com	["ROLE_USER"]	\$2y\$13\$PF0E865Ou/A9qAKrjVbtUzyEvJfQNoI/eGH/cjN2B...	+38(012)-355-69-72	Аркадій	Мітлошук	m	123456789
0x1ee564f768c06b48939f377d2f205189	doc2@example.com	["ROLE_USER"]	\$2y\$13\$Kof93JA6H3rEqRLQ86aOuyopqYLPw6A1lfsveSLY53...	+38(012)-345-67-89	Ірина	Стецюк	m	-103431600

Рис 2.3. Перегляд даних таблиці user

Стовпець «id» - зашифрована версія ідентифікаційного номеру запису.

Стовпець «email» - email користувача було змінено в цілях збереження особистих даних.

Стовпець «roles» - відповідає за роль, надану користувачу під час реєстрації.

Стовпець «password» - зашифрована версія паролю користувача.

Стовпець «phone» - мобільний номер користувача, також змінений в цілях збереження особистих даних.

Стовпець «first_name» - зберігає ім'я користувача.

Стовпець «last_name» - зберігає фамілію користувача.

Стовпець «sex» - зберігає стать користувача.

Стовпець «birthday» - зберігає дату народження користувача у форматі unix timestamp.

На рисунку 2.4 можна подивитися тип даних змінних.

1	<u>id</u>	binary(16)
2	email	varchar(180)
3	roles	json
4	password	varchar(255)
5	phone	varchar(20)
6	first_name	varchar(45)
7	last_name	varchar(45)
8	sex	varchar(45)
9	birthday	bigint

Рис 2.4. Перегляд типу змінних

2.2 Проектування та розробка вебдодатку

Архітектура вебдодатку представляє собою досить складну, але добре організовану систему взаємодії між різними компонентами для створення високоефективного та корисного вебсервісу. Давайте розглянемо деякі ключові аспекти цієї архітектури.

Коли користувач відкриває вебсторінку, сервер, що використовує Symfony, взаємодіє з браузером, відправляючи необхідні дані у відповідь на запит. Вебклієнт, з свого боку, може ініціювати запити до вебресурсів або отримувати різноманітні вебдокументи через цей сервер.

При відкритті сторінок або відправці запитів з вебклієнта, Symfony може виконувати запити до бази даних для отримання або оновлення інформації.

Щоб забезпечити високу ефективність, використовуємо оптимізовані запити та кешування даних, це є деякими з стратегій для зменшення навантаження на базу даних та покращення швидкодії додатку.

Ця взаємодія з базою даних MySQL становить важливий елемент архітектури, дозволяючи зберігати, оновлювати та отримувати дані, необхідні для функціонування вебсервісу [7].

В цій архітектурі Symfony використовується як серверна частина для обробки запитів та відправлення відповідей, а вебклієнт, за який відповідає React, ініціює запити та отримує необхідні дані. Об'єднання Symfony на сервері та React на клієнті в згаданій архітектурі створює потужну платформу для розробки динамічних та інтерактивних вебдодатків.

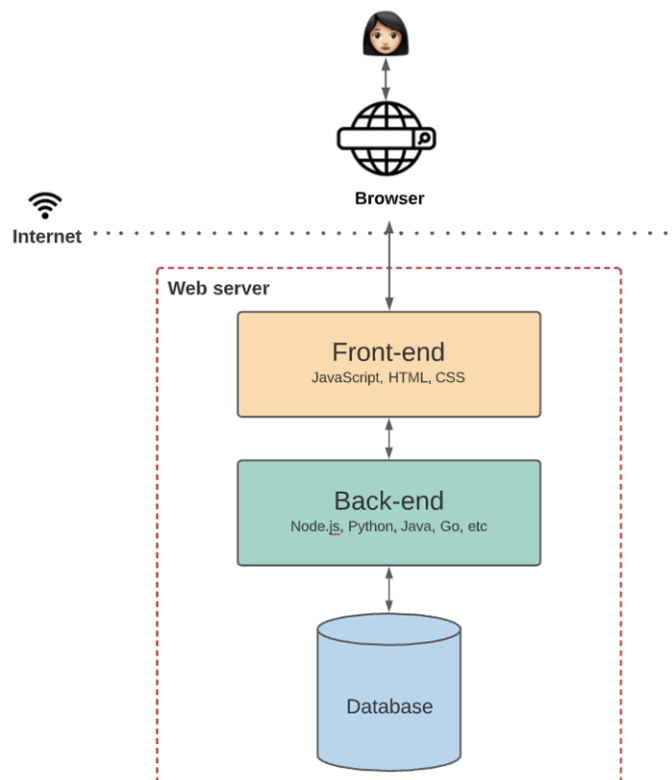


Рис. 2.5. Архітектура вебдодатку

Для більше детального розуміння, що саме потрібно далі робити, для початку потрібно розібратися з ціллю вебдодатку. Для цієї роботи було обрано вебдодаток, який збирає дані від користувачів, що працюють у анімаційній фірмі. Для розробки такого вебдодатку, його спершу потрібно поділити на дві частини [8]:

1) Працівники компанії

Кожного працівника компанії реєструє адміністратор, це зроблено в цілях запобігання витікання інформації за мережі компанії. Після авторизації,

працівник компанії зможе увійти до особистого кабінету, де в свою чергу зможе обрати проект над яким він сьогодні працював, скільки годин він витратив на це та скільки секунд анімації йому вдалось зробити за цей період часу. Заповнивши ці дані він матиме змогу відправити їх. Також має бути вкладка, на якій він зможе подивитися, що саме він відправив.

2) Адміністрація

Усі ці дані, що надійшли від працівників компанії потрапляють до відповідальних за аналіз людей, або прямо кажучи до адміністраторів вебдодатку. В свою чергу вони матимуть змогу переглядати абсолютно всю інформацію про записи, що робились робітниками. Та найголовніше, вони матимуть можливість аналізувати ці дані, тому що вони будуть зображені у вигляді графіків та інших видів зображення інформації.

Тепер отримавши повне розуміння, що саме потрібно зробити, приступимо до створення Frontend частини вебдодатку. Для цього на (рис.2.6) можемо побачити структуру Frontend частини додатку.

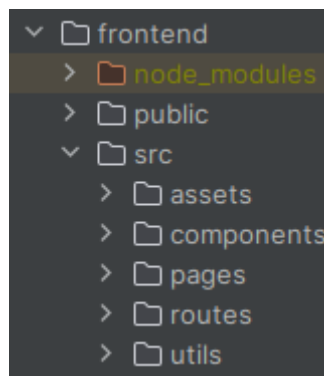


Рис. 2.6. Структура Frontend частини

Папка assets зберігає у собі усі файли зі шрифтами. Папка components зберігає у собі декілька папок, що в свою чергу зберігають у собі елементи, з яких складається сторінка тієї, чи іншої частини сайту, або кабінету користувача чи адміністратора. Це в свою чергу допоможе більш гнучко розробляти сайт, тому що не потрібно буде усюди повторювати однакові елементи, достатньо буде лише імпортувати їх з цієї папки та використовувати. Детальніше про цю папку буде зображено на (рис.2.7). Папка pages зберігає в собі папки, в яких знаходяться файли що відповідають

за окремі сторінки сайту. Папка `routes` зберігає в собі файли, в яких записана усе про структуру цього додатку. Та нарешті папка `utils` зберігає в собі решту дуже важливих файлів, таких як: файл перевірки аутентифікації користувача, файл перевірки наявності токена, файл необхідний для отримання інформації про користувача та інше.

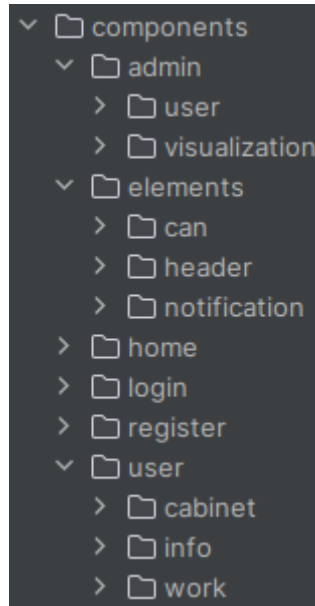


Рис. 2.7. Папка `components`

Одна з функцій яка буде відрізняти цей вебдодаток від багатьох інших, це використання найсучасніших технологій, таких як `React` та `Symfony`.

`React` відіграє ключову роль у вебінтерфейсі додатку. Декларативний підхід до створення компонентів дозволяє нам ефективно керувати станом додатку та легко оновлювати відображення при зміні даних.

Саме тому під час здавалося б звичайного переходу до особистого кабінету, користувача звичайно перенаправляло на іншу сторінку сайту, усі компоненти починають завантажуватися і користувач очікує, поки сторінка збереться до купи. В цьому проекті `React` надає змогу цього уникнути, зробивши тільки ілюзію переходу на іншу сторінку, а насправді, він оновлює поточну сторінку та міняє адрес у строчці пошуку [9]. Це надасть змогу більше ефективно використовувати ресурси комп'ютеру, тому що елементи, які на усіх сторінках однакові, такі як `header` чи `aside` не будуть додатково завантажуватися.

Для забезпечення такої структури сайту нам як раз і потрібні файли routes, про які розказувалося раніше. На рисунку 2.8 зображено файл routes для користувачів з роллю Адміністратора.

```
const adminRoutes = [  
  {  
    path: "/admin-panel",  
    element: <AdminPanelPage />  
  },  
  {  
    path: "/admin-panel/new-user",  
    element: <UserRegisterPage />  
  }  
];
```

Рис. 2.8. Лістинг коду

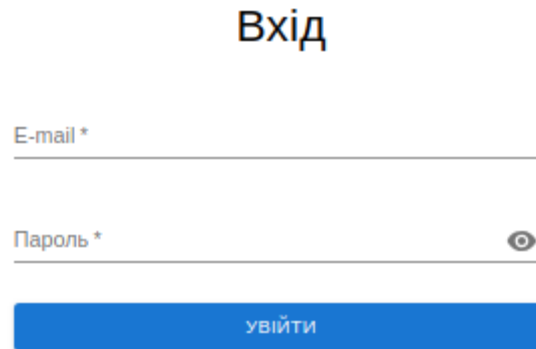
Далі на рисунку 2.9 наведено лістинг коду для загальних користувачів.

```
const routes = [  
  {  
    path: "/",  
    element: <HomePage />  
  },  
  {  
    path: "/login",  
    element: <LoginPage />  
  },  
  {  
    path: "/register",  
    element: <RegisterPage />  
  },  
  {  
    path: "/admin-panel",  
    element: <AdminPanelPage />  
  },  
  {  
    path: "/user-cabinet",  
    element: <UserCabinetPage />  
  }  
];
```

Рис. 2.9. Лістинг коду

Тепер, коли проєкт має свої напрямки розробки, давайте подивимось, як буде виглядати головна сторінка. Так як цей проєкт не є справжнім

заказом, і не має містити справжніх даних існуючої компанії, головна сторінка має досить пустий вигляд (Додаток А). У верхній частині екрану можна побачити header – елемент, який буде знаходитися на кожній сторінці сайту. У ньому є кнопка Увійти. При натисканні на цю кнопку, з’являється форма для авторизації на сайті (рис.2.10) [10].



The image shows a login form titled "Вхід" (Login) in blue text. Below the title are two input fields: "E-mail *" and "Пароль *" (Password *). The password field has a small eye icon to its right. Below these fields is a blue button with the text "увійти" (login) in white.

Рис. 2.10. Авторизація

Звичайно, усі форми на сайті мають перевірку інформації яку в них ввели, та вивід помилок, рисунок 2.11, якщо при авторизації були введені некоректні дані.

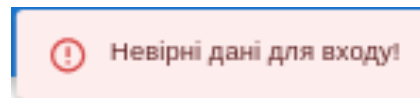


Рис. 2.11. Помилка при авторизації

Після успішної авторизації, кожен користувач отримує свій токен, який буду зберігатися з ним під час усього візиту сайту. У цьому токені зберігається інформація про id користувача, що в свою чергу дає доступ до ролі користувача. Таким чином сайт зможе відрізнити Робітника від Адміністратора, та завдяки цьому при натисканні однакових кнопок, в залежності від ролі, користувач сайту буде потрапляти на відповідні сторінки. Навіть якщо користувач захоче напряду перейти на сторінку, до якої в нього немає доступу, він отримає повідомлення «Сторінку не знайдено». Для цього у розробці використовувався JWT токен.

JWT - це стандарт відкритого обміну токенів, який дозволяє безпечно представляти інформацію між двома сторонами. JWT використовується для аутентифікації та передачі інформації про користувача між клієнтом та

сервером у вигляді JSON об'єкта. Із використанням JWT у вебдодатку отримуємо зручний та безпечний механізм обміну даними між клієнтом та сервером, спрощуючи аутентифікацію та авторизацію.

Для забезпечення більшої безпеки даних проектів компанії, було вирішено, що реєстрацію робітників буде виконувати адміністратор, тому після успішної аутентифікації під роллю адміністратора, потрапляємо в особистий кабінет.

