МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
ІНСТИТУТ ІНФОРМАТИКИ ТА РАДІОЕЛЕКТРОНІКИ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК І ТЕХНОЛОГІЙ
(повне найменування інституту, факультету)
КАФЕДРА СИСТЕМНОГО АНАЛІЗУ ТА ОБЧИСЛЮВАЛЬНОЇ МАТЕМАТИКИ
(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)
бакалавра
(ступінь вищої освіти)

на тему **Програмна система оптимізації керування персоналом приватної фірми**

Виконав: студент(ка) 4 курсу, групи КНТ-818сп

Спеціальності 124 – Системний аналіз
(код і найменування спеціальності)

Освітня програма (спеціалізація)
«Інтелектуальні технології та прийняття рішень в складних системах»

Шовкопляс В.А.
(прізвище та ініціали)

Керівник Рябенко А.Є.
(прізвище та ініціали)

Рецензент Пирожок А.В.
(прізвище та ініціали)

2021

Форма № 25

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

Інститут, факультет Інститут інформатики і радіоелектроніки, факульте комп'ютерних наук та технологій
Кафедра Системного аналізу та обчислювальної математики
Ступінь вищої освіти бакалавр
Спеціальність 124 - Системний аналіз
(код і найменування)
Освітня програма (спеціалізація) «Інтелектуальні технології та прийняття рішень в складних системах»
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ
Завідувач кафедри _____
«08» квітня 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТА(КИ)

Шовкопляс Владислав Андрійович
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) програмна система оптимізації керування персоналом приватної фірми
керівник проекту (роботи) Рябенко А. Є.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від «26» травня 2021 року №215
2. Строк подання студентом проекту (роботи) 08.06.2021
3. Вихідні дані до проекту (роботи) перелік літератури джерел згідно теми дослідження
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) У першому розділі було проведено аналіз предметної області та також було розглянуто стандарти для керування IT – проектами. Також були розглянути та вивчені аналоги програмного забезпечення. У другому розділі було розглянуто програмне забезпечення для створення додатку, язики програмування та також середовища для розробки. Основна мова для програмування була використана PHP, на ній будується основне тіло проекту. Для збереження даних необхідних для коректної роботи проекту використовувалась мова SQL. Для більш комфортного відображення даних для користувача використовуються мова програмування XML та її у поміч HTML, вони призначені для збереження даних та відображення на веб сторінці. У третьому пункті було розглянуто як безпосередньо буде будуватися програма. У четвертому пункті було розроблено безпосередньо сам додаток.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконання завдання
1	Рябенко А.Є., к.ф.-м.н., доц.		
2	Рябенко А.Є. , к.ф.-м.н., доц.		
3	Рябенко А.Є., к.ф.-м.н., доц.		
4	Широкорад Д.В., к.ф.-м.н., ст. викл		

7. Дата видачі завдання «8» жовтня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Сформулювати мету та основні завдання дипломної роботи	08.10.2020 – 30.10.2020	
2	Опрацювати літературу та існуючі аналогів програмного забезпечення	02.11.2020 – 20.11.2020	
3	Розробка програмної реалізації для вирішення задачі	20.11.2020 – 19.03.2021	
4	Тестування програмного забезпечення	19.03.2021 – 30.04.2021	
5	Оформлення пояснювальної записки	03.05.2021 – 24.05.2021	
6	Попередній захист дипломної роботи та отримання рецензій.	25.05.2021 – 07.06.2021	
7	Захист дипломної роботи.	08.06.2021	

Студент _____ Шовкопляс В. А.
(підпис) (прізвище та ініціали)

Керівник проекту (роботи) _____ Рябенко А. Є.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

ПЗ: 92 с., 27 рис., 2 дод., 24 джерел.

Мета роботи – створення системи підтримки керування IT-проектів з розширеним функціоналом.

Для досягнення поставленої мети необхідно виконати наступні задачі:

дослідити предметну область;

виконати аналіз прогресу розробки;

–розробити структуру застосунка;

–виконати програмну реалізацію застосунка;

–оформити пакет документації на створений програмний продукт та провести його тестування.

Програмне забезпечення створено з використанням технології NW.js, сумісне з x86- та x64-бітними платформами на базі ОС Windows, Mac OS X та Linux. Система має зручний, максимально орієнтований на користувача інтерфейс.

PHP, MySQL, XML, WEB, CSS, XHTML, HTML

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ ТА ТЕРМІНІВ

Office	Назва застосування
PHP	Серверна мова сценаріїв, яка вбудовується безпосередньо в HTML-код.
PostgreSQL	об'єктно-реляційна система керування базами даних. Є альтернативою як комерційним СКБД, так і СКБД з відкритим кодом
NW.js	Дозволяє викликати всі модулі Node.js безпосередньо з DOM і створення програмного забезпечення з усіма веб-технологіями
HTML	стандартна мова розмітки веб-сторінок в Інтернеті. Більшість веб-сторінок створюються за допомогою мови HTML
DOM (Document Object Model)	Специфікація прикладного програмного інтерфейсу для роботи зі структурованими документами
CSS	каскадні таблиці стилів, спеціальна мова, що використовується для відображення сторінок, написаних мовами розмітки даних. Найбільш часто CSS використовують для візуальної презентації сторінок, написаних

	HTML та XHTML
MVC (Model-View-Controller)	Архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення
ОП	Оперативна пам'ять
ОС	Операційна система
ПК	Персональный комп'ютер
ПЗ	Програмне забезпечення
СКБД	Система курування базами даних
ТЗ	Технічне завдання
Distributed Replicated Block Device(DRBD)	Програмна система, що забезпечує синхронізацію між локальним блоковим пристроєм та віддаленим
DSWHC	DISTRIBUTION SYSTEM OF WORKING HOURS IN COMPANY

ВСТУП

Збільшення доходів підприємства приносить впорядкування бізнес-процесів у виконанні проектів. Керування проектом — це методологія організації, планування, керівництва, координації трудових, фінансових та матеріально-технічних ресурсів протягом проектного циклу, спрямована на ефективне досягнення його цілей шляхом застосування сучасних методів, техніки та технології керування для досягнення певних результатів щодо складу та обсягу робіт, вартості, часу, якості та задоволення учасників проекту. Стан проекту від ідеї до завершення характеризується зміною низки показників, що визначають його сутність і на основі яких встановлюється успішність проекту. Ця сукупність елементів проекту по суті є об'єктами керування .

Для ефективного керування проектами є продуктивним створення багатофункціональної інформаційної системи наповненням блоками, що відповідають структурі організації, поставленим цілям, кількості та професіоналізму персоналу, може включати підсистеми бухгалтерії, підсистему нормативної документації, підсистему взаємодії між різними відділами, підсистема оптимізації робочого часу співробітників й аналізу ефективності роботи працівника.

Наповнення інформаційної системи визначається предметною областю та потребами організації. Для успішної реалізації ІТ-проектів є ефективними різні методології керування в залежності від розмаху проекту, потужностей фірми-виконавця, професійної підготовки персоналу. Системи, що допомагають в поставленому завдані є системами за типом task manager, тобто програмами керування проектами, які дозволяють централізовано керувати задачами та їх вчасним виконанням, зберігання інформаційних ресурсів; програму аналізу продуктивності працівника; робити знімки екрану і веб-камери; автоматично розраховувати погодинну зарплату; створювати завдання і проекти й бачити не тільки статус «закрито», а також — статистику виконання;

враховувати час витрачений на перерви, наради й роботу за межами комп'ютера, допомагають проводити розподіл робіт між розробниками, своєчасне тестування, облік відпрацьованого часу, оцінку ефективності роботи працівника та нарахування зарплатні.

Ефективність роботи працівника залежить від кваліфікації, завантаженості, складності завдання, відповідності завдання та області знань та вмінь працівника, від вмотивованості, від інноваційності поставленого завдання. Моніторинг робочого часу співробітників проводиться за допомогою підсистем тайм-трекерів, які дозволяють визначати час, необхідний для виконання певних завдань і контролювати знаходження співробітника на робочому місці. Аналіз накопиченої інформації дозволяє корегувати робочі процеси.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Керування проектами як засіб підвищення доходності підприємства

Проект - це тимчасова діяльність, спрямована на створення унікального продукту чи послуги. Менеджер проекту (МП) - це особа, яка працює в компанії або представник, який ініціює проект. Керівник проекту контролює виробництво товарів або послуг на всіх рівнях, своєчасно контролює виконання обов'язків та забезпечує досягнення результатів певного товару чи послуги.

Деталі управління проектами визначаються предметною областю, яка являє собою поєднання набору послуг та продуктів, реалізованих в рамках проекту. Теми проекту включають результати, цілі та робочі структури. Під час виготовлення проекту на всіх етапах відбуваються зміни.

- у результаті вашої роботи мета та структура можуть зазнати значних та незначних змін на різних етапах проекту.

- ви можете змінити обсяг роботи під час реалізації проекту, а в процесі безпосереднього впровадження перейти від основного до більшого в кінці роботи.

Управління самою тематичною областю передбачає управління цими змінними протягом циклу розробки проекту. Це робиться за допомогою валідації, планування, вдосконалення та валідації ділянки проекту.

Контроль якості в проектах-Контроль якості, Забезпечення якості, Планування якості. Важливість підходу до контролю якості проекту охоплює весь спосіб життя, всі його аспекти та аспекти проекту. Це включає: Організаційні рішення, структура та управління. Сировина, доступні матеріали, якість роботи, виконаної під час реалізації проекту nk. Отримайте кращі результати проекту (послуги, продукти). Контроль якості здійснюється шляхом дотримання вимог до якості в процесі проектування проекту за допомогою

стандартів якості та результатів проекту, планування якості, систем підтримки та контролю якості.

Методи управління проектами визначають діяльність команди управління проектами. Усі методи контролю розділені на такі основні розділи:

- управління предметною областю проекту (на основі вмісту).

- короткострокове управління ресурсами (часовий бюджет);

- управління ризиками (зниження рівня невизначеності проекту).

- контроль якості (стандарти та вимоги);

- значення методу управління та включення (фінансовий та матеріальний бюджет).

- Управління персоналом проекту (організація, підготовка, призначення роботи).

- управління комунікацією (прогнозування та моніторинг прогресу та результатів кар'єри).

- управління контрактами (виконавець, передача обладнання тощо).

Кожен метод управління має п'ять незалежних видів управління діяльністю: контроль, активація, координація, організація та планування. Кожна перша діяльність є необхідною умовою наступної діяльності. Ці п'ять типів реалізуються послідовно, поки певний метод не буде повністю реалізований. Іншими словами, ступінь повноти впровадження методу управління залежить від складності управлінської діяльності.

До методів управління часом належать: робота та тривалість, характер початку та закінчення проекту, його складові, найважливіші (контроль) подій, кожна виконана робота, вдосконалення часових характеристик (Скорочення). Ефективно використовуйте економію часу та контролюйте хід проекту за його своєчасними характеристиками. Прогнозні умови завершення робіт, проектів та загальних фаз. Прийміть рішення про усунення найгіршого втраченого часу. Підхід до управління часом активується за допомогою таких процесів, як аналіз проекту та матеріали, графіки роботи календаря, управління графіком виконання робіт, оновлення та коригування..

Методи управління витратами включають планування ресурсів, початкові кошториси, пов’язані з проектами, вищі ціни, грошові потоки, прогнози прибутку, перевірку та отримання цін, а також коли перевищені обмеження цін та інші обмеження фінансових прогнозів. Містить рішення. Найважливіша роль контролю за витратами - це дотримання бюджетних обмежень проекту та отримання вигоди від його реалізації. Управління ризиками - це система поведінки для прогнозування, аналізу, оцінки та попередження несприятливих подій. Вжити заходів для зниження рівня ризику під час реалізації проекту та розподілу втрат ризику серед учасників проекту.

Підхід до управління людськими ресурсами до проекту передбачає визначення потреб у персоналі, кількості та кваліфікації протягом усього періоду проекту. Шукайте та підбирайте кандидатів, реєструйте та звільняйте служби. Організувати та розгорнути персонал на робочому місці. Підвищення кваліфікації та організація навчання. Встановіть підзвітність; працюйте разом над створенням середовища та робочого середовища. Запобігання та вирішення нових конфліктів. Управління людськими ресурсами, наприклад, питання оплати праці, має зосереджуватись на оптимальному використанні людських ресурсів для досягнення цілей проекту.

Система управління та розподілу проектних контрактів складається з процесу вибору стратегії контрактної діяльності. Новини та рекламні робочі місця; Визначте структуру, посилання та умови праці, що входять до контрактів на дослідження. Підготовка контрактних пропозицій; Торги, конкурс, торги, вибір підрядників та розповсюджувачів за допомогою н.к. Висновок та розрахунок виконаних контрактів.

Як управляти комунікацією проекту, включаючи розробку, організацію, сертифікацію розробки та обмін інформацією різними способами. Це адміністративне завдання, як правило, включає процес збору, передачі, відбору, перегляду та інтерпретації достатньої кількості інформації, яка підходить для всіх учасників проекту та їх середовища. Одночасно розглядаються та розрізняються два типи обміну інформацією. Формальні (завдяки

використанню інформаційних технологій) та неформальні (завдяки особистому спілкуванню та різного роду зустрічей). Управління комунікаціями в проекті тісно пов’язане з управлінням персоналом.

Зміни в управлінні - це комплексний інтеграційний процес, який враховує всі внутрішні та зовнішні аспекти проекту та впливає на зміни в проекті. Визначте вже внесені зміни; сплануйте вплив на майбутні проекти. Контролюйте зміни проекту та координуйте зміни всього проекту. [25]

1.2 Стандарти в галузі керування проектами

Найпоширенішими організаціями, що розробляють міжнародні стандарти в галузі управління проектами, є:

• Міжнародна організація зі стандартизації ISO опублікувала ISO 10006 "Система управління якістю. Керівні принципи контролю якості проекту".

• Міжнародна асоціація управління проектами (IPMA) - це поважний професійний орган у галузі управління проектами, який об'єднує 45 національних асоціацій. Україну представляє IRMA Національний інститут управління проектами UKRNET (UPMA).

• Інститут управління проектами (PMI). Члени RMI є експертами з управління проектами по всьому світу і мають філії в різних країнах. RMI повністю розробляє стандарти в галузі управління проектами.

Загалом, ви можете розрізнити такий стандартний перелік та розширену географію, що використовується.

Стандарти ISO - Сімейство стандартів системи управління якістю, ISO 10006: 2003, системи управління якістю та найвищі керівні принципи управління проектами. Стандарт ISO 10006 доповнює раніше опублікований набір стандартів і поширює основні принципи контролю якості безпосередньо на управління проектами. Він базується на моделі процесу управління

проектами та використовує основні методи та принципи стандарту RMVOC 1996 року.

IRMA формує WIS (на основі потужності IRMA). WIS - це документ, який містить вимоги до здібностей керівника проекту.

Посібник з управління проектами знань (PMI). Його вважають одним із найвпливовіших людей у галузі управління проектами. RMVOC містить добре розроблену інформацію про процес управління проектами, інформацію про інструменти управління проектами. На сьогодні опубліковано три основні стандарти, які контролюють проектні процеси, програми, файли проектів та понад 10 додаткових стандартів. Додаткові критерії включають вимоги до інших методів управління проектами (розробка ієрархічного ядра (ISR), планування, управління ризиками тощо), використання управління проектами для конкретних типів проектів (управління будівельними проектами) та управління державними проектами тощо. . PRINCE2 (Проект в контрольованому середовищі) - Проект в контрольованому середовищі. Цей підхід, призначений для управління проектами та проектними групами в рамках організації, є структурованим підходом до управління проектами та затверджений урядом Великобританії як стандарт управління проектами на соціальній арені. PRINCE2 включає методи управління, контролю та планування проектів.

Open GAPPS (iGlobal Alliance for Standards Standards Standards) визначає навички керівників проектів та програм. Ключові компоненти GAPPS: Стандарт 2006 року - це детальний опис шести областей компетенції, кожна із яких визначає конкретну сферу професійної діяльності менеджера проекту, визначає ключові вимоги та робить менеджера здібностей таким.

Приклад потенційного зрілості Інституту інженерів програмного забезпечення (США). Це стосується управління проектами в галузі розробки програмного забезпечення. У рамках цієї моделі ви можете порівняти кожен бізнес з певним рівнем (п'ята частина можливостей), продемонструвавши якість розвитку програмного забезпечення. В даний час останньою версією є SMMi-1.

Система управління загальними витратами - Міжнародний метод AACE. Це інтегрований процес або підхід до управління портфелями, планами та проектами. Розроблена карта розвитку, яка окреслює кожну область практики підвищення цінності стосовно інших сфер діяльності та суміжних сфер.

Логічний системний підхід (LFA-Logical Structure Approach) - Підхід до проектування, заснований на системній структурі процесу ідентифікації, планування та управління проектами. Є аналітичним інструментом, який використовується багатьма агенціями з розвитку для надання двосторонньої та двосторонньої допомоги міжнародним та неурядовим організаціям у галузі управління проектами розвитку.

Microsoft Solution System (MSF) - це метод розробки програмного забезпечення, розроблений корпорацією Майкрософт. MSF базується на практичному досвіді Microsoft та описує управління персоналом та робочий процес у процесі розробки рішень. Це ефективний набір концепцій, моделей та правил.

Oracle Application Process (AIM) Oracle Application Process (OrIM) Процес подання заявки, який реалізує заздалегідь розроблену програму OracleE Business Suite, розроблену Oracle, яка зосереджена на впровадженні бізнес-процесів. Це детальний опис роботи, виконаної під час проекту, що демонструє відповідальність процесу впровадження та роль команди проекту. Кожен процес створює основний (нерозділений) обсяг роботи, який потрібно припинити з офіційно встановленим результатом.

P2M (японський національний стандарт). Підхід P2M залежить не від орієнтації на продукт або процесу, а від організаційного вдосконалення в результаті реалізації проекту. Коротше кажучи, ця методика показує, як ви можете використати досвід, накопичений у реалізації проектів розвитку бізнесу.

Крім того, існують деякі непопулярні національні стандарти, які можна використовувати для збалансування процесів управління проектами та програмами.

• Управління проектами НАСА (США).

• BSIBS 6079 (Великобританія);

• Орган Центру знань (Великобританія).

• OSCEng (Великобританія);

• DIN 69901 (Німеччина);

• V модель (Німеччина).

• VZPM (Швейцарія);

• AFITEP (Франція);

• Вулиця Гермес (Швейцарія);

• ANCSPM (Австралія);

• CAN / CSA-ISO 10006-98 (Канада);

• C-PMBOK (Китай);

• NQF4 у Південній Африці (ПАР).

• CEPM (Індія).

• PROMAT (Корея).

Україна не має національних нормативних стандартів, але деякі стандарти IRMA, PMI та P2M широко використовуються. [2-4]

1.3 Автоматизація керування проектом

Роль управління IT-проектами полягає у наданні користувачам широкого спектру проектної діяльності наступними способами: - Опис параметрів проекту та встановлення логічних зв'язків між завданнями. - Проектуйте презентації на декількох рівнях. - Створіть список доступних ресурсів та вкажіть обладнання та ціни, кількість робочих місць тощо. -- Календар і графік. - Планування ресурсів та ціноутворення. - Представлення зображення дизайну проекту (діаграма Ганта, діаграма PERT); - Керування

розробкою проекту. - Створіть звіт, що відображає хід проектної роботи. - Організація зв'язку (робота в мережевому середовищі).

Інформаційна система управління проектами (СУІБ) - це організаційно-технічний комплекс технологій, технологій, програмного забезпечення та інформаційних засобів, спрямованих на підтримку та підвищення ефективності процесів управління проектами. Ядро ISUP - це єдиний інформаційний простір, який забезпечує: - Єдина база даних планів для всіх корпоративних проектів. -- Одне керівництво та ресурс, використані в проекті. --Уніфікований тип документа, шаблону проекту, звіту. - Єдина база даних документів для всіх корпоративних проектів. IMS призначений для впорядкування роботи, пов’язаної зі збором, обробкою, аналізом, аналізом та прогнозуванням, оцінкою альтернативних структурних рішень та вибором найкращих з них для реалізації проекту. ISUP, як правило, складається з двох компонентів: розподілу та функціональності. Розділ підтримки складається з наступних основних розділів: - Організаційна підтримка - сукупність документів, що описують організаційну структуру, права та обов'язки працівника. - Юридична допомога - Сукупність правових принципів, що регулюють правовідносини при формуванні та функціонуванні системи. - Технічна підтримка - Сукупність технічних методів, призначених для роботи систем, комп’ютерів, зв’язку та організаційного обладнання. Інформаційна підтримка - набір форматів документів, класифікацій, правил та рішень, що застосовуються щодо обсягу, розміщення та складу інформації, що використовується в системі. - програмне забезпечення - набір програм та відповідної експлуатаційної документації, призначених для управління обладнанням та вирішення експлуатаційних проблем. Автоматизовані цілі розроблені для: належного управління проектами та створення програмних ролей, що забезпечують управління проектами протягом життєвого циклу від завершення до завершення.