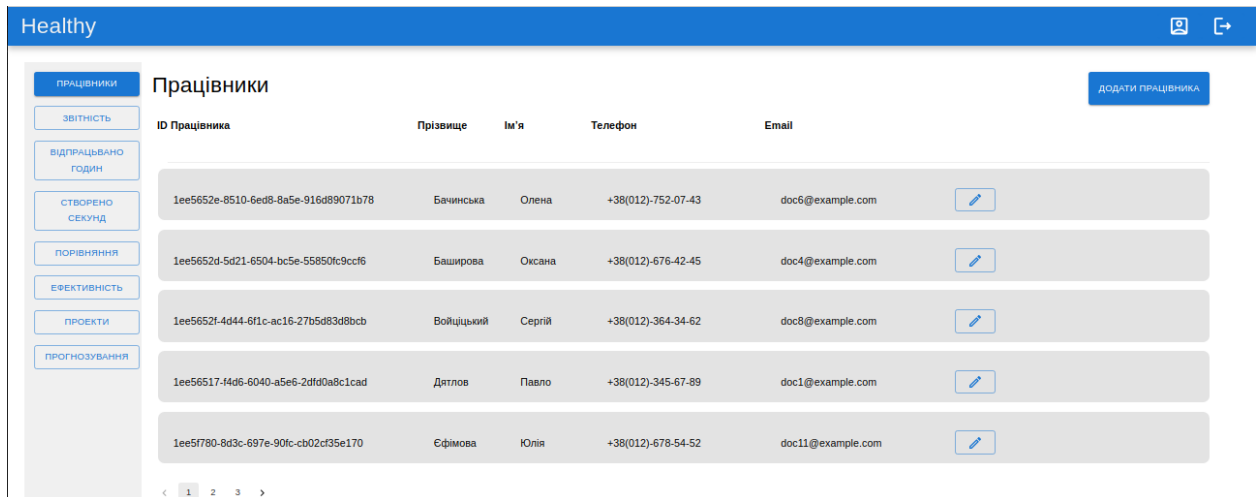


Рис. 2.12. Особистий кабінет адміністратора

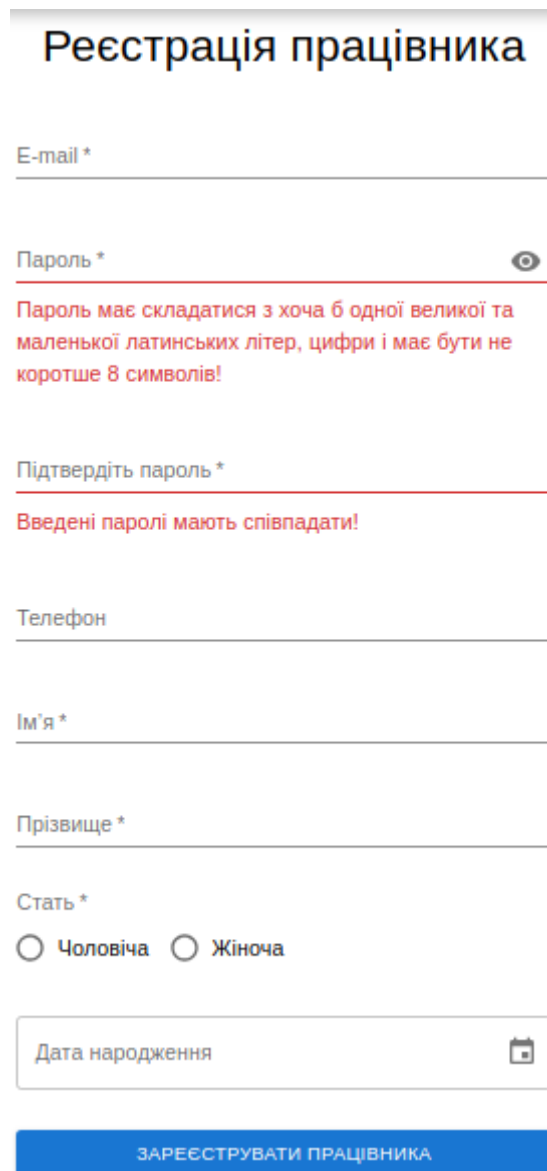
На рисунку 2.12 бачимо ліву панель, на якій розташовані кнопки, при натисканні на кожну з них буде змінюватися лише основна частина сторінки, не зачіпаючи при цьому ліву панель, чи верхню частину сайту та надаючи змогу користуватися іншими функціями адміністраційної панелі. Також на цьому зображенні у правій частині видно кнопку, при натисканні на яку відбудеться перехід на сторінку реєстрації робітників.

На (рис.2.13) зображено форму реєстрації працівника. Для успішної реєстрації потрібно коректно вказати такі данні як:

- e-mail;
- пароль;
- підтвердити пароль;
- телефон;
- ім'я;
- прізвище;

- стать;
- дату народження.

Для кожного з цих полів є свої критерії запису та при написанні хоча б одного поля не правильно, буде виведено повідомлення про помилку (рис.2.13).



The image shows a web form titled "Реєстрація працівника" (Employee Registration). It contains several input fields and validation messages:

- E-mail ***: A text input field.
- Пароль ***: A password input field with a visibility toggle icon. Below it, a red error message reads: "Пароль має складатися з хоча б одної великої та маленької латинських літер, цифри і має бути не коротше 8 символів!" (Password must consist of at least one uppercase and one lowercase Latin letter, a digit, and be at least 8 characters long!).
- Підтвердіть пароль ***: A confirmation password input field. Below it, a red error message reads: "Введені паролі мають співпадати!" (Entered passwords must match!).
- Телефон**: A text input field.
- Ім'я ***: A text input field.
- Прізвище ***: A text input field.
- Стать ***: Radio buttons for "Чоловіча" (Male) and "Жіноча" (Female).
- Дата народження**: A date picker field.
- Зареєструвати працівника**: A blue button at the bottom.

Рис. 2.13. Реєстрація працівника

Після успішної реєстрації працівника фірми, він з'явиться на першій сторінці адмін панелі, як це було показано на рисунку 2.12. Так як зараз розглядаємо роль Адміністратора, було б недоречно не додати йому можливість міняти абсолютно любі дані. Наприклад при натисканні кнопки справа від імені робочого, з'являється форма для редагування даних

(рис.2.14). При цьому блокується кнопка для редагування іншого робітника, це зроблено задля запобігання помилок при надсиланні запитів на сервер.

ID Працівника	Прізвище	Ім'я	Телефон	Email			
1ee5652e-8510-6ed8-8a5e-916d89071b78	Бачинська	Олена	+38(012)-752-07-43	doc6@example.com	✓	✕	🗑
1ee5652d-5d21-6504-bc5e-55850fc9ccf6	Баширова	Оксана	+38(012)-676-42-45	doc4@example.com	✎		

Рис. 2.14. Редагування даних робітника

Наступним кроком після реєстрації робітника, слід авторизуватися під роллю робітника, та спробувати створили новий запис. Для цього, виходимо з облікового запису адміністратору, натискаємо кнопку авторизації як у минулому випадку, але тепер вводимо ті дані, що вказали під час реєстрації робітника. Після успішної авторизації, потрапляємо вже в зовсім інше місце – особистий кабінет користувача (рис.2.15).

Healthy

Звітність

Звітність

Назва проекту *
Кількість годин *
Кількість секунд *
Додати запис

Прізвище	Ім'я	Назва проекту	Години роботи	Секунди	Дата запису
Швець	Юлія	Project C	3.00	10	2023-12-13 23:03:19
Єфімова	Юлія	Project D	4.50	19	2023-12-13 21:38:00
Левицький	Роман	Project A	3.00	15	2023-12-13 18:54:45
Швець	Юлія	Project C	1.50	8	2023-12-13 11:10:15
Войцицький	Сергій	Project B	3.00	11	2023-12-13 10:20:18
Мітлошук	Аркадій	Project C	4.00	8	2023-12-13 09:51:07
Войцицький	Сергій	Project B	2.00	25	2023-12-13 09:20:12

< 1 2 3 4 5 ... 15 >

Сергій Кулик
Email: doc12@example.com
Телефон: +38(012)-686-45-45
Дата народження: 06/08/1988
Стать: чоловік

Рис. 2.15. Особистий кабінет робітника

Цей кабінет має схожу структуру з кабінетом адміністратора, але в лівій частині є лише одна кнопка, яка відповідає за головну сторінку. В майбутньому у цей кабінет можна буде додати ще безліч функціоналу, тому така структуризація кабінету допоможе економити час, який могли б витратити на налаштування кабінету.

З правого боку вказана коротка інформація про користувача.

Основну частину екрану займає інформація про проекти, до якої входять:

- прізвище людини що створила запис;
- ім'я людини що створила запис;
- назва проекту, над яким вона працювала;
- час, який вона витратила;
- секунди, які вона анімувала за цей період;
- дата запису.

Таким чином, людина, котра тільки що створила новий запис, зможе подивитися на нього, і в випадку коли вона в чомусь помилилася, вона зможе звернутися до адміністратора, який в свою чергу має можливість редагувати усе, та виправити помилку.

Далі на рисунку 2.16 буде наведено частину коду, яка допомагає створювати такі сторінки, користуючись компонентами, що були раніше показані в архітектурі проекту.

```

<Grid container direction="column" spacing={2}>
  <Grid item>
    <Button
      variant={currentPage === "info" ? "contained" : "outlined"}
      onClick={() => changePage("info")}
      fullWidth
    >
      Звітність
    </Button>
  </Grid>
</Grid>
</div>

<div style={{ padding: "16px" }}>
  {currentPage === "info" && <UserInfoContainer setNotification={setNotification} />}
</div>

```

Рис. 2.16. Лістинг коду

Також у верхній частині екрану можна побачити форму для створення нового запису. Потрібно ввести дані про проект, які було вказано раніше, при цьому дані про користувача автоматично внесуться, тому що завдяки JWT токenu, сайт знає, хто саме зараз користується додатком. Усі поля цієї форми

звісно перевіряються на коректність вводу даних та у випадку введення хибних даних, видають помилку, а деякі поля, такі як поля годин та секунд, зовсім забороняють писати туди щось, крім цифр. Після відправки даних, без оновлення сторінки, таблиця з даними автоматично оновиться і бачимо актуальні дані (рис.2.17).

Звітність

Назва проекту *		Кількість годин *		Кількість секунд *		ДОДАТИ ЗАПИС
Прізвище	Ім'я	Назва проекту	Години роботи	Секунди	Дата запису	
Кулик	Сергій	Project D	4.00	10	2023-12-14 15:20:37	

Рис. 2.17. Форма створення запису та новий запис

Підводячи підсумки, під час створення Frontend частини проекту, було розроблено вебдодаток, що використовує компоненти для відображення інформації, JWT токен для аутентифікації користувача та частини HTML, CSS коду, для більш приязного відображення інформації на екрані.

2.3 Розробка API частини додатку

Перш за все, слід подивитися на структуру Backend частини проєкту.

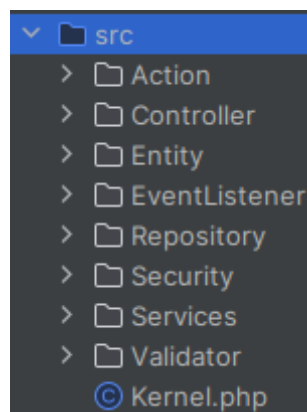


Рис. 2.18. Структура проєкту

Так як було обрано Symfony для розробки Backend частини проекту, слід розповісти про такі речі як:

1) Ентиті (Entities).

У Symfony ентиті використовуються для моделювання об'єктів даних, які відображають структуру бази даних [11]. Це дозволяє легко взаємодіяти з базою даних за допомогою об'єктно-орієнтованого підходу. Приклад частини коду ентиті можна побачити на рисунку 2.19. Весь код можна побачити у додатку В.

```
class Work implements JsonSerializable
{
    /**
     * @var Uuid|null
     */
    #[ORM\Id]
    #[ORM\Column(type: "uuid", unique: true)]
    #[ORM\GeneratedValue(strategy: "CUSTOM")]
    #[ORM\CustomIdGenerator(class: 'doctrine.uuid_generator')]
    #[Groups([
        "get:collection:work",
        "get:item:work"
    ])]
    private ?Uuid $id = null;
```

Рис. 2.19. Лістинг коду

У Symfony існує компонент ApiResource [13] та анотація Groups, які допомагають оптимізувати та керувати серіалізацією та десеріалізацією об'єктів при роботі з API. Давайте розглянемо ці аспекти більш детально:

ApiResource:

Компонент ApiResource дозволяє легко визначити API-ресурси та їхню конфігурацію. Використовуючи цей компонент, можна швидко створювати контролери та дії для взаємодії з ресурсами, а також налаштовувати формат виводу та вхідних даних.

Приклад використання анотації ApiResource можна побачити на рисунку 2.20.

```
#[ApiResource(
    collectionOperations: [
        "get" => [
            "method" => "GET",
            "normalization_context" => ['groups' => ["get:collection:work"]]
        ],
        "post" => [
            "method" => "POST",
            "denormalization_context" => ['groups' => ["post:collection:work"]],
            "normalization_context" => ['groups' => ["get:item:work"]],
            "security" => "is_granted('" . User::ROLE_ADMIN . "')"
        ]
    ]
),
```

Рис. 2.20. Лістинг коду

У цьому прикладі Post оголошено як API-ресурс, який підтримує операції отримання (get), створення (post), оновлення (put) та видалення (delete). Можна налаштувати це відповідно до API.

Groups.

Анотація Groups використовується для визначення, які групи властивостей об'єкта повинні бути включені чи виключені при серіалізації та десеріалізації. Це корисно для управління обсягом даних, які хочете повертати через API.

Приклад використання анотації Groups можна побачити на рисунку 2.21.

```
/**
 * @var User|null
 */
#[ORM\ManyToOne(targetEntity: User::class, inversedBy: 'doctorWork')]
#[Groups([
    "post:collection:work",
    "get:item:work",
    "get:collection:work"
])]
private ?User $user = null;
```

Рис. 2.21. Лістинг коду

У цьому прикладі поле «user» включено в групи post та get, що дозволяє використовувати його і при читанні, і при записі.

Також на рисунку 2.21 можна побачити використання зв'язку ManyToOne. В Symfony зв'язки ManyToOne використовуються для встановлення багатозначних відносин між сутностями бази даних, коли багато записів одного типу (many) пов'язані з одним записом іншого типу (one). Це використовується для моделювання взаємозв'язків, де багато записів одного типу пов'язані з одним записом іншого типу.

Давайте розглянемо приклад, де маємо дві сутності: User та Work. Кожен користувач може мати багато записів, але кожен запис може мати тільки одного автора. В цьому випадку використовується зв'язок ManyToOne.

Гетери і сетери (getter та setter) - це методи, які використовуються для отримання (getter) і встановлення (setter) значень властивостей об'єктів в програмі. Вони дозволяють контролювати доступ до властивостей класу і надають інтерфейс для звертання до даних об'єкта.