Робоче місце складається з об’єктів (підсистем), які визначають його призначення, метод управління та метод обробки інформації. Функціональні компоненти ISUP розглядаються як модель управління проектами. Це залежить

від трьох матеріалів: структури проектної роботи, структури ресурсів та матриці розподілу ресурсів для проектної роботи. IMS можна організувати наступним чином: Метод управління проектами Стандарт управління. Ви можете виділити принаймні три рівні управління в структурі вашої проектної організації. Стратегічний рівень управління дверима проекту (топ-менеджмент). Рівень управління проектами (управління проектами). Рівень реалізації проекту (проектна група).[1]

Деякі функції управління проектами включають програмні продукти, які становлять основу інтегрованої системи управління бізнесом. Система прикладних програм, яка підтримує роботу менеджера проекту на всіх етапах життєвого циклу проекту, - це автоматизована система, яка фокусується на початкових стадіях проекту (етапи визначення, планування проекту, структура системи) та напрямок реалізації (так званий менеджмент). Його можна розділити на. Впровадження) або так званий проектний менеджмент). Практики управління проектами використовують як універсальні програмні пакети, так і спеціалізовані програмні пакети. Для складання документів і розрахунків використовуються універсальні програмні пакети, що включають текстові процесори та програмне забезпечення для роботи з електронними таблицями та базами даних. Ця ж група має програми для підготовки та реалізації презентаційних та комунікаційних програм. Додаток для роботи з електронною поштою, факсом, доступом та публікацією в Інтернеті. Рівень автоматизації використання цих програм багато в чому визначається природними особливостями цих програм, такими як можливість використання шаблонів та наявність вбудованих мов програмування. Спеціальні програми можна розділити на конкретні та «персональні». Спеціальні програми спеціально розроблені для вирішення проблем та управління проектами. Ця група планів включає плани бізнес-планів. Ці два типи програмного забезпечення часто використовуються на практиці. Група "Спеціальні" містить програмні пакети, призначені для вирішення таких проблем, як моделі, але вони корисні для вирішення таких проблем, як структурний аналіз та управління проектами.

Спеціальне програмне забезпечення (системи інструментів) включає програмні пакети, спеціально розроблені для управління проектами або настроювані для вирішення цих особистих проблем. Класифікація певного програмного пакету базується на обсязі проекту та характеристиках керівника проекту, який є користувачем. На основі запропонованих критеріїв класифікації систему управління проектами (пакет) можна розділити на досвід та початковий рівень.

Критерій оцінки доцільності програмного забезпечення в системі управління інформацією. Це пов’язано з вимогами програмного та апаратного забезпечення, інтеграцією з іншими програмами. Критерії, пов'язані з ціною програми (Software Bike): придбання, встановлення, оплата технічної допомоги, підтримка загальної експлуатації [5-8]

1.4 Особливості керування IT проектами

Характер управління проектами в IT-галузі визначається тим, що асортимент товарів і послуг асоціюється з невидимими сферами. Частина досліджень полягає у створенні програмного забезпечення або продуктів обслуговування в галузі інформаційних технологій, що визначають структуру роботи. Управління IT-проектами використовує як поведінковий підхід, так і "прості" методи та технології управління проектами. Управління ризиками є важливою частиною процесу управління проектами, оскільки існує велика частка невизначеності в проектах IT-проектів та реалізації проектів. Планування та реалізація IT-проектів відбувається за невизначених обставин, спричинених змінами у внутрішньому та зовнішньому середовищі.

Управління IT-проектами включає такі етапи:

- Дослідження потреб клієнтів

--Формування бізнес-цілей

- Оцінка наявних ресурсів та технічне обґрунтування

--Аналіз ефективного використання існуючої інфраструктури для реалізації нових цілей

- Визначення зацікавлених сторін у реалізації проекту (кінцевих споживачів, менеджерів, інвесторів),

- Створіть групу проектів.

- управління проектами (розробка, впровадження, тестування тощо),

--Підтримка та поради.

Тому найважливішими блоками управління проектами є:

- Керування часом (термін повинен бути дотриманий)

- Контроль витрат (необхідний бюджет)

- Управління вмістом (зрозуміти вимоги замовника та правильно впровадити)

- Управління якістю

- кризовий менеджмент

Management Управління закупівлями (модернізація IT-інфраструктури)

- Управління людськими ресурсами

--Комунікаційний менеджмент

- Інтегрований менеджмент.

Під час розробки проекту в IT-галузі створюється комплексний коефіцієнт (TOR), прийнятний для клієнта. Цей документ містить ключові обов'язки, вимоги до кінцевих результатів та терміни реалізації проекту. Саме ТК визначає та виправляє помилки на етапі планування та забезпечує відповідність результатів розробки потребам замовника. [9]

Економічною ефективністю інформаційної системи вважається взаємозв'язок між ефективністю впровадженої інформаційної системи, результатом її впровадження та ціною на досягнення запланованого результату. Економічну ефективність IT-проекту можна розрахувати на різних етапах життєвого циклу інформаційної системи. Тобто розробка, проектування, проектування, впровадження та продуктивність інформаційної системи.

Виходячи з цього, ви отримаєте різні розрахунки, як це зробити та деякі деталі. Залежно від життєвого циклу IT-проекту, існуючі методи оцінки можна розділити на дві групи залежно від методу.

Майбутні підходи - комбінувати методи оцінки прямих наслідків впровадження інформаційної системи на етапі ефективності системи. Ці підходи зосереджуються на різних пріоритетах до та після впровадження інформаційної системи та порівнюють результати та зусилля, використані при реалізації проекту впровадження системи.

Перший метод - поєднує процес прийняття рішень із методами оцінки та прогнозування результатів впровадження інформаційної системи в координації інвестицій. Ці методи використовують ключові прогнози, встановлені вбудованою моделлю, і можуть враховувати різні типи ризиків, що впливають на результат та ціну системи впровадження інформаційної системи, а також абсолютно різні можливості.

Майбутні методи можуть бути використані для аналізу ефективності виділених коштів та прийняття рішення про продовження чи завершення IT-проекту з метою оцінки його ефективності на етапі експлуатації IT-проекту. Зняття з експлуатації (якщо застосовується) Оцінка економічної ефективності на основі майбутнього підходу передбачає порівняльний аналіз фактичних цін, впливу обізнаності щодо проекту та фактичного досягнення цілей IT-проекту. За допомогою цього методу ви можете визначити відхилення від фактично запланованої ціни для виявлення економічно неможливих інвестицій у розробку та реалізацію вашого IT-проекту. Насправді методи, орієнтовані на майбутнє, забезпечують високий рівень точності оцінки ефективності IT-проектів, але забезпечують оцінку та впровадження IT-проектів після впровадження, коли витрачаються певні кошти. З різних способів оцінки ефективності інформаційного проекту найважливішим способом оцінки такої ефективності до самого проекту є стадія техніко-економічного обґрунтування, перший метод. Сюди входять фінансові, евристичні та можливі методи: IRR (внутрішня норма прибутку), ROI (рентабельність інвестицій), TEI (загальний

економічний вплив), NPV (поточна вартість), BSC (оцінка) Збалансована оцінка), EVA (збільшення економічної ставка). Слід зазначити, що найважливішим фактором використання цих методів є оцінка очікуваного ефекту від впровадження IT-системи. Це не гарантує надійність розрахункової кількості прямих вигод і є джерелом ризику.

Ризики IT-проекту: • Недоліки планування. • Продаж персоналу. • Вимоги до інфляції. Порушення вимірювань. • Обмежене виробництво.

Спосіб реалізації IT-проектів характеризується багатьма формальними процесами та їх детальним оглядом. Сучасні технології IT-проектування представлено на рис 1.1.

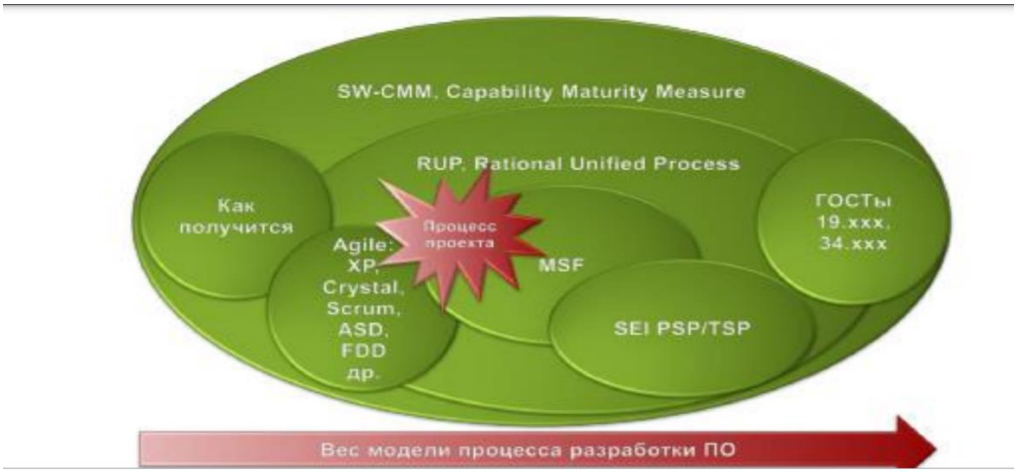

Рисунок 1.1 — Сучасні методології IT-проектування

Критерії.

SW-CMM. У середині 80-х років Пентагон зосередився на тому, як вибрати розробників програмного забезпечення для великих програмних проектів. Лабораторія інженерії програмного забезпечення, яка входить до Університету Карнегі-Меллона, представлена армією, розробила SW-CMM, модель зрілості програмного потенціалу, як еталонну модель для програмування розробки програмного забезпечення. [9-11]

Ця модель розпізнає п’ять рівнів розвитку при розробці програмного забезпечення.

1. Раніше - процес розробки був хаотичним. Він зосереджений лише на створенні команди спеціальних художників.

2. Огляд-Визначається основний процес управління проектами (ціна, правила, ефективність). Спрощення цих процесів є важливим для імітації минулих успіхів у таких проектах.

3. Визначення - Процеси розробки програмного забезпечення та управління проектами визначаються та реалізуються в єдиній структурі процесів компанії. Усі проекти використовують стандартні процеси розробки програмного забезпечення та процеси організаційної підтримки конкретних проектів.

4. Управління - Збір детальних даних про ефективність процесу розробки та якість кінцевої продукції. Значення та динаміка цих даних буде проаналізовано.

5. Удосконалення - Стале вдосконалення процесу базується на кількісних даних про процеси та нових ідеях та запуску технологій. Повний профіль SW-CMM займає приблизно 500 сторінок і визначає набір з 312 вимог, яким повинна відповідати організація, якщо бажає пройти сертифікацію на цьому рівні на рівні зрілості 5.

RUP-Rational Unified Process (RUP) був розроблений Філіппом Крухтеном, Іваром Якобсоном, іншими співробітниками програм персоналу та мовою моделі UML. Модель RUP описує загальний процес, за допомогою якого організація або команда проекту створює конкретний процес, який враховує їх потреби. Найбільш критикується ця межа RUP. Це може бути що завгодно і не є надійним. Завдяки цій загальній структурі, RUP може бути використаний як основа для традиційних моделей розвитку водоспаду та як проста розробка. [12-13]

MSF. Система рішення Microsoft (MSF) - це проста та проста у використанні модель, заснована на ітераційній розробці. Цікавою особливістю MSF є готовність створити хорошу неформальну команду проектів. Для

досягнення цієї мети MSF надає нетрадиційний підхід до організаційної структури для розподілу принципів відповідальності та взаємодії в команді.