Гетер - це метод, який призначений для отримання значення конкретної властивості об'єкта. Зазвичай назва гетера визначається за зразком "getPropertyName", де "PropertyName" - назва властивості. Наприклад рисунок 2.22.

У прикладі getSeconds - це гетер для властивості seconds.

Сеттер - це метод, який використовується для встановлення значення конкретної властивості об'єкта. Зазвичай назва сеттера також визначається за зразком "setPropertyname", де "PropertyName" - назва властивості. Наприклад (рис.2.22):

У прикладі setSeconds - це сеттер для властивості seconds.

2) Репозиторії (Repositories).

Репозиторії в Symfony використовуються для взаємодії з базою даних та виконання запитів до ентиті. Репозиторій надає методи для отримання, оновлення, видалення та пошуку записів у базі даних. Приклад репозиторію для ентиті Work (рис.2.23):

```

/**
 * @return int|null
 */
public function getSeconds(): ?int
{
    return $this->seconds;
}

/**
 * @param int|null $seconds
 * @return $this
 */
public function setSeconds(?int $seconds): self
{
    $this->seconds = $seconds;

    return $this;
}

```

Рис. 2.22. Лістинг коду

```

<?php

namespace App\Repository;

use ...

/**
 * @extends ServiceEntityRepository<Work>
 *
 * @method Work|null find($id, $lockMode = null, $lockVersion = null)
 * @method Work|null findOneBy(array $criteria, array $orderBy = null)
 * @method Work[] findAll()
 * @method Work[] findBy(array $criteria, array $orderBy = null, $limit = null, $offset = null)
 */
class WorkRepository extends ServiceEntityRepository
{
    /**
     * @param ManagerRegistry $registry
     */
    public function __construct(ManagerRegistry $registry)
    {
        parent::__construct($registry, Work::class);
    }
}

```

Рис. 2.23. Лістинг коду

3) Слухачі (Listeners).

JWTCreatedListener - це слухач подій (event listener) в Symfony, який використовується для обробки події створення JWT токена під час аутентифікації. Цей слухач дозволяє вам втручатися в процес створення JWT токена та додавати додаткові дані до його підпису [14].

Використання JWTCreatedListener дозволяє динамічно розширювати JWT токен і додавати в нього додаткову інформацію, яка може бути корисною на клієнтському боці. Наприклад, можна додати користувацькі дані або права доступу, які будуть включені в токен.

Основним методом для реалізації JWTCreatedListener є метод onJWTCreated. Давайте розглянемо приклад використання JWTCreatedListener (рис.2.24).

```
1  <?php
2
3  namespace App\EventListener;
4
5  > use ...
6
7
8  class JWTCreatedListener
9  {
10
11     /**
12      * @param JWTCreatedEvent $event
13      *
14      * @return void
15      */
16     public function onJWTCreated(JWTCreatedEvent $event): void
17     {
18         /** @var User $user */
19         $user = $event->getUser();
20         $payload = $event->getData();
21         $payload['id'] = $user->getId()->toRfc4122();
22
23         $event->setData($payload);
24     }
25
26 }
27
```

Рис. 2.24. Лістинг коду

У цьому прикладі JWTCreatedListener слухає подію створення JWT токена. Далі до токена додаються додаткові дані. У цьому випадку, id додається до пейлоаду токена.

4) Валідатори (Validators).

ConstraintValidator - це клас в Symfony, який використовується для визначення правил валідації для полів вашого об'єкта. Використовуючи ConstraintValidator, можна створювати власні правила валідації, які будуть автоматично виконуватися при спробі виконати операції валідації в Symfony, наприклад, при обробці форм чи при збереженні об'єкта в базі даних.

Основними елементами при роботі з ConstraintValidator [15] є сам клас валідатора, а також асоційований з ним Constraint (Обмеження), яке визначає саме правило валідації. Ось приклад валідатора, для дня народжування користувача (рис.2.25):

```
class UserBirthdayConstraintValidator extends ConstraintValidator
{
    /**
     * @param $value
     * @param Constraint $constraint
     * @return void
     */
    public function validate($value, Constraint $constraint): void
    {
        if (!$constraint instanceof UserBirthdayConstraint) {
            throw new UnexpectedTypeException($constraint, expectedType: UserBirthdayConstraint::class);
        }

        if (!$value instanceof User) {
            throw new UnexpectedTypeException($value, expectedType: User::class);
        }

        if ($value->getBirthday() >= time()) {
            $this->context->addViolation( message: "Date of birth can't be equal to or greater than the current date");
        }
    }
}
```

Рис. 2.25. Лістинг коду

5) ApiPlatform [12].

ApiPlatform - це фреймворк для розробки вебсервісів та API на основі Symfony. Він надає можливості для швидкої розробки API, забезпечуючи автоматизований CRUD (Create, Read, Update, Delete) [17] та інші операції з даними. ApiPlatform використовує концепції REST та GraphQL для

спрощення взаємодії з вашими ресурсами. ApiPlatform базується на анотаціях, які вже бачили на рисунку 2.20, та використовує специфікації OpenAPI та JSON-LD для опису API. Ресурси описуються як класи з анотаціями, і ApiPlatform автоматично створює CRUD операції для цих ресурсів.

GET - використовується для отримання ресурсів з сервера. Наприклад, отримання списку всіх книг або конкретної книги.

POST - використовується для створення нових ресурсів на сервері. Наприклад, створення нової книги.

PUT - використовується для оновлення існуючого ресурсу на сервері. При використанні PUT, потрібно надсилати всі дані для оновлення ресурсу.

PATCH - використовується для часткового оновлення існуючого ресурсу на сервері. Можна надсилати тільки ті дані, які хочете змінити, без необхідності надсилати всю інформацію.

DELETE - використовується для видалення існуючого ресурсу на сервері.

На рисунку 2.26 наведено приклад того, як ApiPlatform виглядає у цьому вебдодатку, для класу Work.

Work			^
GET	/api/works	Retrieves the collection of Work resources.	▼ 🔒
POST	/api/works	Creates a Work resource.	▼ 🔒
GET	/api/works/{id}	Retrieves a Work resource.	▼ 🔒
PUT	/api/works/{id}	Replaces the Work resource.	▼ 🔒
DELETE	/api/works/{id}	Removes the Work resource.	▼ 🔒
PATCH	/api/works/{id}	Updates the Work resource.	▼ 🔒

Рис. 2.26. API частина додатку, клас Work

Тепер, якщо нам потрібно отримати усі дані про записи, що зберігаються у базі даних, нам лише потрібно надіслати запит на сервер, в якому передамо GET запит за адресою /api/works, і в відповідь отримаємо декілька записів, що матимуть наступну структуру (рис.2.27).

```

{
  "id": "12cc38be-96a3-11ee-8e5c-0242ac164b08",
  "title": "Project A",
  "hours": "5.50",
  "seconds": 27,
  "created": "2023-12-08 12:41:26",
  "user": {
    "id": "1ee564f5-33d5-68b8-ba9d-670bb1ff19b5",
    "firstName": "Павло",
    "lastName": "Дятлов"
  }
},

```

Рис. 2.27. Результат GET запиту

Для отримання даних, треба спочатку внести їх до бази даних. У вебдодатку це робиться завдяки інтерфейсу, а насправді це все робиться за допомогою POST запису на сервер. Для створення нового запису, потрібно зробити POST запит за адресою `/api/works`, дані потрібно заповнити за такою структурою (рис.2.28):

Example Value | Schema

```

{
  "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "title": "string",
  "hours": "string",
  "seconds": 0,
  "created": "string",
  "user": {
    "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
    "firstName": "string",
    "lastName": "string"
  }
}

```

Рис. 2.28. Структура POST запиту

Після відправки такого запиту, відповідно створиться запис у базі даних, який зможемо подивитися за допомогою GET запиту. Також зможемо змінити дані запиту, для цього, можемо використати PUT чи PATCH запит, відповідно обравши id запису, який хочемо відредагувати. Далі потрібно заповнити нові дані та надіслати запит. І нарешті, можемо видалити запис, для цього потрібно обрати id запиту, який вже не потрібен, і просто надсилаємо DELETE запит на сервер.

Підсумовуючи, можна сказати, що завдяки використанню усіх цих компонентів і працює серверна частина вебдодатку.

Висновок до розділу 2

Другий розділ роботи фокусується на практичній реалізації вебдодатку, розглядаючи аналіз вхідних даних, проектування та розробку додатку, а також розробку API частини системи. Аналіз вхідних даних дозволив визначити ключові вимоги до функціоналу додатку, що забезпечує ефективне вирішення завдань, поставлених у розділі 1.

Проектування та розробка вебдодатку представляють собою етап, на якому теоретичні концепції трансформуються в реальний продукт. Було розглянуто процес проектування, надано відомості про використані технології та методології розробки.

Розробка API частини додатку є ключовою для забезпечення його взаємодії з іншими системами та сервісами. Розглянули процес розробки API, визначили його основні функції та можливості.

В цілому, розділ 2 відображає практичний аспект дослідження, представляючи реалізацію теоретичних концепцій, визначених у попередньому розділі. Описана практична реалізація вебдодатку вносить вагомий внесок у розвиток обраної теми та демонструє можливості використання отриманих знань у практичних задачах.

РОЗДІЛ 3. ОЦІНКА ПРАЦЕЗДАТНОСТІ ВЕБДОДАТКУ

3.1 Опис та обґрунтування вибору засобів розробки

Після проведеного аналізу функціональних вимог та вимог до продуктивності, вибір засобів для реалізації вебсервісу моніторингу продуктивності працівників є критично важливим кроком у розробці. У цьому розділі будуть розглянуті вибрані технології та інструменти, які використовуватимуться для створення системи.

PHP [27].

- Висока продуктивність.

PHP 8.0 приніс значне покращення продуктивності порівняно зі своїми попередніми версіями. Це важливо для вебсервісу, який очікує багато запитів і операцій.

- Широка популярність.

PHP є однією з найпопулярніших мов програмування для веброзробки, що означає наявність великої спільноти розробників, різноманітних бібліотек та розширень.

- Спрощена розробка.

PHP має зручний синтаксис та велику кількість готових рішень для веброзробки. Це сприяє швидкому розгортанню та розвитку проекту [16].

- Підтримка об'єктно-орієнтованого програмування.

PHP підтримує об'єктно-орієнтований підхід, що полегшує структурування коду та розвиток великих проектів.

Застосування PHP 8.0 в проекті:

PHP 8.0 буде використовуватися для реалізації серверної логіки вебсервісу. Це включає:

- 1) Авторизація та аутентифікація.

PHP використовуватиметься для реалізації системи авторизації та аутентифікації користувачів, забезпечуючи безпеку доступу до особистих кабінетів.

2) Взаємодія з базою даних.

PHP буде використовуватися для обробки запитів до бази даних, включаючи зберігання та витягування інформації про користувачів та їх продуктивність.

3) Обробка та зберігання даних користувачів.

Всі дані, які користувачі вводять у систему, такі як робочий час та інформація про продуктивність, будуть оброблятися PHP та зберігатися відповідно до вимог безпеки.

4) Створення та перегляд статистики.

PHP допоможе у створенні статистичних звітів та графіків, що відображають продуктивність користувачів.

Підсумовуючи, PHP 8.0 обрана як мова програмування для розробки серверної сторони проекту через свою продуктивність, популярність, зручний синтаксис та готові рішення для веброботи. Використовуючи PHP, забезпечимо швидкий та ефективний вебсервіс для моніторингу продуктивності працівників.

Symfony [28].

Symfony 5.4 - це один з фреймворків для розробки вебдодатків на мові PHP. Його вибір для реалізації проекту вебсервісу моніторингу продуктивності працівників є обґрунтованим з ряду причин:

- Модульність та розширюваність.

Symfony побудований на принципах модульності та компонентної архітектури, що дозволяє обрати лише ті компоненти, які потрібні для конкретного проекту. Це дозволяє побудувати ефективну систему моніторингу.

- Зручний синтаксис.

Фреймворк має зручний синтаксис та готові рішення для багатьох завдань, що полегшує розробку та прискорює процес.

- Безпека.

Symfony має вбудовані інструменти для забезпечення безпеки додатків, включаючи захист від атак та можливість легко впроваджувати аутентифікацію та авторизацію користувачів.

- Запити та відповіді.

Symfony забезпечує зручну роботу з HTTP-запитами та HTTP-відповідями, що дозволяє легко створювати API для взаємодії з фронтом.

Symfony 5.4 буде використовуватися для побудови вебсервісу та реалізації різних частин проекту:

- 1) Маршрутизація.

Symfony дозволить визначити маршрути та контролери для обробки HTTP-запитів.

- 2) Модульність.

Кожен компонент Symfony можна використовувати окремо, що полегшує розширення функціоналу в майбутньому.

- 3) Аутентифікація та авторизація.

Symfony надає готові засоби для реалізації системи авторизації та аутентифікації користувачів.

- 4) Інтерфейс для фронту.

Symfony може обробляти запити та надсилати дані на фронтенд, включаючи графіки та статистику.

- 5) База даних.

Symfony легко інтегрується з базами даних, що дозволить нам зберігати та обробляти дані про працівників та їх продуктивність.

Вибір Symfony 5.4 допоможе створити стабільний, ефективний та безпечний вебсервіс для моніторингу продуктивності працівників.

React є однією з найпопулярніших бібліотек для розробки користувацьких інтерфейсів у сучасних вебдодатках [30]. Вибір React для використання в проєкті вебсервісу моніторингу продуктивності працівників обґрунтований з ряду факторів:

Ефективність: React пропонує відмінну продуктивність завдяки використанню віртуального DOM (Document Object Model). Він оновлює лише ті частини сторінки, які фактично змінилися, що призводить до оптимізації завантаження сторінки.

Спрощений розробка: React пропонує чітку систему компонентів, які розділяються на невеликі і перевикористовувані частини. Це полегшує підтримку коду та його подальший розвиток.

Велика спільнота: React має велику та активну спільноту розробників, яка підтримує багато сторонніх бібліотек та розширень, що полегшує інтеграцію та розвиток проєкта.