Одна з останніх розробок Інституту розробки програмного забезпечення PSP / TS / Програмне забезпечення для командних процесів. Процес особистого програмування визначає вимоги до можливостей розвитку. Залежно від моделі кожна програма може: Орієнтуйтеся на час, витрачений на проект. Розгляньте виявлені дефекти; класифікуйте типи дефектів. Оцініть обсяг роботи; використовуйте систематичний підхід для пояснення результатів випробувань. Завдання планування; насиченість часом і графіком. Робіть особисті огляди проектів та архітектури. Перевірте окремі номери. Виконайте ретроактивний тест. [14]

Процес командного програмування базується на групі з 3 до 20 розробників, які керують собою. Команда повинна зробити наступне: Складіть процес і план. Обов'язок; підтримувати максимальну мотивацію та продуктивність. Завжди використання моделі PSP / TSP робить п’ятий рівень ШМ загальним у вашій організації. Agile Основна ідея всіх простих моделей полягає в тому, що процеси, що використовуються при розробці програмного забезпечення, потребують модифікації. Вони стверджують, що їх найвища цінність базується на людях та їх взаємодії, а не на процесах чи методах. По суті, так звані прості прийоми - це не прийоми, а набір практик, якими можна (а може і не досягти) гарної розробки програмного забезпечення на основі ітеративного зростання, управління командою та гнучкості розробки. [15]

Модель моделі розвитку IT-проекту обирається на основі технічної роботи, кількості та професіоналізму розробника, і кожен проект має техніку.

1.5 Огляд аналогів

1.5.1 Simple Time Track

Simple Time Track являє собою розширення для Chrome. Після установки воно додає свою іконку біля правого краю панелі інструментів браузера. Натискання на неї викликає інтерфейс тайм-трекера: зверху кнопки для створення, експорту та видалення завдань, нижче — список завдань, створених раніше.

Рисунок 1.2 – Приклад додатку Simple Time Track

Щоб почати облік часу, потрібно додати нове завдання і запустити таймер. Відлік буде йти до тих пір, поки таймер не зупиниться. У майбутньому до будь-якої із зупинених завдань можна повернутися і запустити таймер знову. Додатковий час додається до того, що було підраховано раніше. Це може знадобитися в тому випадку, якщо завдання неможливо виконати за один присід і доводиться перериватися, щоб зайнятися чимось іншим.

Більш просунуті тайм-трекери дозволяють вивчати історію виконання завдань, будувати складні звіти з обмеженнями за часом і іншим критеріям і навіть генерувати рахунки для відправки замовникам. Simple Time Track вміє тільки рахувати час. Результати можна подивитися в тому ж списку або експортувати їх в файл CVS, щоб перенести, наприклад, в електронну таблицю. Найпростіші потреби задовольнить і це, але якщо потрібно щось більше (а це, швидше за все, так), то краще знайти альтернативу серйозніше.

1.5.2 TimeEdition

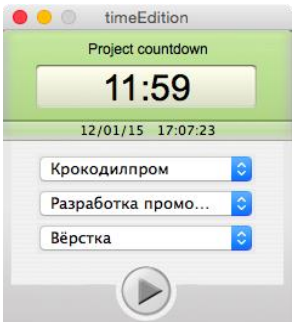

Рисунок 1.3 – Приклад додатку TimeEdition

Німецький опенсорсний timeEdition, пристосовано до певного робочого прогресу, який укладається в певні рамки, і в цих рамках, ймовірно, зручний. Людям, для яких вона розроблена, мабуть, доводиться мати справу з декількома замовниками. При цьому проекти, які вони для них реалізують, однотипні і складаються з одних і тих же етапів. Інтерфейс основного вікна timeEdition лаконічний. Він складається з великої кнопки, яка запускає і зупиняє таймер, самого таймера і трьох меню для вибору замовника, проекту і завдання, до яких відноситься цей таймер. З цими меню і пов'язана основна складність. Просто запустити таймер і почати роботу не вийде. Спочатку ці меню потрібно заповнити, додавши до програми інформацію про замовника, про проект, а також про всі можливі завдання, над якими передбачається працювати. Тільки після того, як все це зроблено, можна починати відлік часу. Справа ускладняється тим, що завдання, по суті справи, загальні для всіх проектів.

На відміну від Simple Time Track, timeEdition записує не тільки сумарний час, витрачений на роботу. Додаток запам'ятовує початок і кінець кожного відрізка часу, протягом якого користувач працював над завданням. Це дозволяє обчислювати тривалість роботи, виконаної протягом дня, тижня чи місяця. Для отримання цих відомостей служить окреме діалогове вікно. Крім того, їх можна імпортувати в Excel.

Цікава особливість timeEdition: з додатком можна доручити на льоту переносити інформацію про врахований часу в електронний календар —

наприклад, в Outlook або в Google Calendar. Це ще один альтернативний метод перегляду інформації, накопиченої в timeEdition.

1.5.3 Toggl

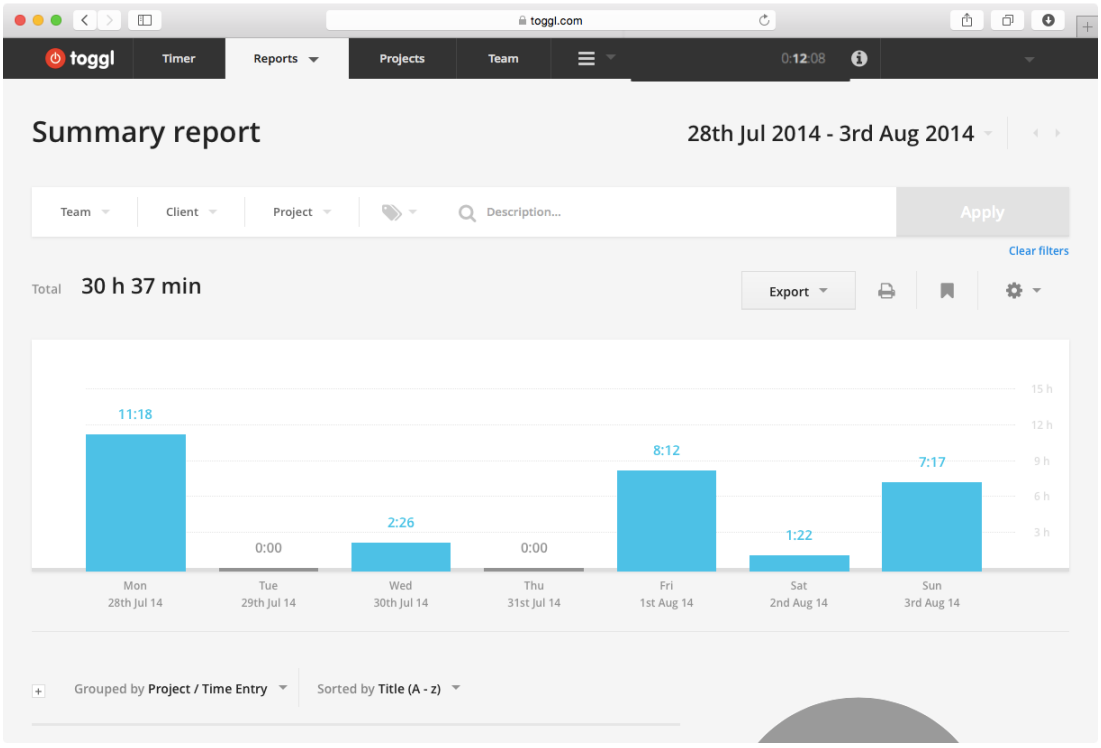

Рисунок 1.4 – Приклад додатку Toggl

Toggl набагато ліберальніше, ніж timeEdition, і не намагається загнати користувача в відомі, але тісні для багатьох рамки. У цьому сенсі він ближче до Simple Time Track, тільки вже без зайвого примітивізму. Toggl веде свого роду журнал активності користувача і вміє складати звіти з відбором завдань за заданими критеріями. Це не саме функціональне, не найзручніше і не найприємніше засіб обліку робочого часу, але воно має резонним мінімумом можливостей і надає їх безкоштовно. Це рідкість: за серйозні тайм-трекери доводиться платити.

Сам Toggl — це веб-сервіс, але до нього додається опціональний додаток під всі популярні платформи. Веб-сервіс відповідає за облік, звіти і

налаштування. Додаток же спрощує створення нових задач, а також запуск і зупинку таймерів. Крім того, воно додає кілька можливостей, які неможливо реалізувати в інтернеті. Зокрема, воно помічає, коли користувач занадто довго не діє, і пропонує зупинити таймер.

Для кожної з задач можна вказати замовника і проект (втім, якщо не потрібно, то можна і не вказувати). Це можна використовувати при генерації звітів. Веб-сервіс дозволяє відфільтрувати записи за датою, замовнику і проекту.

Таким чином вдається дізнатися, скільки часу було витрачено протягом місяця на виконання проекту, або частку робочого часу, яку з'їли доручення певного замовника протягом тижня. Платна версія Toggl (вона обійдеться в 5 доларів на місяць) додає підтримку спільної роботи декількох користувачів і можливість задати рівень погодинної оплати (це потрібно для того, щоб підраховувати, скільки грошей принесуть відпрацьовані години).

1.5.4 Timely

Простота створення нових завдань робить Toggl схожим на планувальник навпаки: можна запускати окремий таймер для кожного кроку і отримувати в результаті найдокладніші записи про виконану роботу. Але ця подібність навряд чи було навмисним. З норвезьким веб-сервісом Timely все інакше: його автори свідомо робили гібрид електронного календаря і тайм-трекера. «Замість того щоб запитувати, що ви робили на цьому тижні, я питаю, що ви плануєте робити на цьому тижні», — пояснює автор Timely Матіас Міккельсен на сайті компанії. Передбачається, що користувач буде спочатку планувати завдання, а потім стежити за їх виконанням.

Основа Timely — це повнофункціональний веб-календар в дусі Google Calendar, причому вельми непоганий. Він помітно зручніше і, як не безглуздо це звучить, красивіше Гуглівского. Вражаюча плавність роботи, рідкісна для веб-додатків, теж йде йому в плюс. Timely цілком міг би замінити і Google Calendar, і Apple Calendar, і Outlook для багатьох користувачів. Втім, у тих, хто звик до іншого календарем і не потребує нового, залишається шанс обійтися без хворобливої міграції. Timely вміє автоматично завантажувати події з будь-якого календаря, що підтримує формат iCal. В цьому випадку планування можна продовжувати в іншому додатку, а Timely використовувати тільки для обліку часу.