Реактивний підхід: React реагує на зміни стану додатка та автоматично оновлює відображення, що допомагає побудувати інтерактивні та динамічні користувацькі інтерфейси.

Застосування React в проєкті:

React використовуватиметься для створення користувацького інтерфейсу вебсервісу моніторингу продуктивності працівників. Це означає, що всі вебсторінки, на яких користувачі взаємодіють з системою, будуть побудовані з використанням React.

Деякі з функціональностей React в проєкті включають:

- Компоненти інтерфейсу: Створення компонентів для різних частин системи, таких як особистий кабінет працівника, панель адміністратора та інші.
- Взаємодія з бекендом: Використання React для взаємодії з бекендом, отримання та відправки даних про продуктивність працівників.
- Інтерактивність: Додавання можливостей для користувачів фіксувати робочий час та вводити дані про продуктивність.
- Візуалізація даних: Відображення даних у вигляді графіків, діаграм та таблиць для зручного аналізу даних адміністраторами.

Маючи на увазі всі ці переваги, вибір React допоможе створити сучасний та ефективний користувацький інтерфейс вебсервісу моніторингу

продуктивності працівників, який буде відповідати сучасним стандартам та вимогам користувачів.

WebStorm.

WebStorm - це інтегроване середовище розробки (IDE), розроблене компанією JetBrains, спеціалізоване на розробці вебдодатків, включаючи клієнтську та серверну частини. Ця IDE є важливим інструментом для розробників, які працюють з вебтехнологіями, оскільки вона надає широкий функціонал та інструменти для комфортної та продуктивної розробки [31]. Ось деякі ключові характеристики та можливості WebStorm:

Підтримка JavaScript, HTML та CSS: WebStorm надає розширений редактор для мов програмування та технологій, таких як JavaScript, HTML та CSS. Редактор має функцію автодоповнення, що полегшує написання коду та зменшує кількість помилок.

Підтримка React та інших фреймворків: Для розробників, які працюють з React або іншими JavaScript-фреймворками, WebStorm надає підтримку для цих фреймворків, включаючи автодоповнення та інші корисні функції.

Відлагодження: WebStorm має вбудовану підтримку для відлагодження JavaScript, що дозволяє розробникам виявляти та виправляти помилки в коді.

Підтримка систем контролю версій: WebStorm легко інтегрується з системами контролю версій, такими як Git, що полегшує відстеження та управління змінами в коді.

Підтримка плагінів: WebStorm підтримує плагіни, які дозволяють розширити функціонал IDE залежно від потреб розробника.

Візуальне оформлення коду: WebStorm має можливості для оформлення та форматування коду за стандартами розробки, що робить код більш зрозумілим.

WebStorm є популярним вибором серед веброзробників завдяки своєму багатофункціональному підходу та інструментам, які спрощують розробку вебдодатків, включаючи ті, які стосуються проекту моніторингу продуктивності працівників.

GitHub.

GitHub - це платформа для спільної роботи над програмним забезпеченням з використанням системи контролю версій Git [32]. У проєкті GitHub використовуватиметься для зберігання, відстеження та керування кодом додатку. Важливі аспекти використання GitHub включають:

Версії коду: Git дозволяє відстежувати зміни в коді, зберігаючи копії кожного коміту. Це дозволяє відновити попередні версії коду у випадку помилок або несправності.

Спільна робота: GitHub надає можливість багаторазового розроблення та спільної роботи над кодом. Кожен учасник проєкту може створювати гілки (branches), вносити зміни та об'єднувати їх (merge) у головну гілку.

Зберігання коду та резервне копіювання: GitHub надає централізоване зберігання коду, що забезпечує його безпеку та доступність навіть у разі втрати даних на локальних комп'ютерах розробників.

Відстеження проблем: GitHub дозволяє створювати та відстежувати завдання (issues), а також вести дискусії про різні аспекти розробки.

Інтеграція WebStorm та GitHub сприяє ефективній та організованій розробці вебсервісу моніторингу продуктивності працівників. Розробники можуть легко взаємодіяти, відстежувати та вносити зміни в код проєкту, забезпечуючи надійність та стабільність додатку.

Підсумовуючи все вище сказане, для відповідності вимогам та завданням проєкту мій вибір технологій буде наступним:

- Php 8.0;
- Symfony 5.4;
- MySQL;
- React;
- WebStorm;
- Github.

Обрані технології та інструменти відповідають потребам проєкту і дозволять мені реалізувати вебсервіс моніторингу продуктивності

працівників з необхідною функціональністю та продуктивністю. Такий вибір забезпечить успішну розробку та впровадження проекту.

3.2 Опис інтерфейсу системи

Для більшого розуміння зібраної інформації, краще всього вивести її у вигляді графіку, який зображено на рисунку 3.1.

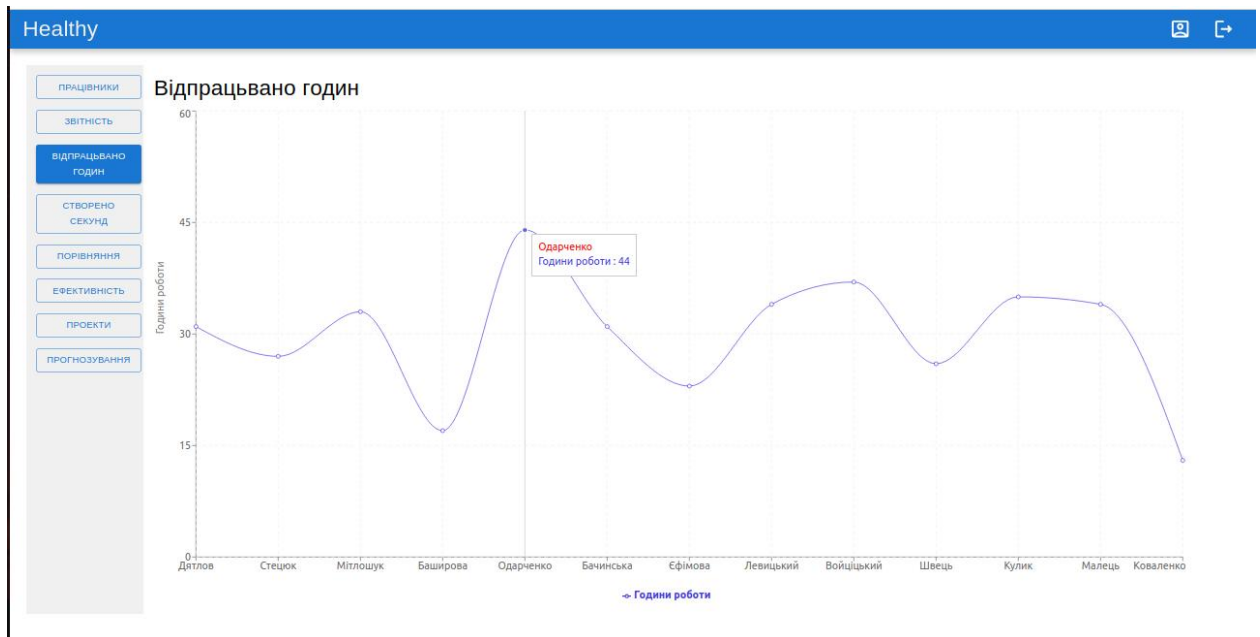


Рис. 3.1 Графік відпрацьованих годин

На цьому графіку на шкалі X зображено Прізвища робочих і відповідно на шкалі Y зображено кількість годин, що вони відпрацювали за певний період часу. Для прикладу, візьмемо людину, з фамілією Одарченко. Бачимо що за весь час вона відпрацювала 44 години, і порівнюючи з іншими, це майже найбільший результат. Але що він дає? Як зрозуміти, чи ефективно людина витратила свій час? Для цього переходимо до наступного пункту в лівому меню вебдодатку. Наводячись на місце, де має бути дані працівника Одарченко (рис.3.2), тепер бачимо кількість створених секунд анімації. 180 секунди відео, це рівно 3 хвилини відео. Це дасть змогу зрозуміти ефективність цього працівника. Дізнаючись скільки секунд анімації працівник робить за годину часу, дізнаємося його ефективність, але з чим це порівнювати? Правильно, з ефективністю інших працівників (рис.3.3).

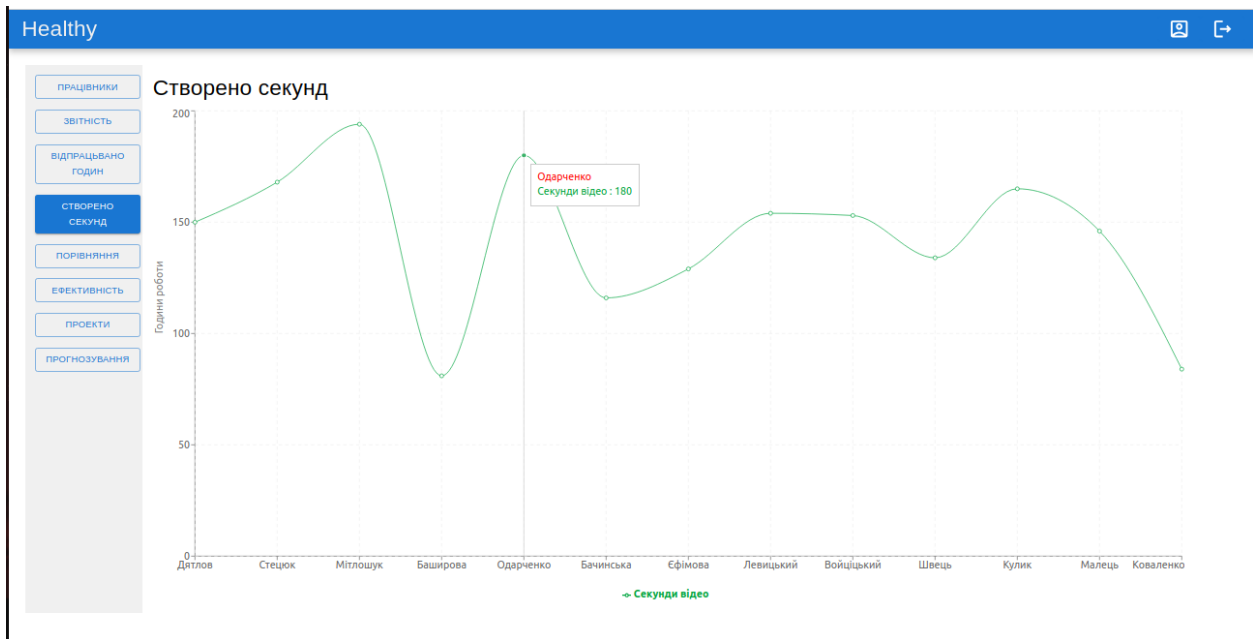


Рис. 3.2. Графік створених секунд анімації

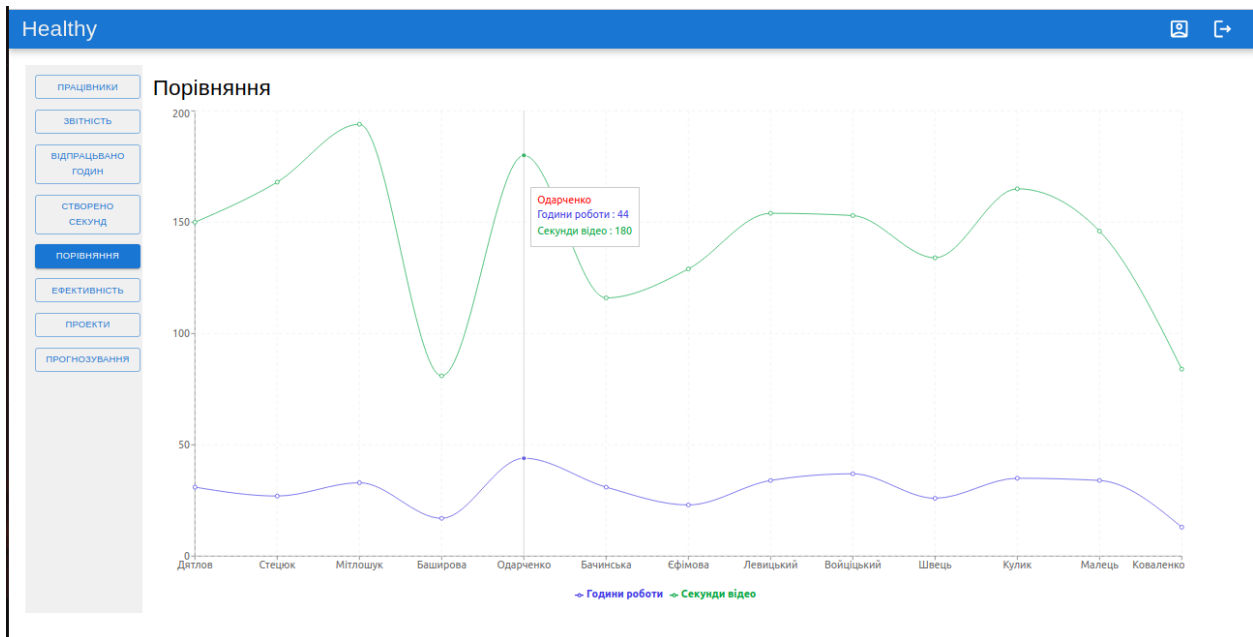


Рис. 3.3. Порівняння годин та секунд працівників

Але таке зображення даних все ще не досить інформативне, тому було прийнято рішення, вивести окремо графік, на якому буде відображатися ефективність кожного працівника. Вона рахувалася за формулою: кількість створених секунд анімації поділити на кількість відпрацьованих годин. Таким чином дізнаємося, скільки секунд анімації людина робить за одну годину часу, це в подальшому буде називатися ефективністю.

Після того, як дізналися ефективність кожної людини, можемо порівняти їх між собою, і відразу побачити людину з найнижчою ефективністю – Бачинська (рис.3.4). Такі дані допоможуть керівництву компанії зрозуміти, хто досить добре справляється зі своєю роботою, а хто не дуже. Можливо потрібно змінити команду у цієї людини, чи можливо навіть звільнити. А людині яка набагато краще справляється з поставленою задачею, можливо навіть зроблять премію. Такі статистичні дані допоможуть при такому виборі.