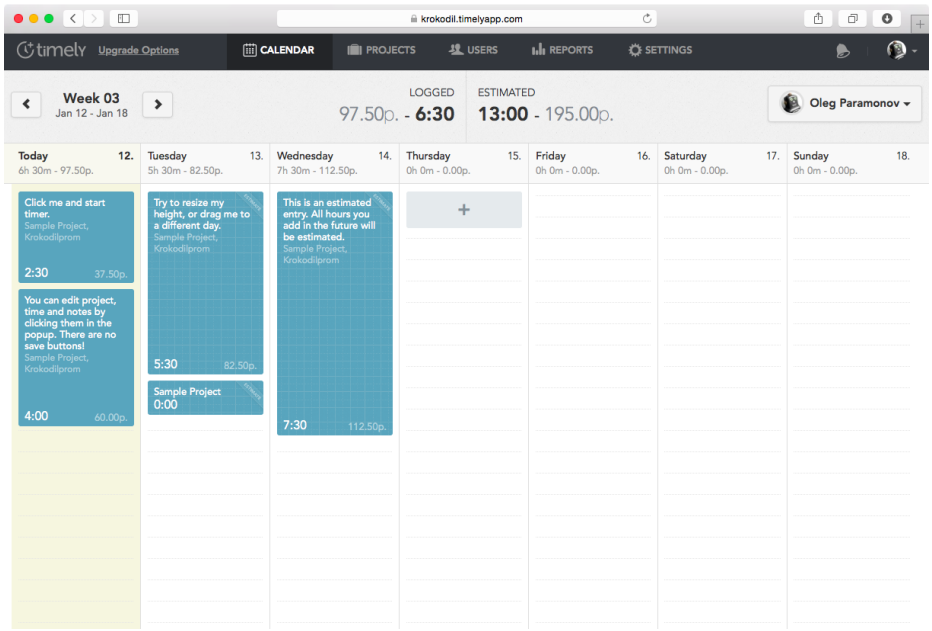

Рисунок 1.5 – Приклад додатку Timely

Кожне завдання, додана в календар, має вбудований таймер, який можна запустити, клікнувши по ній. Крім того, для кожного завдання ще на стадії планування можна встановити орієнтовний час виконання. Сумарне орієнтовний час виконання всіх завдань, що відносяться до проекту, і реальне відпрацьований час відображаються над календарем. Крім календаря, в Timely є сторінки, що дозволяють переглянути історію роботи над кожним проектом і історію роботи кожного користувача окремо.

Можливість спільної роботи декількох користувачів — важлива риса Timely. Цей сервіс цілком може стати зручним інструментом менеджера,

керуючого цілою командою, особливо якщо частина учасників проекту трудиться віддалено. За допомогою Timely легко дізнатися, хто і чим займається прямо зараз і на що йде час і гроші. Ще одна з можливостей, націлених саме на таке застосування, — це так званий «бюджет». Для кожного проекту можна вказати максимальний час виконання або максимальну суму, яку можна цініити на погодинну оплату працівників. Timely буде стежити за вичерпанням «бюджету» і попередить про наближення до фінішу.

Безкоштовна версія сервісу дозволяє одному користувачеві вести не більше трьох проектів. Фрілансерські версія коштує 14 доларів в місяць і знімає обмеження з кількості проектів. Крім того, є кілька тарифних планів, які розраховані на компанії і пропонують різні поєднання максимальної величини робочої групи і максимальної кількості проектів. Вони обійдуться в суми від 49 до 199 доларів на місяць.

1.5.5 Chrometa

Це один з небагатьох тайм-трекерів, який не дозволяє враховувати час, запускаючи і зупиняючи таймер вручну. Замість цього він використовує підхід, який кілька років тому популяризував сервіс RescueTime: спеціальна програма, встановлена на комп'ютері користувача, стежить за активними додатками і відкриваються документами, а потім зводить зібрану статистику в звіти, які допомагають зрозуміти, на що було витрачено час. Різниця в тому, що RescueTime був призначений для людей, які прагнуть підвищити власну продуктивність.

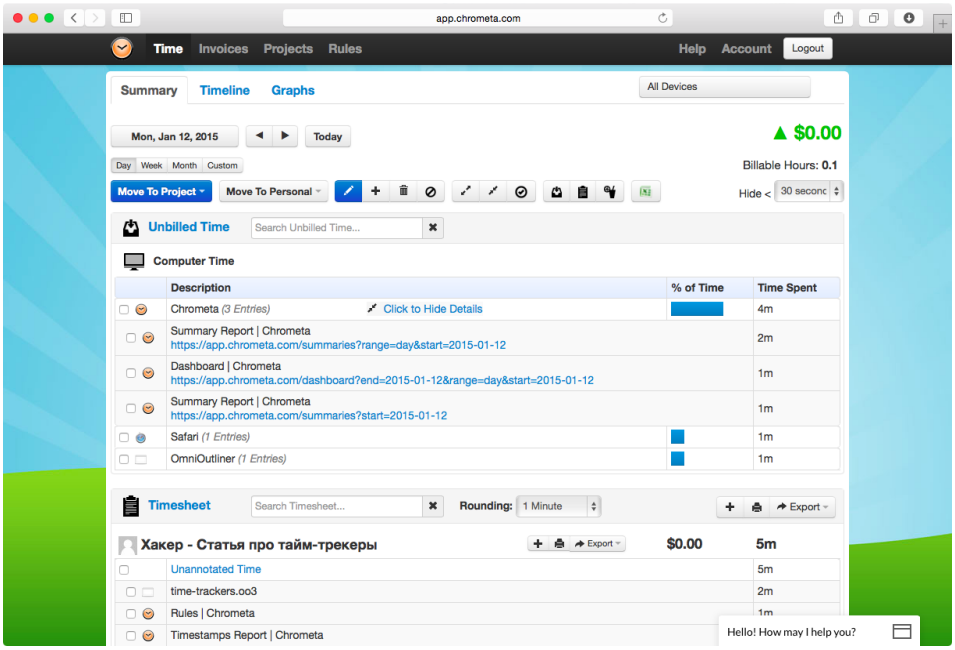

Рисунок 1.6 – Приклад додатку Chrometa

Він ділив сайти і додатки на корисні й марні і показував, куди витікає цінний час. Chrometa ділить їх за іншим принципом: за допомогою правил, що задаються користувачем, цей сервіс розподіляє завдання за різними проектами, а потім підраховує витрачений на них час і, якщо треба, генерує рахунки для виставлення замовникам. Всупереч очікуванням, це не особливо зручно. Без ручного сортування записів по проектам все одно не обійтися.

1.5.6 Harvest

Веб-сервіс Harvest схожий на timeEdition. З останнім його ріднить діловитість, обходиться, втім, без передозування бюрократії. Harvest здатний враховувати цінии, а також генерувати рахунки, причому по цій частині його можливості, мабуть, навіть ширше, ніж у румунського побратима. В Harvest, наприклад, можна призначати різну оплату для різних виконавців (як і Timely, Harvest дозволяє стежити за роботою цілої команди). Оскільки створювати завдання в міру потреби можна, в Harvest вони використовуються скоріше для позначення типу виконуваної роботи, а не для вказівки її мети.

Цільова аудиторія сервісу — це менеджери, яким потрібно дізнатися загальний прогрес по проектам і не особливо вдаючись у деталі. Безкоштовна версія Harvest існує, але виключно обмежена. За індивідуальної версію необхідно виплачувати 12 доларів в місяць. Командні версії коштують 49 і 99 доларів і розраховані на різну кількість користувачів (9 і 99 осіб відповідно).

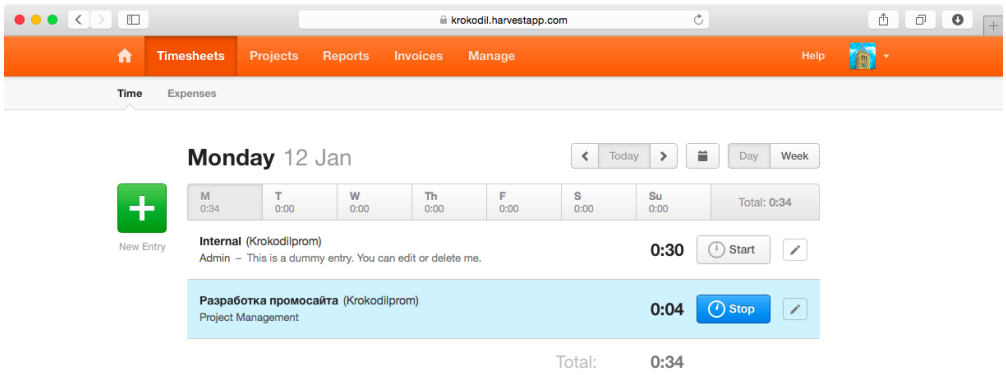

Рисунок 1.7 – Приклад додатку Harvest

1.5.7 Freckle

Freckle, один з кращий тайм-трекер. За різноманітністю можливостей, зручності і продуманості його можна порівняти з Timely. У Freckle, правда, немає вбудованого календаря-планувальника, але натомість цей веб-сервіс пропонує повні звіти і відсутню в Timely «бухгалтерську» функціональність.

Легкість створення нових завдань і обліку часу наближає Freckle до Toggl. І дійсно, цей сервіс можна використовувати для настільки ж детального обліку виконаних завдань. Однак не менш доступний і інший підхід — облік напрямків діяльності замість окремих завдань. Для цього достатньо вводити не опису конкретних завдань, а хештеги, що позначають, що саме ти робиш. Freckle їх потім сам вносить в збірну статистику.

Для того щоб скласти уявлення про те, як йде робота, у Freckle не потрібно переходити на сторінку звітів. Інформація про відпрацьований час

представлена прямо на головній сторінці у вигляді таблиці з круговою діаграмою на кожен день. Діаграми показують, як робочі години розподілилися за різними проектами. Звіти, в свою чергу, ефективно зводять на одній сторінці інформацію з різних джерел.

Інструменти для виставлення рахунків, вбудовані у Freckle, демонструють, як уникнути ускладнення інтерфейсу, яке тягне на дно інші серйозні тайм-трекери. Замість того щоб змушувати користувача вводити всю необхідну інформацію заздалегідь і заповнювати нескінченні форми, сервіс запитує відсутні у міру потреби.

В сумі всі ці якості пояснюють популярність Freckle. Такою кількістю іменитих користувачів не може похвалитися жоден з його конкурентів.

Головний недолік Freckle — ціна. У сервісу немає безкоштовної версії — навіть настільки урізаною, як у Harvest. Протягом двох тижнів їм можна користуватися безкоштовно, а потім доведеться платити не менше 19 доларів на місяць (це індівідуальний тарифний план). Дорожчі тарифні плани (49 і 199 доларів на місяць) відрізняються тільки збільшеною кількістю користувачів і підвищеної уважністю техпідтримки.

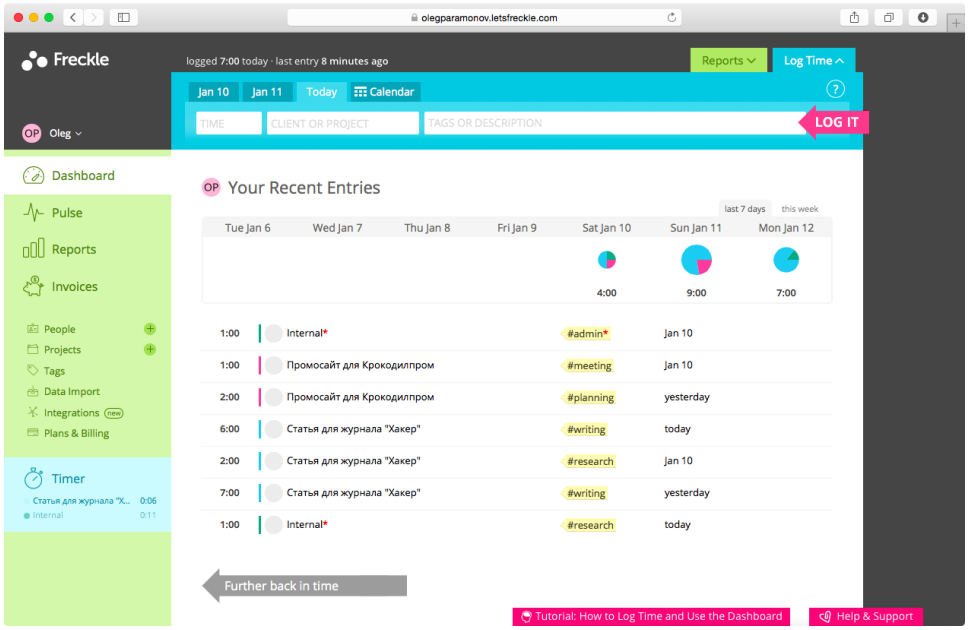

Рисунок 1.8 – Приклад додатку Freckle

1.6 Постановка задачі дипломного проектування

Необхідно створити систему керування IT проектом фірми, працюючої за офшорною схемою. Форма керування ієрархічна: керівник проекту, лідер групи, розробники. Кількість працівників до 20 осіб.