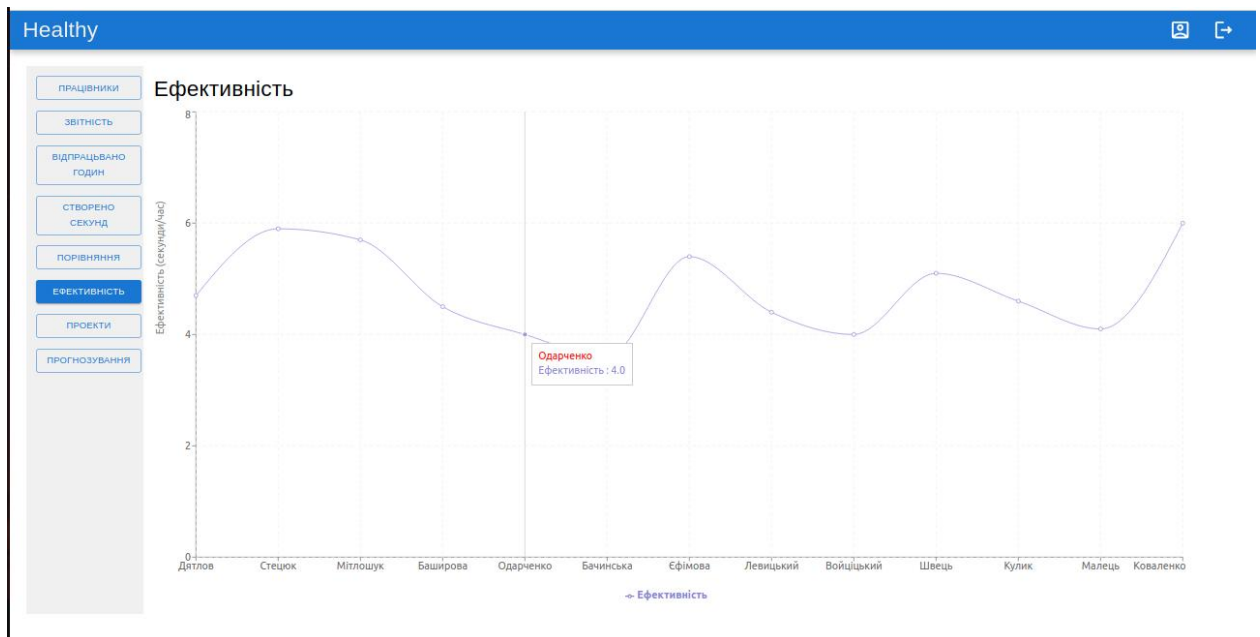


Рис. 3.4. Ефективність

Також схоже зображення інформації можна зробити на прикладі проектів, над якими завжди працює певна кількість людей, майже завжди це одна команда. На рисунку 3.5 можемо побачити 4 проекти, над якими працюють 4 команди, 2 з яких мають 3 працівника і 2 команди має 4 працівника. Для кожного проекту відображається кількість секунд, що людина вже зробила, це відображає відсоток вкладу людини у цей проект. І з відси витікає цікава деталь, справ в тому, що компанія яка надає роботу цим людям може мати декілька варіантів оплати, таких як: фіксована щомісячна оплата, оплата по годині праці та оплата за кількість створених секунд. У деяких випадках компанії не важливо, скільки людина витратила на свою

роботу часу, головне щоб робота була виконана, і таке зображення даних допоможе розібратися з цим.

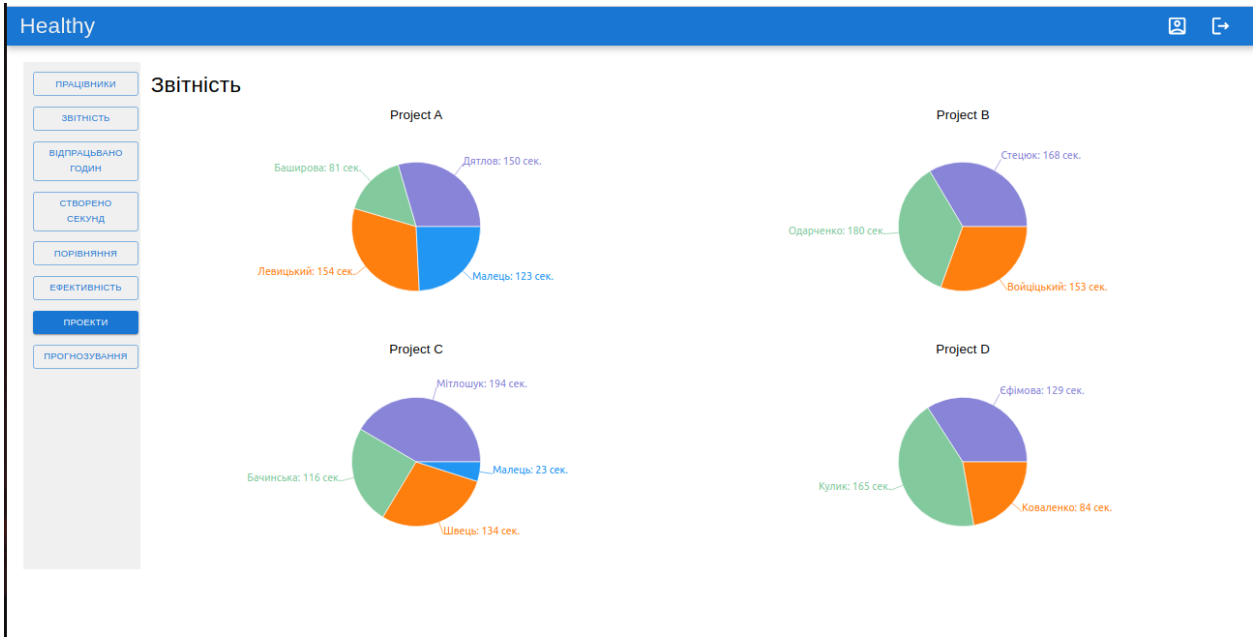


Рис. 3.5. Статистика проектів

Також детально можна побачити особистий вклад кожного у проект, і на основі цього, можна спробувати замінити дуже ефективних людей з однієї команди на менш ефективних людей з іншої команди, для створення балансу між ними (рис.3.6).

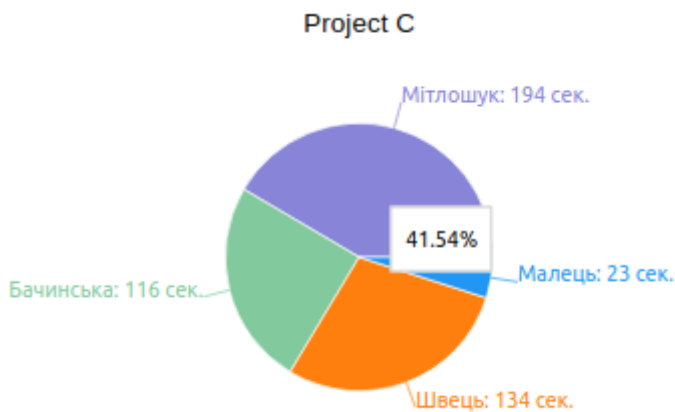


Рис. 3.6 Вклад у команду

І останнім пунктом в лівому меню буде пункт під назвою «Прогнозування» (рис.3.7). У кожного проекту, який замовляють у компанії, є свої певні рамки, а саме, довжина відео чи анімації. Також важливою частиною кожного проекту є дата здачі або як кажучи дедлайн. І якщо дедлай

у проекту закінчується через 2 неділі, а у проекту зроблено лише 52,5%, як у випадку з проектом D, то скоріш за все потрібно поквипитися, тому як за 2 минулі неділі, за які збиралися ці дані, команда зробила лише половину. Керівництво компанії повинно повідомити працівників, щоб вони більше працювали над цим проектом, чи у крайньому випадку, залучити когось з іншого проекту, в кого справи не такі кепські.

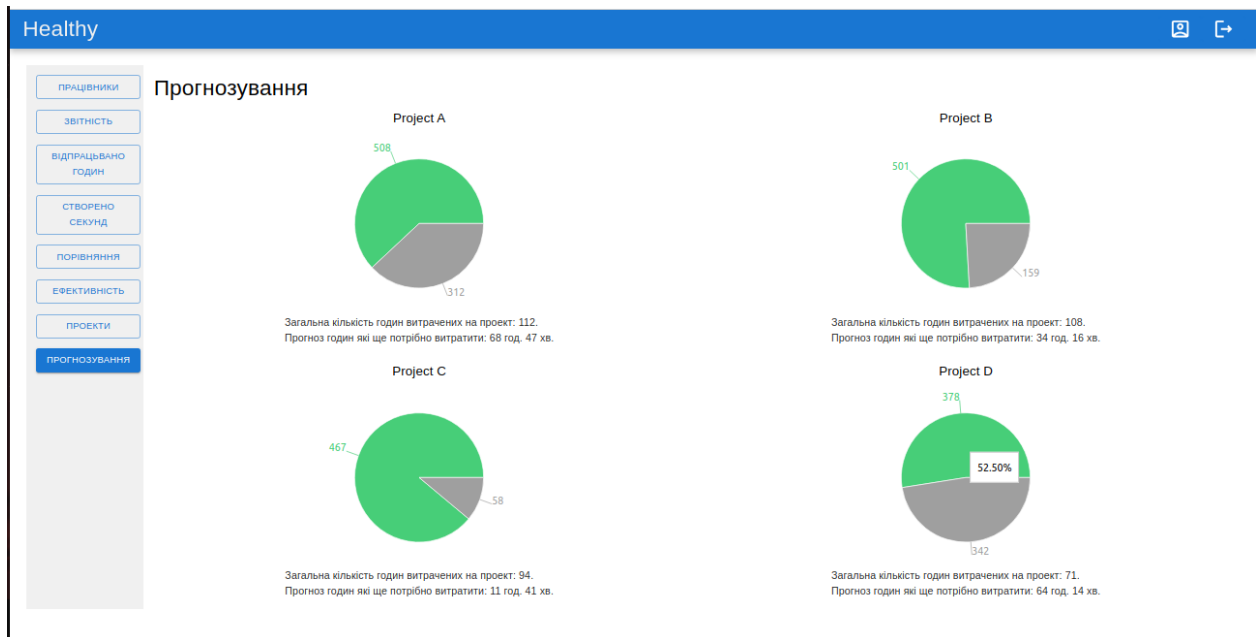


Рис. 3.7. Прогнозування

Підводячи підсумки, можна сказати що таке графічне зображення інформації допомагає більш детально зрозуміти усі аспекти компанії, її команд та працівників. І на основі цих даних, робити кроки у сторону покращення роботи компанії.

3.3 Тестування вебдодатку

Одним з найголовніших етапів розробки програмного продукту є його тестування, з метою виявлення неочікуваних недоліків чи багів. Та просто з ціллю переконатися у справності усіх аспектів вебдодатку.

Для цього оберемо профіль працівника Одарченко, якого розглядали у минулому пункті.

Перш за все, авторизація. Вводимо відповідні дані та натискаємо кнопку Увійти (рис.3.8).

Вхід

E-mail *

doc5@example.com

Пароль *

.....

УВІЙТИ

Рис. 3.8 Авторизація

Далі потрапляємо у особистий кабінет, де бачимо форму для створення нового запису (рис.3.9).

Healthy

Звітність

Назва проекту *

Кількість годин *

Кількість секунд *

ДОДАТИ ЗАПИС

Прізвище	Ім'я	Назва проекту	Години роботи	Секунди	Дата запису	
Кулик	Сергій	Project D	4.00	10	2023-12-14 15:20:37	
Швець	Юлія	Project C	3.00	10	2023-12-13 23:03:19	
Єфімова	Юлія	Project D	4.50	19	2023-12-13 21:38:00	
Левицький	Роман	Project A	3.00	15	2023-12-13 18:54:45	
Швець	Юлія	Project C	1.50	8	2023-12-13 11:10:15	
Войціцький	Сергій	Project B	3.00	11	2023-12-13 10:20:18	
Мітлошук	Аркадій	Project C	4.00	8	2023-12-13 09:51:07	

<

1

2

3

4

5

...

15

>

Сергій Одарченко

Email: doc5@example.com

Телефон: +38(012)-782-75-14

Дата народження: 11/30/1973

Стать: чоловік

Рис. 3.9. Особистий кабінет працівника

Далі заповнюємо форму для створення нового запису, вказуючи при цьому довільні дані, але обираючи проект В як показано на рисунку 3.10.

Звітність

Назва проекту *

Project B

Кількість годин *

8

Кількість секунд *

24

ДОДАТИ ЗАПИС

Рис. 3.10. Форма створення нового запису

Натискаючи кнопку «Додати запис», відправляємо запит на сервер, який в свою чергу створить новий запис у базі даних та після цього автоматично оновить дані на сайті, без необхідності оновлювати сторінку, як це довелося б робити, використовуючи сайт без використання React.

На рисунку 3.11 бачимо що новий запис з’явився в таблиці, з тими даними що було вказано раніше, та з даними, які взялись автоматично, а саме : Прізвище працівника, Ім’я працівника та дата створення запису.

Звітність

Назва проекту *

Кількість годин *

Кількість секунд *

ДОДАТИ ЗАПИС

Прізвище	Ім'я	Назва проекту	Години роботи	Секунди	Дата запису
Одарченко	Сергій	Project B	8.00	24	2023-12-15 14:36:14

Рис. 3.11. Оновлені дані

Після того як змінили дані, давайте зайдемо у вебдодаток під профілем адміністратора, щоб переглянути статистику. Для цього натискаємо кнопку «Вихід» що знаходиться справа з верху (рис.3.12).