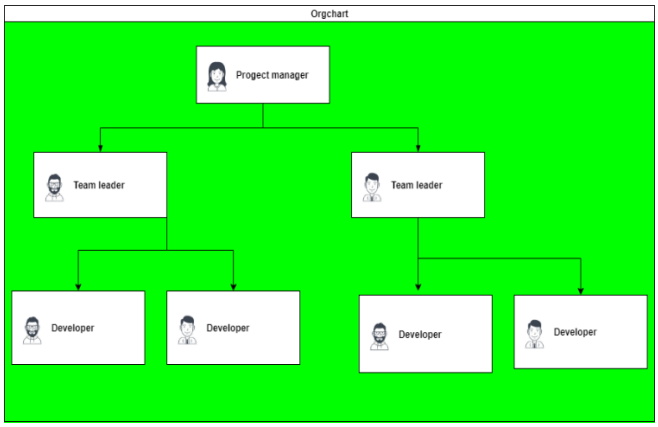

Рисунок 1.9 – Форма управлыння

Функціонал системи має містити стандартний набір функцій task manager для здійснення взаємозв'язків між різними проектними підгрупами та специфічний функціонал, що відповідає потребам замовника:

-лист «TO DO», на якому міститься перелік активних проектів/завдань з можливістю їх обговорення в окремій гілці;

-стан виконання кожного активного проекту та його загальний час виконання

- Система визначення особових рейтингів виконавців
- Система визначення загальних рейтингів підрозділів
- сторінка інфорозклади, що містить особові рейтинги виконавців, загальні рейтинги підрозділів,;
- розрахунок зарплати в залежності від часу виконання проекту та особових рейтингів виконавців та рейтингів підрозділів;
- система ступенів доступу
- репозиторій активних та вже закінчених проектів організації з різними ступенями доступу для внутрішнього використання.

Проведено аналіз переваг і недоліків систем task manager, що є аналогами системи, яка розробляється в дипломній роботі. Фірма -замовник має

більше 20 осіб працівників. Розглянуті аналоги не підходили по функціоналу та вартості програмного продукту. Тому було висунуто такі вимоги до системи task manager та має стандартний набір функцій такого типу систем для здійснення взаємозв'язків між різними проектними підгрупами:

- містити перелік активних проектів/завдань з можливістю їх обговорення в окремій гілці;
- інфорозклада, що містить особові рейтинги виконавців, загальні рейтинги підрозділів, стан виконання кожного активного проекту та його загальний час виконання;
- розрахунок зарплати в залежності від часу виконання проекту та особових рейтингів виконавців та рейтингів підрозділів;
- репозиторій активних та вже закінчених проектів організації з різними ступенями доступу для внутрішнього використання;
- складання звітів;
- архівація й захист даних.

1.7 Аналіз використання робочого часу

1. Коефіцієнт екстенсивного використання робочого часу

$$К_э = (Ф - Р)/Ф = 1 - Р/Ф,$$

де Ф - фонд робочого часу, хв; Р - регламентовані і нерегламентірованние перерви в роботі, хв.

2. Коефіцієнт втрат робочого часу, які залежать від працівників.

$$К_п = П_з/Ф,$$

де ПЗ - втрати робочого часу, що залежать від працівника, хв; Ф - фонд робочого часу, хв.

3. Коефіцієнт втрат робочого часу, які обумовлені організаційно-технічними причинами і не залежать від працівника де За - втрати з організаційно-технічних причин, які не залежать від працівників, хв; Ф - фонд робочого часу, хв.

$$К_п = П_о/Ф,$$

4. Коефіцієнт ціни робочого часу на відпочинок і особисті потреби працівників

$$К_{ол} = ОЛ/Ф,$$

де ОЛ - цінии часу працівника на особисті потреби (обідня перерва, виробнича гімнастика, гігієна і т.п.), хв; де Ф - фонд робочого часу, хв.

Аналогічно можна проаналізувати цінии часу працівника на виконання властивих йому функцій (КСВ), невластивих йому функцій (Київ), на виконання творчих робіт (Кт), організаційно-адміністративної роботи (Коа) і т.п.

Одне з найважливіших місць в системі успішної організації займає особистий і командний працю. У міру зростання конкуренції на ринку і постійно мінливих умов, навички тайм-менеджменту стає не просто бажаними - без них вже немислимий сучасний менеджер. Але при навчанні тайм-менеджменту нерідко виникають труднощі, пов'язані саме з практичною частиною ТМ. Для визначення якості ТМ-менеджменту в компанії був складений ТМ-профіль.

ТМ-профіль - статистика, що відображає ситуацію в компанії або підрозділі з усіх аспектів тайм-менеджменту.

Профіль будується на основі даних анкетування команди менеджерів за трьома основними напрямками впровадження тайм-менеджменту в компанії:

• особистий тайм-менеджмент-ступінь володіння навичками тайм менеджменту в середньому по даній групі менеджерів;

• командний тайм-менеджмент - якість «ТМ-взаємодії» по горизонталі всередині команди;

• корпоративний тайм-менеджмент - якість «ТМ-взаємодії» менеджера зі своїми підлеглими.

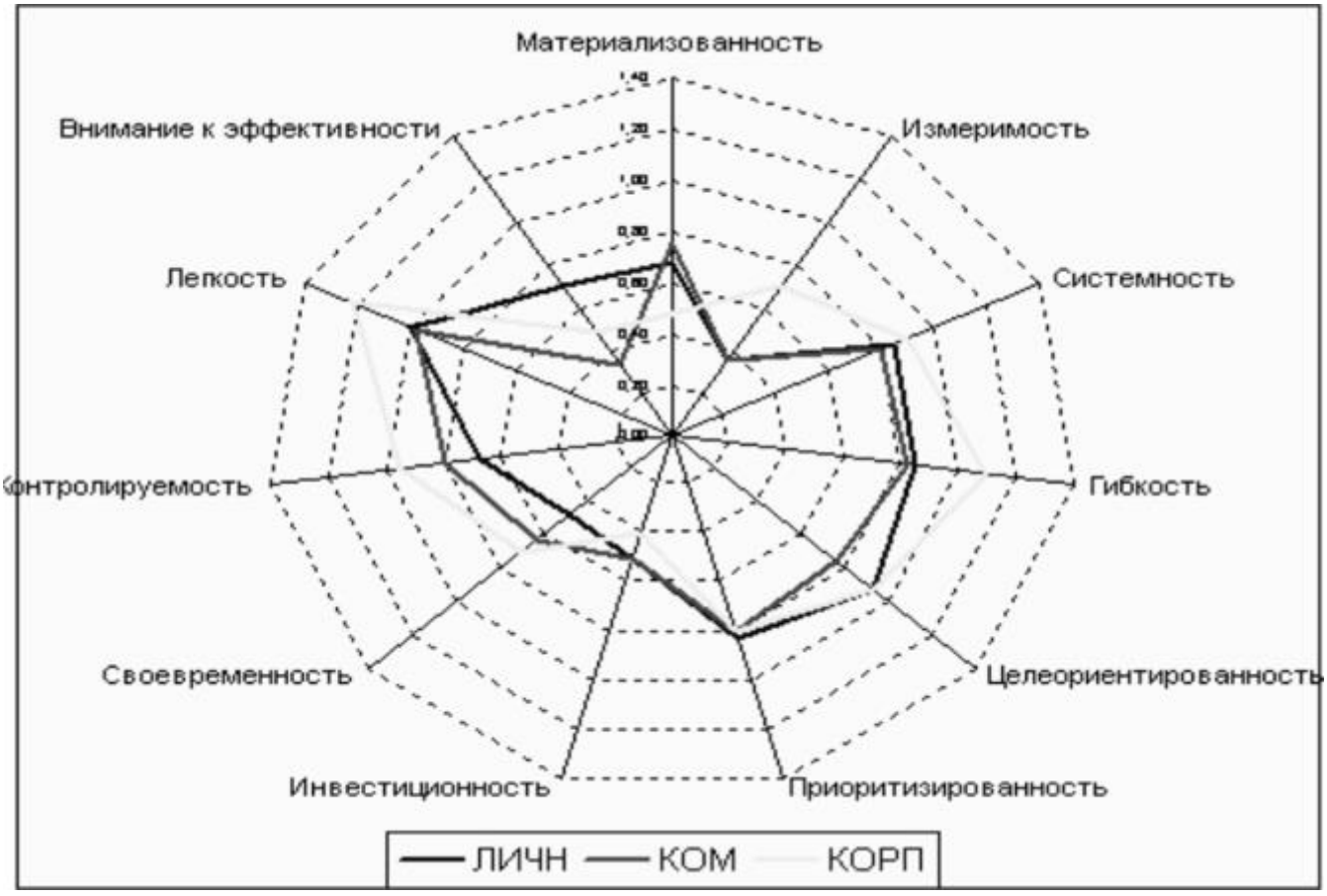

Рисунок 1.10 – ТМ профіль команди

В результаті аналізу представленого вище ТМ-профілю можна виявити головні проблеми системи та визначити шляхи їх вирішення. Наприклад,

нестача часу на перспективні завдання (низькі показники за критерієм інвестиційної) часто буває очевидним чином пов'язана з погано налагодженим оглядом і структуруванням інформації (низькі показники по матеріалізоване ™) і перевантаженістю керівника численними незначними запитами колег і підлеглих (низькі показники по «увазі до часу »в команді і / або підрозділі). Далі проводиться детальний аналіз за окремими критеріями, що враховує статистику розподілу відповідей всередині групи. Наприклад, на рисунку 1.11 відображена статистика розподілу відповідей за критерієм «Інвестиційні». По горизонталі - бал оцінки, по вертикалі - кількість респондентів, які обрали цю оцінку.