Рис. 3.12 Кнопка «Вихід»

Далі авторизуємося як це було показано раніше на рисунку 3.8, але використовуємо дані адміністратора. Після натискання кнопки «Вхід», потрапляємо до профілю адміну. Тепер переглянемо дані, котрі щойно ввели, натискаємо на кнопку «Відпрацьовано годин» і після наведення на працівника, бачимо що кількість годин змінилася рівно на стільки, скільки було вказано на рисунку 3.13. Аналогічно, натискаючи кнопку «Створено секунд», бачимо що кількість секунд анімації також змінилися на стільки, скільки вказали раніше рисунку 3.14.

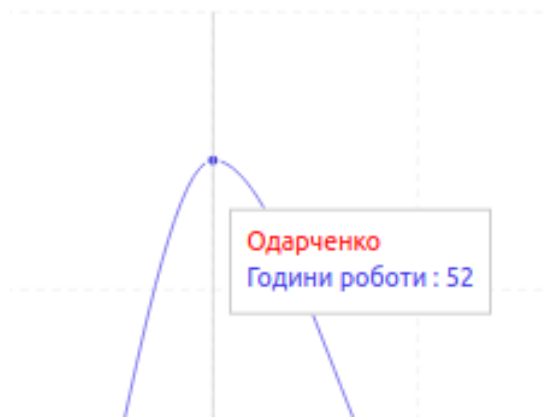


Рис. 3.13. Зміна годин роботи

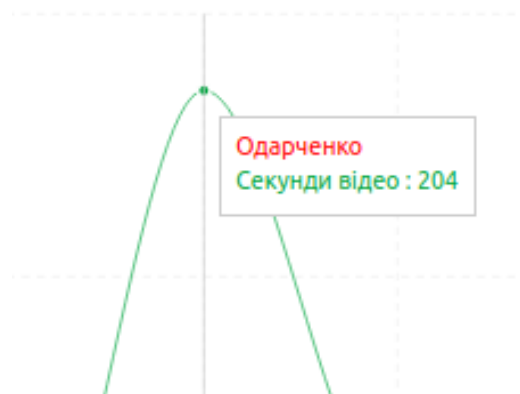


Рис. 3.14. Зміна секунд анімації

А на останнє, можемо подивитися як змінився вклад працівника у проект (рис.3.15).

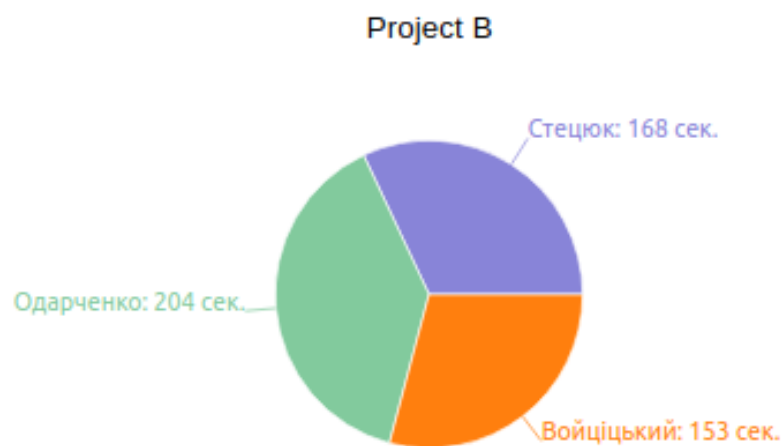


Рис. 3.15. Зміна вкладу у проект

Таким чином, протестувавши вебдодаток, переконалися що він працює справно. І головне, він ідеально виконує свою роботу, а саме відображення зібраних даних працівників.

Висновок до розділу 3

Розділ 3 присвячений оцінці працездатності вебдодатку, що є важливим етапом у процесі його впровадження та експлуатації. Опис та обґрунтування вибору засобів розробки висвітлює стратегічні рішення, які вплинули на архітектуру та функціональність додатку.

Опис інтерфейсу системи дозволяє зрозуміти, як користувачі будуть взаємодіяти з вебдодатком. Було детально розглянуто дизайн інтерфейсу, його зручність та ефективність в контексті вирішення завдань, які стоять перед системою.

Тестування вебдодатку в розділі 3 визначається як критичний етап, спрямований на перевірку його функціональності, безпеки та ефективності. Представили результати тестування, що дає можливість оцінити відповідність розробленого додатку поставленим вимогам та виявити можливі несправності.

Завершуючи розділ 3, можна визначити, що вивчені аспекти оцінки працездатності вебдодатку грають важливу роль у його успішному впровадженні та подальшому функціонуванні. Такий підхід дозволяє забезпечити надійність та ефективність системи, що є ключовими факторами у забезпеченні задоволення користувачів та досягненні поставлених цілей.

ВИСНОВКИ

У магістерській кваліфікаційній роботі було вирішено актуальне завдання по розробці вебдодатку моніторингу продуктивності працівників.

Основний результат цієї дослідницької роботи полягав в створенні вебсервісу для моніторингу продуктивності працівників. Було визначено ключовий функціонал, який включає в себе можливість реєстрації та авторизації користувачів, ведення обліку робочого часу та кількості виконаної роботи, фіксацію продуктивності, а також інструменти для адміністратора, які дозволять переглядати та аналізувати дані працівників, зокрема фільтрування, сортування та виведення графіків на основі цих даних.

Для цього було обрано фреймворк Symfony за його гнучкість та легкість. Для взаємодії з динамічними даними було обрано React, що дає змогу легко редагувати та оновлювати потрібні записи або додавати нові.

Практична цінність.

1) Ефективне управління ресурсами. Вебсервіс дозволить організаціям ефективно управляти робочим часом та ресурсами, виявляти слабкі місця у виробничому процесі та швидко реагувати на них.

2) Підвищення продуктивності. Через систематичний моніторинг та аналіз даних, працівники матимуть можливість вдосконалювати свою продуктивність, виправляти проблемні аспекти та збільшувати ефективність своєї роботи.

3) Покращення комунікації. Адміністраторські інструменти дозволять керівникам ефективно спілкуватися зі своєю командою, створюючи більш прозоре та сприятливе середовище для обміну інформацією та вирішення завдань.

4) Стимулювання працівників. Спостереження за власною продуктивністю може слугувати джерелом мотивації для працівників, що в свою чергу позитивно впливає на загальний дух колективу та підвищує рівень залученості.

Напрямки подальших досліджень.

Попри досягнені результати, дослідження в сфері моніторингу продуктивності працівників може бути подальше розвинуто та удосконалене. Ось кілька напрямків, які варто розглядати для подальших наукових досліджень:

1) Розширення функціоналу. Дослідження може бути розширене для включення нових функцій, таких як вивчення структури команди, або взаємодія з іншими системами управління бізнес-процесами.

2) Аналіз впливу на робочий клімат. Дослідження може охоплювати аналіз впливу вебсервісу на робочий клімат та задоволеність працівників, досліджуючи їхні відгуки та реакції на впровадження нової системи.

3) Мобільні додатки. Розробка мобільних додатків для вебсервісу може розширити його досяжність та зручність для працівників, особливо для тих, хто працює в дистанційному режимі.

Ці напрямки дозволяють поглибити розуміння впливу створеного вебсервісу на організаційний контекст та створити більш повне та адаптоване рішення для вирішення завдань у сфері моніторингу продуктивності працівників.

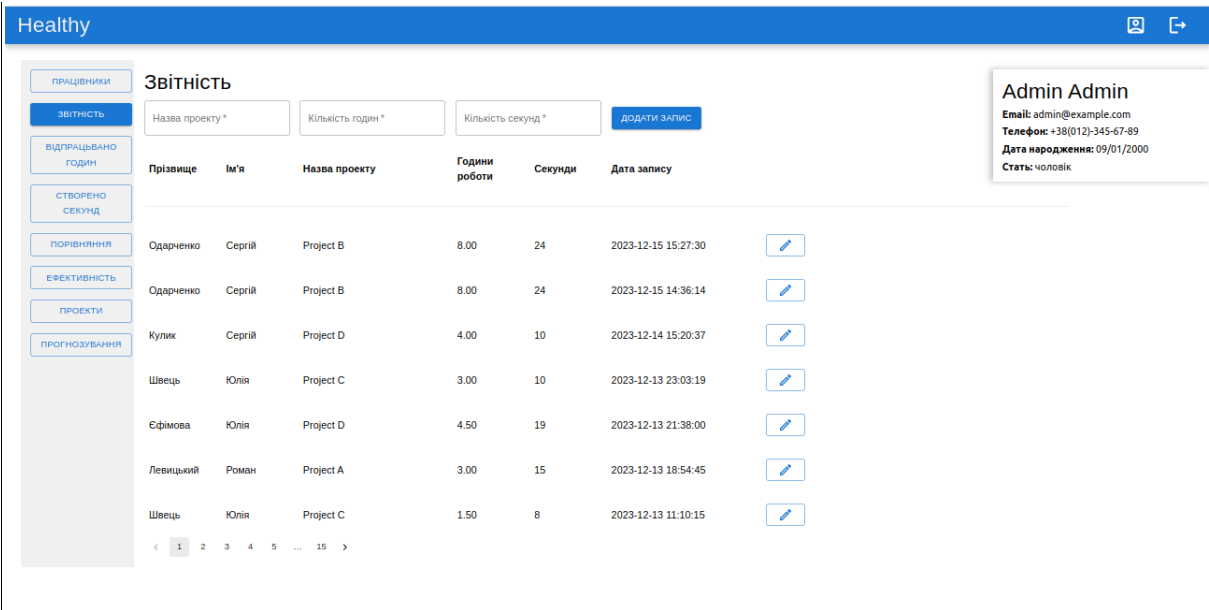
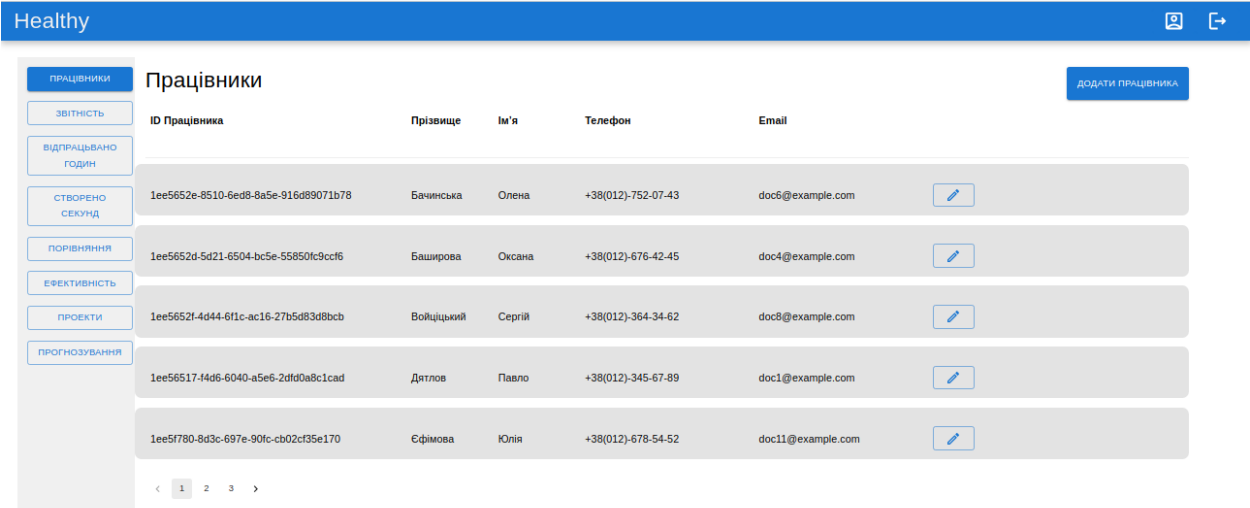
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Гринчуцький, В.І. Економіка підприємства: навчальний посібник [Текст] / В.І. Гринчуцький, Е.Т. Карапетян, Б.В. Погріщук. – К.: Центр учбової літератури, 2010. – 304 с.
2. Kanungo T. An Efficient k-means clustering algorithm: analysis and implementation / T. Kanungo, D. M Наукові праці ВНТУ, 2012, № 2
3. Олійник, С. В. і Бойко, Б. М. (2023) «Порівняння методів кластеризації для створення цільових груп клієнтів у наборі даних», Науковий вісник НЛТУ України, 33(5), с. 77-83.
4. Бідюк П. І., Романенко В. Д., Тимощук О. Л. Аналіз часових рядів: навч. посіб. ННК «Інститут прикладного системного аналізу» Національний технічний університет України «Київський політехнічний інститут». 2010. 317 с.
5. A. Nielsen, Practical Time Series Analysis: Prediction with Statistics and Machine Learning. O'Reilly Media, Inc., 2020, pp. 504.
6. Vyshnyakova, M., “How to implement the KRI system in the company”, Lyudi I liderstvo, Upravlenie personalom, vol. 2(74), 2012, p. 22-28.
7. Sotnik S., Manakov V., Lyashenko V. Overview: PHP and MySQL Features for Creating Modern Web Projects // International Journal of Academic Information Systems Research (IJAIRS). – 2023. – Vol. 7(1). – P. 11-17.
8. Sotnik S., Shakurova T., Lyashenko V. Development Features Web-Applications // International Journal of Academic and Applied Research (IJAAAR) . – 2023. – Vol. 7(1). – P. 79-85.
9. Rajabov, S. B. (2023). The role of backend and frontend information systems infrastructure. Science and Education, 4(3), 212–216.
10. Mukhtar B, A., Lumingkewas, C. S., & Rofi'i, A. (2023). The Implementation of User Centered Design Method in Developing UI/UX. Journal of Information System, Technology and Engineering, 1(2), 26–31.
11. Salehi S. Mastering Symfony / Sohail Salehi., 2016. – 290 с.

12. Rehben E. Sharing data through an API platform - API AGRO / E. Rehben, B. Balvay, T. P. Haezebrouck. // ICAR, Via Savoia 78, 00198 Rome, Italy. – 2018. – С. 122.
13. Serrano D, Stroulia E, Lau D, Ng T (2017) Linked REST APIs: a middleware for semantic REST API integration. IEEE Int Conf Web Serv 2017:138–145.
14. Laaziri M. A comparative study of laravel and symfony PHP frameworks / M. Laaziri, K. Benmoussa, S. Khouliji., 2019. – 704-712 с.
15. Späth P. Bean Validation / P. Späth. // Pro Jakarta EE. – 2023. – №10.
16. Sotnik S., Manakov V., Lyashenko V. Overview: PHP and MySQL Features for Creating Modern Web Projects // International Journal of Academic Information Systems Research (IJASIR). – 2023. – Vol. 7(1). – P. 11-17.
17. Ciprian-Octavian T. Performance Evaluation for CRUD Operations in Asynchronously Replicated Document Oriented Database / T. Ciprian-Octavian, R. Florin. // IEEE. – 2015. – С. 1318.
18. Bernard M. Key Performance Indicators: The 75 Measures Every Manager Needs To Know / Marr Bernard. – Great Britain: Financial Times Press, 2012. – 347 с.
19. FRIGO. The Balanced Scorecard / FRIGO, L. MARK, R. KRUMWIEDE. // Strategic Finance. – 2023. – №7. – С. 50.
20. Що таке КРІ. [Електронний ресурс] – 2023. Режим доступу до ресурсу <https://www.dsnews.ua/ukr/economics/strashnyy-son-menedzhera-chto-takoe-kpi-18062021-428687>
21. Збалансована система показників. [Електронний ресурс] – 2023. Режим доступу до ресурсу <https://www.maxzosim.com/balanced-scorecard-bsc/>
22. Методика "360 градусів". [Електронний ресурс] – 2023. Режим доступу до ресурсу <https://www.osoba.ua/services/staff-assessment/ocenka-360/>
23. Workpuls. [Електронний ресурс] – 2023. Режим доступу до ресурсу <https://www.insightful.io/>

24. Hubstaff. [Електронний ресурс] – 2023. Режим доступу до ресурсу <https://hubstaff.com/>
25. Teramind. [Електронний ресурс] – 2023. Режим доступу до ресурсу <https://www.teramind.co/>
26. Yaware. [Електронний ресурс] – 2023. Режим доступу до ресурсу <https://yaware.com.ua/uk/>
27. PHP. [Електронний ресурс] – 2023. Режим доступу до ресурсу <https://www.php.net/docs.php>
28. Symfony. [Електронний ресурс] – 2023. Режим доступу до ресурсу <https://symfony.com.ua/doc/current/index.html>
29. MySQL. [Електронний ресурс] – 2023. Режим доступу до ресурсу <https://dev.mysql.com/doc/>
30. React. [Електронний ресурс] – 2023. Режим доступу до ресурсу <https://uk.legacy.reactjs.org/docs/getting-started.html>
31. WebStorm. [Електронний ресурс] – 2023. Режим доступу до ресурсу <https://www.jetbrains.com/webstorm/>
32. GitHub. [Електронний ресурс] – 2023. Режим доступу до ресурсу <https://github.com/>
33. Вакалюк Т.А, Кузьмук В.О. ПЕРЕВАГИ ТА НЕДОЛІКИ ВИКОРИСТАННЯ МОБИ JAVASCRIPT. Сучасні комп'ютерні та інформаційні системи і технології: матеріали III Всеукраїнської наук.-практ. інтернет-конф. / Т.А. Вакалюк, В.О.Кузьмук // Запоріжжя: ТДАТУ. – 2022. – С. 331-332
34. Вакалюк Т.А, Кузьмук В.О. ВИКОРИСТАННЯ LAMP ДЛЯ РОЗРОБКИ ВЕБ-ЗАСТОСУНКІВ. Сучасні комп'ютерні та інформаційні системи і технології: матеріали III Всеукраїнської наук.-практ. інтернет-конф. / Т.А. Вакалюк, В.О.Кузьмук // Запоріжжя: ТДАТУ. – 2022. – С. 326-327

ДОДАТКИ



Додаток Б

id (DC2Type:uuid)	title	seconds	user_id (DC2Type:uuid)	hours	created
0x12cc38be96a311ee8e5c0242ac164b08	Project A	27	0x1ee564f533d568b8ba9d670bb1ff19b5	5.50	2023-12-08 12:41:26
0x12cc3f0096a311ee8e5c0242ac164b08	Project B	24	0x1ee564f768c06b48939f377d2f205189	2.00	2023-12-10 18:17:02
0x12cc3fea96a311ee8e5c0242ac164b08	Project C	39	0x1ee56507b58f6bf8be34e1d42bcd65f	3.00	2023-12-07 05:20:52
0x12d046dd96a311ee8e5c0242ac164b08	Project A	20	0x1ee565081bcb67828265f3e6045f16e9	2.00	2023-12-10 19:53:17
0x12d04a4e96a311ee8e5c0242ac164b08	Project B	30	0x1ee5650848fe642a9a782f031f27eb16	6.00	2023-12-11 11:23:47
0x12d04ae696a311ee8e5c0242ac164b08	Project C	11	0x1ee5650886c46fd4a79a9585812ef5d4	5.50	2023-12-10 21:19:04
0x12d04b5996a311ee8e5c0242ac164b08	Project D	14	0x1ee5f7808d296c5ca54bcb02cf35e170	6.00	2023-12-13 00:24:01
0x12d4e7a896a311ee8e5c0242ac164b08	Project A	14	0x1ee56508ea066b74aa347d4680b7876d	5.00	2023-12-11 10:02:53
0x12d4ec4996a311ee8e5c0242ac164b08	Project B	7	0x1ee565091ea0679abdead53a21dda506	4.50	2023-12-11 01:02:24
0x12d4eda496a311ee8e5c0242ac164b08	Project C	10	0x1ee5650966c360cc993661cc32f3adec	3.00	2023-12-13 23:03:19
0x12d4ee1c96a311ee8e5c0242ac164b08	Project D	11	0x1ee5f79b7ef269848a5e4bf8d0869cd1	3.00	2023-12-08 16:09:23
0x12db43cf96a311ee8e5c0242ac164b08	Project A	11	0x1ee5f76d6814606a9a12f76f0a9b1d9f	3.50	2023-12-08 11:37:01
0x12db46f596a311ee8e5c0242ac164b08	Project B	11	0x1ee564f768c06b48939f377d2f205189	4.00	2023-12-09 09:36:56
0x12db479796a311ee8e5c0242ac164b08	Project C	13	0x1ee56507b58f6bf8be34e1d42bcd65f	2.50	2023-12-07 13:13:37
0x1498647096a311ee8e5c0242ac164b08	Project A	23	0x1ee564f533d568b8ba9d670bb1ff19b5	3.00	2023-12-09 13:17:19
0x1498726a96a311ee8e5c0242ac164b08	Project B	23	0x1ee5650848fe642a9a782f031f27eb16	4.00	2023-12-11 02:45:42
0x149873d196a311ee8e5c0242ac164b08	Project C	6	0x1ee5650886c46fd4a79a9585812ef5d4	5.00	2023-12-12 21:56:33
0x149ced4996a311ee8e5c0242ac164b08	Project A	27	0x1ee565081bcb67828265f3e6045f16e9	4.50	2023-12-10 05:25:39
0x149cf34896a311ee8e5c0242ac164b08	Project B	17	0x1ee565091ea0679abdead53a21dda506	6.00	2023-12-12 09:18:40
0x149cf4a596a311ee8e5c0242ac164b08	Project C	30	0x1ee5650966c360cc993661cc32f3adec	2.00	2023-12-10 06:16:21
0x149cf55796a311ee8e5c0242ac164b08	Project D	21	0x1ee5f7a73d6a647e9eae71d3f128f71	4.00	2023-12-07 03:25:45
0x14a088de96a311ee8e5c0242ac164b08	Project A	8	0x1ee56508ea066b74aa347d4680b7876d	1.50	2023-12-11 22:19:43
0x14a08c1096a311ee8e5c0242ac164b08	Project B	12	0x1ee564f768c06b48939f377d2f205189	2.00	2023-12-10 05:21:19
0x14a08cba96a311ee8e5c0242ac164b08	Project C	7	0x1ee56507b58f6bf8be34e1d42bcd65f	4.00	2023-12-08 07:47:30
0x14a08d2796a311ee8e5c0242ac164b08	Project D	18	0x1ee5f7808d296c5ca54bcb02cf35e170	2.00	2023-12-10 22:53:30
0x14a3e98596a311ee8e5c0242ac164b08	Project A	15	0x1ee5f76d6814606a9a12f76f0a9b1d9f	2.50	2023-12-08 19:05:03
0x14a3eca096a311ee8e5c0242ac164b08	Project B	9	0x1ee5650848fe642a9a782f031f27eb16	6.00	2023-12-11 02:44:47
0x14a3ed4796a311ee8e5c0242ac164b08	Project C	14	0x1ee5650886c46fd4a79a9585812ef5d4	8.00	2023-12-08 04:28:43
0x161663d996a311ee8e5c0242ac164b08	Project A	12	0x1ee564f533d568b8ba9d670bb1ff19b5	2.50	2023-12-07 14:09:16
0x1616696b96a311ee8e5c0242ac164b08	Project B	25	0x1ee565091ea0679abdead53a21dda506	2.00	2023-12-13 09:20:12
0x16166a4696a311ee8e5c0242ac164b08	Project C	10	0x1ee5650966c360cc993661cc32f3adec	2.00	2023-12-09 04:13:12

id (DC2Type:uuid)	title	seconds	user_id (DC2Type:uuid)	hours	created
0x1a2345a396a311ee8e5c0242ac164b08	Project A	26	0x1ee56508ea066b74aa347d4680b7876d	5.00	2023-12-10 07:32:45
0x1a2345a696a311ee8e5c0242ac164b08	Project B	17	0x1ee564f768c06b48939f377d2f205189	3.00	2023-12-11 12:54:30
0x1a2345a996a311ee8e5c0242ac164b08	Project C	31	0x1ee56507b58f6bf8be34e1d42bcdd65f	4.00	2023-12-12 16:16:15
0x1a2345ac96a311ee8e5c0242ac164b08	Project D	19	0x1ee5f7808d296c5ca54bcb02cf35e170	4.50	2023-12-13 21:38:00
0x1a2345af96a311ee8e5c0242ac164b08	Project A	12	0x1ee5f76d6814606a9a12f76f0a9b1d9f	2.50	2023-12-07 02:59:45
0x1a2345b296a311ee8e5c0242ac164b08	Project B	25	0x1ee5650848fe642a9a782f031f27eb16	5.00	2023-12-08 08:21:30
0x1a2345b596a311ee8e5c0242ac164b08	Project C	14	0x1ee5650886c46fd4a79a9585812ef5d4	3.50	2023-12-09 13:43:15
0x1a2345b896a311ee8e5c0242ac164b08	Project D	11	0x1ee5f7a73d6a647e9eae71d3f128f71	2.50	2023-12-10 19:05:00
0x1a2345bb96a311ee8e5c0242ac164b08	Project A	18	0x1ee56508ea066b74aa347d4680b7876d	4.00	2023-12-11 00:26:45
0x1a2345be96a311ee8e5c0242ac164b08	Project B	29	0x1ee565091ea0679abdead53a21dda506	6.00	2023-12-12 05:48:30
0x1a2345c196a311ee8e5c0242ac164b08	Project C	8	0x1ee5650966c360cc993661cc32f3adec	1.50	2023-12-13 11:10:15
0x1a2345c496a311ee8e5c0242ac164b08	Project D	21	0x1ee5f79b7ef269848a5e4bf8d0869cd1	4.00	2023-12-07 22:32:00
0x1a2345c796a311ee8e5c0242ac164b08	Project A	13	0x1ee5f76d6814606a9a12f76f0a9b1d9f	3.00	2023-12-08 03:53:45
0x1a2345ca96a311ee8e5c0242ac164b08	Project B	22	0x1ee5650848fe642a9a782f031f27eb16	4.50	2023-12-09 09:15:30
0x1a2345cd96a311ee8e5c0242ac164b08	Project C	11	0x1ee5650886c46fd4a79a9585812ef5d4	2.50	2023-12-10 14:37:15
0x1a2345d096a311ee8e5c0242ac164b08	Project D	17	0x1ee5f7a73d6a647e9eae71d3f128f71	3.00	2023-12-11 19:59:00
0x1a2345d396a311ee8e5c0242ac164b08	Project A	24	0x1ee56508ea066b74aa347d4680b7876d	4.00	2023-12-12 01:20:45
0x1a2345d696a311ee8e5c0242ac164b08	Project B	15	0x1ee565091ea0679abdead53a21dda506	3.50	2023-12-13 06:42:30
0x1a2345d996a311ee8e5c0242ac164b08	Project C	30	0x1ee5650966c360cc993661cc32f3adec	6.00	2023-12-07 18:04:15
0x1a2345dc96a311ee8e5c0242ac164b08	Project D	28	0x1ee5f79b7ef269848a5e4bf8d0869cd1	5.50	2023-12-08 23:26:00
0x1a2345df96a311ee8e5c0242ac164b08	Project A	9	0x1ee5f76d6814606a9a12f76f0a9b1d9f	2.00	2023-12-10 04:47:45
0x1a2345e296a311ee8e5c0242ac164b08	Project B	14	0x1ee564f768c06b48939f377d2f205189	2.50	2023-12-11 10:09:30
0x1a2345f196a311ee8e5c0242ac164b08	Project C	23	0x1ee5f76d6814606a9a12f76f0a9b1d9f	4.00	2023-12-08 04:35:30
0x1ee96a8ac09f6be296cb992ab32822dc	Project A	45	0x1ee564f533d568b8ba9d670bb1ff19b5	10.00	2023-12-10 00:15:51
0x1ee98cb4a33a6826984cd5317d6d4d11	Project D	13	0x1ee5f79b7ef269848a5e4bf8d0869cd1	4.00	2023-12-12 08:48:57
0x1ee9a9455906691e99c8f1ae4c4ad3c9	Project D	10	0x1ee5f79b7ef269848a5e4bf8d0869cd1	4.00	2023-12-14 15:20:37
0x1ee9b574cbf263529d0f8d08a61fe952	Project B	24	0x1ee5650848fe642a9a782f031f27eb16	8.00	2023-12-15 14:36:14
0x1ee9b5e761736e36aeb97144d009d8af	Project B	24	0x1ee5650848fe642a9a782f031f27eb16	8.00	2023-12-15 15:27:30
0x978e82fd96a211ee8e5c0242ac164b08	Project A	10	0x1ee564f533d568b8ba9d670bb1ff19b5	4.00	2023-12-11 11:21:12
0x97acb01696a211ee8e5c0242ac164b08	Project B	18	0x1ee564f768c06b48939f377d2f205189	3.00	2023-12-13 07:58:27
0x97acb2d196a211ee8e5c0242ac164b08	Project C	36	0x1ee56507b58f6bf8be34e1d42bcdd65f	3.00	2023-12-11 05:48:42