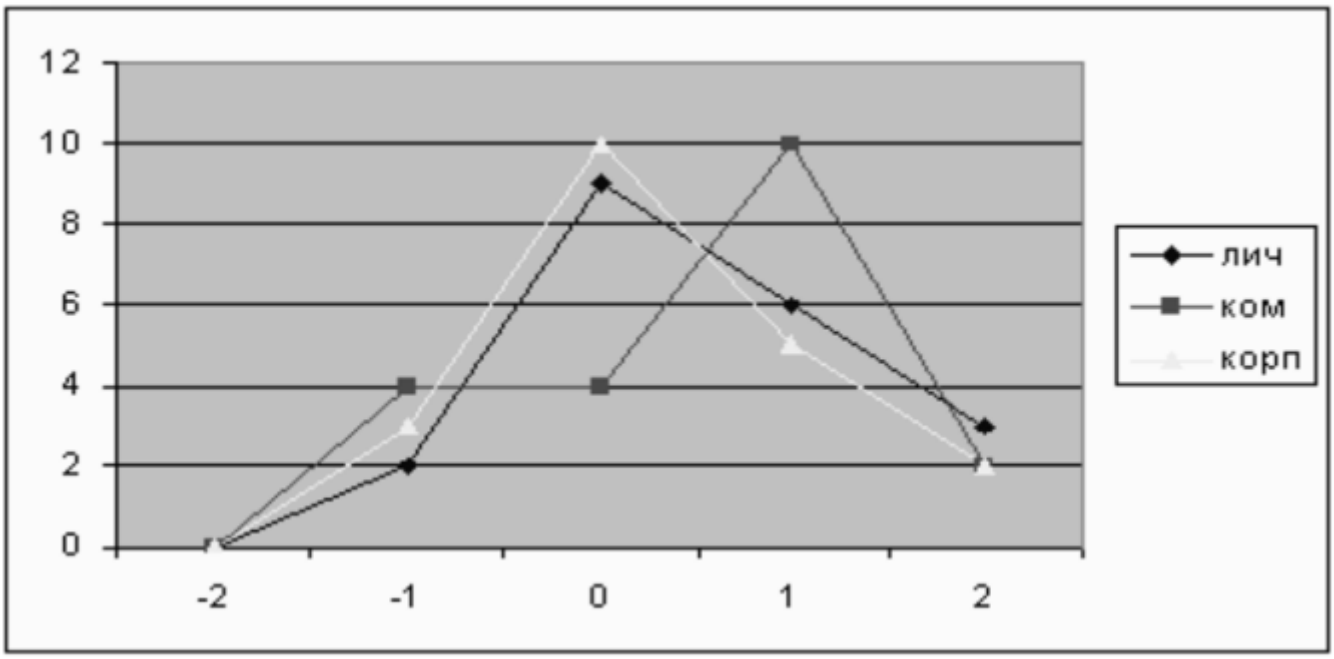

Рисунок 1.11 – Розподіл відповідей респондентів всередині групи

Особистий ТМ, оцінка 0: Час на «інвестиції в себе» в принципі виділяється, але далеко не в такій кількості, як хотілося би.

Командний ТМ, оцінка 1: У нашій команді свідомо виділяються час, енергія і ресурси на проекти «інвестиційного» характеру. У нас немає проблеми «колись вчитися», «немає часу впроваджувати нові технології». Але при цьому 20-30% часу все одно займає діяльність, що дає тільки короткострокові результати.

Корпоративний ТМ, оцінка 0: У моєму підрозділі і моїми підлеглими в принципі приділяється увага робіт перспективного характеру, але в центрі уваги все-таки короткострокові результати, їх легко виміряти і по ним оцінюють нашу діяльність.

Таким чином, виходячи з наведених даних, можна сказати, що в команді топ-менеджерів час на перспективні справи виділяється в достатній кількості, а в особистій праці співробітників – немає.

2 ВИБІР ЗАСОБІВ ПРОГРАМУВАННЯ ДЛЯ РОЗРОБКИ СИСТЕМИ

2.1 Серверна мова програмування PHP

Щоб розпочати роботу з дизайном, потрібно детально проаналізувати мову тексту - PHP. PHP (Private Home Pages) був розроблений Расмусом Лердорфом у 1994 році. Однак у 1997 році перекладача переписав інший програміст. Була дуже популярна мова PHP3 та більше послуг. Крім того, абревіатура PHP офіційно позначила його як "До гіпертекстового PHP". В даний час він використовує мову PHP4, розроблену Zend Technologies.

Хоча стратегії з відкритим кодом та розповсюдження з відкритим кодом, безсумнівно, мали позитивний вплив на багато проектів, особливо на Linux, успіх проекту Apache значно зміцнив його позиції як адвоката з відкритим кодом. Це також стосується історії створення PHP. Це тому, що підтримка користувачів у всьому світі була дуже важливим фактором у розвитку PHP-проектів.

PHP - це текстова мова на стороні сервера (на стороні сервера), яка вводиться безпосередньо в HTML-код. За допомогою PHP ви можете не тільки створювати динамічні сторінки, але й створювати інтерфейси баз даних. Синтаксис цієї мови простий, а вихідний код невеликий.

Використання PHP важливо при написанні часто оновлюваних або складних програм. Швидкість не є важливим критерієм. Скрипти PHP працюють дуже швидко, але не так швидко, як раніше розроблене програмне забезпечення. Якщо вам потрібно створити високопродуктивну програму, яка обробляє кілька додатків в секунду, вам слід скористатися компільованою мовою. Історично склалося так, що PHP був популярний серед розробників веб-систем завдяки своїй простоті та простоті.

У PHP існує безліч способів роботи з базами даних. Бібліотека, створена для роботи з MySQL, PostgreSQL, mSQL, Oracle, DBM, Hyperware, Informix,

InterBase та Sybase. Ви можете підключитися до будь-якої бази даних, де знаходиться драйвер, через відкриту базу даних ODBC.

Ефективність є дуже важливим фактором не тільки в Інтернеті, але і в програмуванні в багатокористувацькому середовищі. Головна перевага PHP полягає в тому, що ця мова призначена для перекладачів. Ви можете швидше обробляти документи. Деякі оцінки свідчать про те, що більшість документів PHP швидші за аналогічні програми, написані на Perl. Однак зібрані файли працюють дуже швидко. Кілька разів, а в деяких випадках і сотні. Однак продуктивність PHP досить хороша, щоб створити хороший веб-додаток. [16-17]

2.2 Мова та стандарти XML

Міжнародна організація W3C www.w3.org затвердила тест XML 1.0 на початку лютого 1998 року. XML є під компонентом загальної мови розмітки SGML (ISO 8879), розробленої командою розробників IBM у 1960-х роках. XML - описує об'єкти даних. Він називається документом XML і частково описує поведінку обробленої комп'ютерної програми. Він служить засобом пояснення граматики іншими мовами та перевірки точності документа. Отже, сам XML не має мітки для макета, він лише визначає створений макет.

Тому розробники мають власну можливість визначати власні команди, які можуть визначати дані в документі. Наприклад, наступну структуру можна використовувати для опису групи студентів:

Лістинг 2.1:

```
<groop>
<student>Іванов С.П.</student>
<student>Костюк А.П.</student>
<student>Храмцов Р.А.</student>
```

</groop>

Документи XML можна використовувати як унікальний спосіб зберігання даних. Він включає всі інструменти для аналізу інформації та представлення її клієнтам. Однією з очевидних переваг XML є можливість використання XML як глобальної мови запитів для інформаційних ресурсів.

На сьогоднішній день існує багато мов, побудованих на XML (рис. 2.1).

XML може використовуватися в будь-якій програмі, яка вимагає структурованої інформації, від складних геоінформаційних систем до загального використання "єдиного комп'ютера", який використовує мови для подання службової інформації.

Основними програмами XML сьогодні є:

Документи XML служать глобальним форматом для обміну інформацією між окремими пристроями в складних інформаційних системах.

XML є основним стандартом для RDF, нової мови визначення мови. RDF спрощує багато проблем в Інтернеті, знаходячи потрібну інформацію, керуючи вмістом на веб-ресурсах, створюючи електронні бібліотеки тощо.

XML може використовуватися для інтерпретації всіх видів даних і використовується для представлення конкретної інформації, такої як хімічні речовини, математика, формули тіла, інструкції та примітки. Це означає, що XML може бути потужним доповненням до HTML, поширюючи "рідкісну" інформацію в Інтернеті..

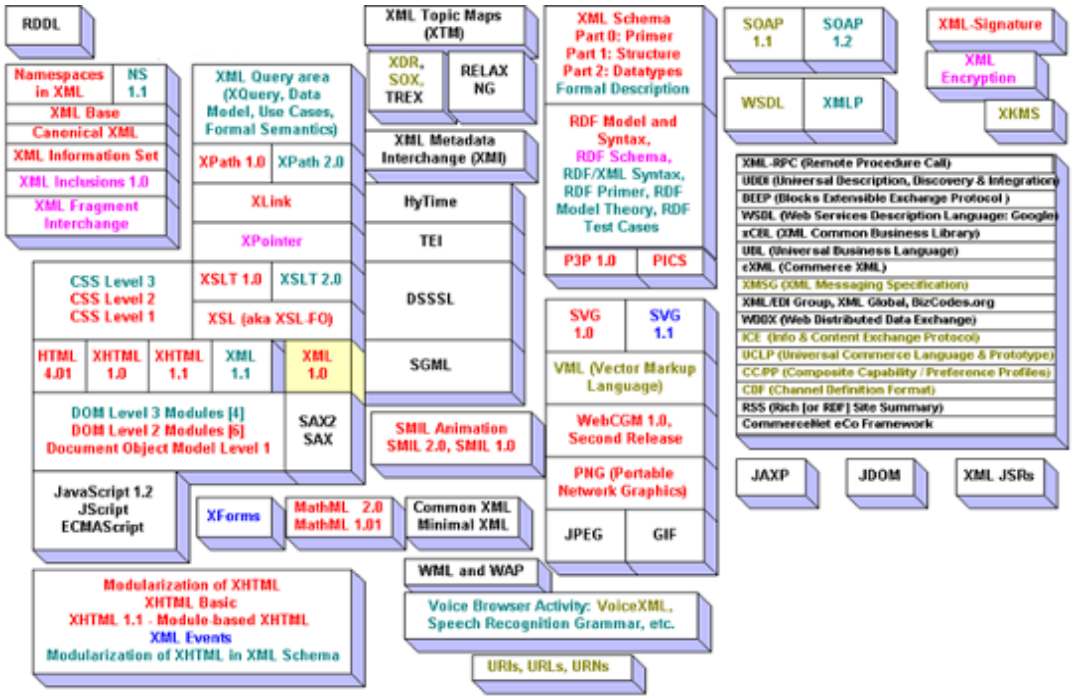

Рисунок 2.1 – XML-сімейство специфікацій

Документи XML можна використовувати як проміжний формат даних для 3-рівневих систем. Приклади взаємодії між серверами додатків та базами даних зазвичай залежать від конкретної бази даних та варіанту SQL, що використовується для доступу до даних. Якщо результати запиту відображаються у глобальному текстовому форматі, тоді посилання з такої СУБД на додаток є «відкритим». Крім того, сьогодні W3C рекомендує нову класифікацію мови запитів для баз даних XQL. У майбутньому це може бути альтернативою SQL.

Інформацію, що міститься в документі XML, можна змінювати, передавати на клієнтську машину та оновлювати з різних місць. Класифікації XLink та Xpointer дозволяють переглядати окремі елементи документа на основі вкладеності та значень атрибутів.