id (DC2Type:uuid)	title	seconds	user_id (DC2Type:uuid)	hours	created
0x161e968c96a311ee8e5c0242ac164b08	Project A	18	0x1ee565081bcb67828265f3e6045f16e9	4.00	2023-12-12 17:05:25
0x161e9b5396a311ee8e5c0242ac164b08	Project B	28	0x1ee564f768c06b48939f377d2f205189	4.00	2023-12-08 00:47:31
0x161e9cf696a311ee8e5c0242ac164b08	Project C	13	0x1ee56507b58f6bf8be34e1d42bcdd65f	6.00	2023-12-09 00:41:19
0x161e9d6b96a311ee8e5c0242ac164b08	Project D	24	0x1ee5f79b7ef269848a5e4bf8d0869cd1	5.00	2023-12-07 01:04:04
0x162285b796a311ee8e5c0242ac164b08	Project A	12	0x1ee56508ea066b74aa347d4680b7876d	4.00	2023-12-08 03:16:23
0x162288e596a311ee8e5c0242ac164b08	Project B	11	0x1ee5650848fe642a9a782f031f27eb16	4.00	2023-12-12 13:09:42
0x1622898696a311ee8e5c0242ac164b08	Project C	16	0x1ee5650886c46fd4a79a9585812ef5d4	2.00	2023-12-10 07:59:22
0x162289fb96a311ee8e5c0242ac164b08	Project D	19	0x1ee5f7a73d6a647e9eaed71d3f128f71	2.00	2023-12-07 00:27:46
0x162689c296a311ee8e5c0242ac164b08	Project A	13	0x1ee5f76d6814606a9a12f76f0a9b1d9f	6.50	2023-12-11 02:20:43
0x16268e3e96a311ee8e5c0242ac164b08	Project B	11	0x1ee565091ea0679abdead53a21dda506	3.00	2023-12-13 10:20:18
0x16268f4196a311ee8e5c0242ac164b08	Project C	7	0x1ee5650966c360cc993661cc32f3adec	5.00	2023-12-12 20:39:21
0x1a23456796a311ee8e5c0242ac164b08	Project A	33	0x1ee564f533d568b8ba9d670bb1ff19b5	7.00	2023-12-09 15:23:45
0x1a23456a96a311ee8e5c0242ac164b08	Project B	18	0x1ee564f768c06b48939f377d2f205189	3.50	2023-12-10 20:45:30
0x1a23456d96a311ee8e5c0242ac164b08	Project C	28	0x1ee56507b58f6bf8be34e1d42bcdd65f	3.00	2023-12-11 10:12:15
0x1a23457096a311ee8e5c0242ac164b08	Project D	22	0x1ee5f7808d296c5ca54bcb02cf35e170	4.50	2023-12-12 14:33:00
0x1a23457396a311ee8e5c0242ac164b08	Project A	15	0x1ee56508ea066b74aa347d4680b7876d	3.00	2023-12-13 18:54:45
0x1a23457696a311ee8e5c0242ac164b08	Project B	20	0x1ee565091ea0679abdead53a21dda506	4.00	2023-12-08 05:08:30
0x1a23457996a311ee8e5c0242ac164b08	Project C	12	0x1ee5650966c360cc993661cc32f3adec	3.00	2023-12-07 17:29:15
0x1a23457c96a311ee8e5c0242ac164b08	Project D	33	0x1ee5f79b7ef269848a5e4bf8d0869cd1	5.50	2023-12-09 21:51:00
0x1a23457f96a311ee8e5c0242ac164b08	Project A	25	0x1ee5f76d6814606a9a12f76f0a9b1d9f	4.00	2023-12-12 01:12:45
0x1a23458296a311ee8e5c0242ac164b08	Project B	14	0x1ee564f768c06b48939f377d2f205189	2.50	2023-12-08 06:33:30
0x1a23458596a311ee8e5c0242ac164b08	Project C	19	0x1ee56507b58f6bf8be34e1d42bcdd65f	4.50	2023-12-10 11:55:15
0x1a23458896a311ee8e5c0242ac164b08	Project D	27	0x1ee5f7808d296c5ca54bcb02cf35e170	3.00	2023-12-11 03:17:00
0x1a23458b96a311ee8e5c0242ac164b08	Project A	30	0x1ee56508ea066b74aa347d4680b7876d	5.00	2023-12-09 08:38:45
0x1a23458e96a311ee8e5c0242ac164b08	Project B	11	0x1ee5650848fe642a9a782f031f27eb16	2.00	2023-12-10 13:00:30
0x1a23459196a311ee8e5c0242ac164b08	Project C	23	0x1ee5650886c46fd4a79a9585812ef5d4	4.00	2023-12-11 18:22:15
0x1a23459496a311ee8e5c0242ac164b08	Project D	16	0x1ee5f7a73d6a647e9eaed71d3f128f71	2.50	2023-12-12 22:44:00
0x1a23459796a311ee8e5c0242ac164b08	Project A	18	0x1ee5f76d6814606a9a12f76f0a9b1d9f	3.00	2023-12-13 04:05:45
0x1a23459a96a311ee8e5c0242ac164b08	Project B	22	0x1ee565091ea0679abdead53a21dda506	4.50	2023-12-08 09:27:30
0x1a23459d96a311ee8e5c0242ac164b08	Project C	13	0x1ee5650966c360cc993661cc32f3adec	3.00	2023-12-07 20:49:15
0x1a2345a096a311ee8e5c0242ac164b08	Project D	8	0x1ee5f79b7ef269848a5e4bf8d0869cd1	2.00	2023-12-09 02:11:00

```

<?php

namespace App\Entity;

use ApiPlatform\Core\Annotation\ApiResource;
use App\Repository\WorkRepository;
use ApiPlatform\Core\Annotation\ApiFilter;
use ApiPlatform\Core\Bridge\Doctrine\Orm\Filter\OrderFilter;
use DateTimeImmutable;
use Doctrine\DBAL\Types\Types;
use Doctrine\ORM\Mapping as ORM;
use JsonSerializable;
use Symfony\Component\Serializer\Annotation\Groups;
use Symfony\Component\Uid\Uuid;
use Symfony\Component\Validator\Constraints\NotBlank;
use Symfony\Component\Validator\Constraints\NotNull;
use Symfony\Component\Validator\Constraints\PositiveOrZero;

#[ORM\Entity(repositoryClass: WorkRepository::class)]
#[ApiResource(
    collectionOperations: [
        "get" => [
            "method"          => "GET",
            "normalization_context" => ['groups' => ["get:collection:work"]]
        ],
        "post" => [
            "method"          => "POST",
            "denormalization_context" => ['groups' => ["post:collection:work"]],
            "normalization_context"  => ['groups' => ["get:item:work"]],
            "security"              => "is_granted('" . User::ROLE_ADMIN . "') or is_granted('" . User::ROLE_DOCTOR .
        "')"
    ],
    itemOperations: [
        "get" => [
            "method"          => "GET",
            "normalization_context" => ['groups' => ["get:item:work"]]
        ],
        "put" => [
            "method"          => "PUT",
            "denormalization_context" => ['groups' => ["put:item:work"]],
            "security"          => "is_granted('" . User::ROLE_ADMIN . "') or is_granted('" . User::ROLE_DOCTOR .
        "')"
    ],
        "patch" => [
            "method"          => "PATCH",
            "denormalization_context" => ['groups' => ["patch:item:work"]],
            "security"          => "is_granted('" . User::ROLE_ADMIN . "') or is_granted('" . User::ROLE_DOCTOR .

```

```

    ""
    ],
    "delete" => [
        "method" => "DELETE",
        "security" => "is_granted('' . User::ROLE_ADMIN . '') or is_granted('' . User::ROLE_DOCTOR . '')"
    ]
]
)]
#[ApiFilter(OrderFilter::class, properties: ['created' => 'DESC'])]
class Work implements JsonSerializable
{
    /**
     * @var Uuid|null
     */
    #[ORM\Id]
    #[ORM\Column(type: "uuid", unique: true)]
    #[ORM\GeneratedValue(strategy: "CUSTOM")]
    #[ORM\CustomIdGenerator(class: 'doctrine.uuid_generator')]
    #[Groups([
        "get:collection:work",
        "get:item:work"
    ])]
    private ?Uuid $id = null;

    /**
     * @var string|null
     */
    #[NotNull]
    #[NotBlank]
    #[ORM\Column(length: 255)]
    #[Groups([
        "get:collection:work",
        "post:collection:work",
        "get:item:work",
        "put:item:work",
        "patch:item:work"
    ])]
    private ?string $title = null;

    /**
     * @var string|null
     */
    #[PositiveOrZero]
    #[NotNull]
    #[ORM\Column(type: Types::DECIMAL, precision: 7, scale: 2, nullable: true)]
    #[Groups([
        "get:collection:work",
        "post:collection:work",
        "get:item:work",
        "put:item:work",
        "patch:item:work"
    ])]

```

```

private ?string $hours = null;

/**
 * @var int|null
 */
#[NotNull]
#[ORM\Column(type: "integer")]
#[Groups([
    "get:collection:work",
    "post:collection:work",
    "get:item:work",
    "put:item:work",
    "patch:item:work"
])]
private ?int $seconds = null;

/**
 * @var string
 */
#[ORM\Column(type: "string", length: 255, nullable: false)]
#[Groups([
    "get:collection:work",
    "post:collection:work",
    "get:item:work",
    "put:item:work",
    "patch:item:work"
])]
private string $created;

/**
 * @var User|null
 */
#[ORM\ManyToOne(targetEntity: User::class, inversedBy: 'doctorWork')]
#[Groups([
    "post:collection:work",
    "get:item:work",
    "get:collection:work"
])]
private ?User $user = null;

public function __construct()
{
    $this->created = (new DateTimeImmutable())->format('Y-m-d H:i:s');
}

/**
 * @return Uuid|null
 */
public function getId(): ?Uuid
{
    return $this->id;
}

```

```
/**
 * @return string|null
 */
public function getTitle(): ?string
{
    return $this->title;
}

/**
 * @param string|null $title
 * @return $this
 */
public function setTitle(?string $title): self
{
    $this->title = $title;

    return $this;
}

public function getHours(): ?string
{
    return $this->hours;
}

public function setHours(?string $hours): self
{
    $this->hours = $hours;

    return $this;
}

/**
 * @return int|null
 */
public function getSeconds(): ?int
{
    return $this->seconds;
}

/**
 * @param int|null $seconds
 * @return $this
 */
public function setSeconds(?int $seconds): self
{
    $this->seconds = $seconds;

    return $this;
}

public function getCreated(): string
{

```

```

        return $this->created;
    }

    public function setCreated(?string $created): self
    {
        $this->created = $created;

        return $this;
    }

    /**
     * @ORM\PrePersist
     */
    public function setCreatedValue(): void
    {
        $this->created = (new DateTimeImmutable())->format('Y-m-d H:i:s');
    }

    public function getUser(): ?User
    {
        return $this->user;
    }

    public function setUser(?User $user): self
    {
        $this->user = $user;

        return $this;
    }

    /**
     * @return array
     */
    public function jsonSerialize(): array
    {
        return [
            'id' => $this->id,
            'title' => $this->title,
            'hours' => $this->hours,
            'seconds' => $this->seconds,
            'created' => $this->created,
            'user' => [
                'id' => $this->getUser()->getId()
            ],
        ];
    }
}

```

```

import React from "react";
import {
  Button,
  FormControl,
  IconButton,
  Input,
  InputAdornment,
  InputLabel,
  TextField,
  Typography
} from "@mui/material";
import { Visibility, VisibilityOff } from "@mui/icons-material";
import { NavLink } from "react-router-dom";

const LoginForm = ({ setAuthData, loading }) => {
  const handleSubmit = (event) => {
    event.preventDefault();

    setAuthData({
      username: event.target.username.value,
      password: event.target.password.value
    });
  };

  const [showPassword, setShowPassword] = React.useState(false);

  const handleClickShowPassword = () => setShowPassword((show) => !show);

  return (
    <form className="auth-form" onSubmit={handleSubmit}>
      <Typography
        align="center"
        variant="h4"
      >
        Вхід
      </Typography>
      <TextField
        variant="standard"
        id="username"
        type="email"
        label="E-mail"
        name="username"
        required
      />
      <FormControl variant="standard" required>
        <InputLabel htmlFor="standard-adornment-password">Пароль</InputLabel>
        <Input
          type={showPassword ? "text" : "password"}
          id="password"
          name="password"
          endAdornment={

```

```

        <InputAdornment position="end">
          <IconButton
            aria-label="toggle password visibility"
            onClick={handleClickShowPassword}
          >
            {showPassword ? <VisibilityOff /> : <Visibility />}
          </IconButton>
        </InputAdornment>
      }
    />
  </FormControl>
  <Button
    variant="contained"
    type="submit"
    disabled={loading}
  >
    Увійти
  </Button>
  <Typography
    align="center"
    variant="p"
    sx={{color: '#a0a0a0'}}
  >
    або
  </Typography>
  <Button
    variant="text"
    type="submit"
    to="/register"
    component={NavLink}
    disabled={loading}
  >
    Зареєструватися
  </Button>
</form>
);
};

export default LoginForm;

```