Ви можете використовувати таблиці стилів (XSL) для перевірки безкоштовних XML-документів. Розширити можливості надання інформації на популярні КПК та мобільні телефони.XML може використовуватися в звичайних додатках для збереження й обробки структурованих даних у єдиному форматі.

Документи XML - це стандартні текстові файли, які використовують спеціальні символи для створення об’єктів даних, макетів та гнізд, що визначають структуру та зміст документа.

Тіло документа XML складається з макета, що міститься в документі, та прямих об'єктів, або даних (вмісту). Теги XML використовуються для визначення елементів документа, їх властивостей та складу інших мов.

Загалом документи XML повинні відповідати таким вимогам:

Документи XML завжди повинні починатися з "<? Xml?>". Це також дозволяє вказати номер версії мови, номер сторінки та інші параметри (інформація про тип документа), які потрібні аналітикам при аналізі документа.

Вам потрібен один елемент, який називається елементом документа. Цей елемент не відображає частину вмісту інших предметів.

Кожна пластина розблокування, що визначає певну область даних у документі, вимагає власного аналога. Тобто, на відміну від HTML, ви не можете залишити фіксатор.

XML чутливий до регістру.

Усі атрибути, що використовуються для визначення тегів, повинні бути в лапках.

Оскільки вкладання етикеток XML є дуже контрольованим, важливо дотримуватися порядку відкривання та закриття етикеток.

Вся інформація між початковим і кінцевим символами розглядається як дані в XML, тому розглядаються всі структурні символи (пробіли, плавні розриви та вкладки не враховуються, як у HTML).

Якщо документ XML не порушує вищезазначені правила, тоді документ XML має офіційну назву, і всі аналітики, створені для аналізу документа XML, можуть працювати належним чином. Зміст документа XML - це набір елементів, розділи CDATA, навчальні посібники, коментарі, спеціальні символи та текстові дані.

Характеристика - це структурна одиниця XML-документа, між якими є інші мітки та інформація. Колекція всіх елементів у документі визначає його

структуру та визначає взаємозв'язки для всіх стовпців. Документи XML зазвичай визначають принаймні один об’єкт, який називається коренем, і програма Explorer починає перегляд документа. Якщо елемент не містить вмісту, його називають порожнім. Ви повинні з’єднати позначки початку та кінця порожнього предмета по одному та розмістити шов перед закриваючим скобкою.

Коментарями є будь-як область даних між послідовностями символів «<! – і –>».

Атрибут - це конкретне "ім'я = значення", яке визначається, коли елемент визначається в оригінальному трактаті та коли ви визначаєте властивості цього елемента.

Спеціальні символи (квадратні дужки, пробіли тощо) включаються в документ за допомогою спеціальних символів або числових позначок. CDATA - Використовується під час аналізу для визначення місця розташування документа, на якому аналітик зосередиться у простому тексті, але, на відміну від думки, його можна використовувати програмно. – «<! [CDATA] і]]>».[18-19]

2.3 База даних PostgreSQL

Сьогодні системи управління базами даних (СУБД) є важливими інструментами у багатьох сферах, включаючи бізнес, дослідження, освіту та розробку онлайн-пошукової системи. Однак, незважаючи на важливість належних баз даних для підтримки та доступу до інформаційних ресурсів, багато організацій не використовують бази даних у своєму бізнесі. Історично склалося, що СУБД стали дуже дорогими, і постачальники встановили дуже високі ціни на все програмне забезпечення та послуги технічної підтримки.

Крім того, системи СУБД повинні відповідати вимогам до продуктивності апаратної платформи, додатково збільшуючи вартість таких рішень.

За останні кілька десятиліть все різко змінилося з точки зору програмного та апаратного забезпечення. Персональні комп'ютери дешевші та одночасно потужніші. У той же час спостерігалась тенденція до створення високоякісних систем, які можна було придбати за ціною дешевих лазерних дисків або безкоштовно отримати через Інтернет. До таких баз даних на основі BSD UNIX належать FreeBSD, NetBSD, OpenBSD та різні версії Linux (RedHad, Caldera, LsnuxPPC).

Створення операційної системи, яка використовує багато покращених функцій комп'ютера, в першу чергу завдяки розробці та інструментам розробки, таким як безкоштовний компілятор GNU.

Одним із представників таких баз даних є СУБД, пов'язана з клієнтом-сервером PostgreSQL.. [20-21]

2.4. Метод

Шлях PostgreSQL є блокувачем коду, який працює на сервері, а не на клієнті бази даних. Це може бути написано на чистому SQL, але реалізація додаткової логіки, наприклад, зміна рядків і циклів, виходить за рамки власне SQL і вимагає використання розширень мови. Метод може бути написаний однією з наступних мов:

- подібна PL вбудована мова програмування PL / pgSQL
- мова SQL, що використовується базою даних PL / Oracle.
- мови тексту - PL / Lua, PL / LOLCODE, PL / Perl, plPHP, PL / Python, PL / Ruby, PL / sh, PL / Tcl та PL / програми.
- спільні мови-C, C ++, Java (через модуль PL / Java);
- мовна статистика Мова R (через модуль PL / R).

У PostgreSQL ви можете використовувати функції, що генерують колекцію записів, які ви можете використовувати так само, як і стандартні результати запитів.

Цей метод можна виконати за допомогою авторських прав та поточних прав користувача. Іноді збережений процес визначає метод, але між цими поняттями існують відмінності..

2.5 Тригери

Тригери визначаються як метод, встановлений за допомогою маніпуляцій DML. Наприклад, операція INSERT може посилатися на доданий запис або активувати тригер, який відповідає певним умовам. Ви можете використовувати різні мови програмування під час написання стимулюючих функцій.

Тригери пов'язані з таблицею. Деякі подразники реалізуються в алфавітному порядку.

2.6 Правила та представлення

Правила процедури полягають у створенні стандартних обробників для вибірки, а також операцій DML. Основна відмінність у процедурі стимулювання полягає в тому, що правила активуються в процесі аналізу вимог та вибору оптимального плану реалізації та прогресу в реалізації. Правила дозволяють змінювати поведінку вашої системи під час виконання операцій SQL над таблицею.

Хорошим прикладом є перспективна реалізація: Створення коментаря створює правило, яке вказує на те, що система повинна виконувати операції з моделлю в базовій таблиці / таблиці замість моделі операції попередньої надсилання. Приклади умов на основі визначення. Щоб створити коментарі, що підтримують операції оновлення, користувачі повинні визначити правила вставки, модифікації та видалення стовпців.

2.7 Індекси

PostgreSQL підтримує такі типи індексів: B-дерева, хеші, R-дерева, GiST, GIN (4). Ви можете створити новий індекс, якщо хочете, але це непростий процес. Індекси PostgreSQL мають такі характеристики:

- ви можете індексувати різні стовпці в таблиці та різні типи стовпців даних.
- індекс працює. Тобто він будується на наборі значень, а не на певному значенні стовпця / стовпця Шлях із набору значень.

Індекс може бути частковим. Тобто деякі прогнози будують лише частину таблиці. У деяких випадках це може допомогти вам створити більш компактний індекс або покращити продуктивність, використовуючи різні типи індексів для різних частин (наприклад, частоти оновлення) таблиць.

2.8 Механізм «Multiversion Concurrency Control»

PostgreSQL використовує Мультиверсійну систему фінансового управління (MVCC) для підтримки одночасної модифікації бази даних багатьма

користувачами. Як результат, вимоги до ACID виконуються, і немає необхідності запобігати читання.

2.9 Типи даних

PostgreSQL підтримує ряд вбудованих типів даних.

--Тип цифр.

- мета;

-І дана дія;

--З плаваючою дією.

Тип валюти (залежно від конкретного формату виводу, інакше він відповідає двом знакам після коми)

- модель типу будь-якої довжини.
- бінарний тип (включаючи BLOB);
- дата / тип часу "(підтримує останні зміни в різних форматах, точності, вихідних форматах та часових поясах);
- булевий тип.
- передача;
- геометричне походження.
- тип мережі.
- IP та IPv6 адреси.
- формат CIDR.
- мак-адреса;
- UUID ID.
- статистика XML;
- колонка;
- OID тип.

- псевдотип.

Крім того, користувачі можуть створювати власні нові важливі для них типи та планувати процес складання списку за допомогою GiST.

PostgreSQL може масштабуватися користувачами під власні потреби практично в кожній ситуації. Ви можете додавати типи даних.

- тип змін.
- тип даних.
- домен (раніше поширений тип та обмеження);
- метод (плюс загальний);
- індекс;
- оператори (і перевизначення існуючих);
- процедурна мова.

Таблиця може успадковувати властивості та набори з інших таблиць (батьків). У цьому випадку дані, додані до сформованої таблиці, автоматично братимуть участь у питанні в батьківській таблиці (якщо не вказано інше).

Ця функція ще не завершена. Однак цього достатньо для практичного використання.

2.10 Інші можливості

Існує ще декілька можливістей:

- дотримання кислотного принципу.
- відповідає стандартам ANSI SQL-92 та SQL-99.
- підтримка програм від OUTER JOIN, UNITY, UNITY OSEMTT, RESPONSIBLE та нижчих рівнів.
- Стовпець.
- монітор цілісності.
- імітація.

- загальні описи таблиць та повторювані запитання.
- метод аналізу.
- підтримка Unicode (UTF-8).
- підтримка поширених виразів у стилі Perl.
- вбудована підтримка SSL та Kerberos.
- протокол спільного блокування.
- завантажувані розширення, що підтримують SHA1, MD5, XML та інші служби (відкриті API).
- інструменти для створення та імпорту відповідного коду SQL та інших систем.

2.11 Надійність

За результатами автоматизованого розслідування програмних помилок у вихідному коді PostgreSQL було виявлено 20 проблемних зон для 775 000 рядків вихідного коду (в середньому одна помилка на 39 000 числових рядків). Для порівняння: випуск MySQL --97, одна помилка на 4000 рядків. Помилка FreeBSD - одна помилка в 306, 4000 рядків. Linux (лише базовий) - 950 неполадок, 1 помилка на кожні 10000 рядків.

3 РІВНІ ТА СПОСОБИ ВЗАЄМОДІЇ ЗАСОБІВ РОЗРОБКИ

3.1 Система керування контентом

Система управління вмістом (CMS) - це інформаційна система або комп’ютерна програма, яка використовується для забезпечення та сприяння розвитку спільного створення, редагування та управління вмістом. Основною метою таких систем є можливість поєднувати всі різні джерела знань та інформацію, що є в організації та за її межами, залежно від їх ролі та відповідальності, а також для персоналу, робочих груп, а також забезпечувати взаємодію між проектами, заснованими на знаннях. , Інформація та дані можуть бути легко доступні, отримані та використані користувачами звичайним способом.

Системи управління вмістом можуть виявляти різноманітні дані, включаючи документи, фільми, фотографії, телефонні номери та наукові дані. Такі системи використовуються для зберігання, управління, перегляду та публікації документів. Переваги CMS:

- не потрібно добре володіти знаннями в області html і css, достатній базовий рівень;
- високий рівень безпеки при регулярному оновленні CMS;
- безліч доповнень і розширень, можливо створити сайт будь-якої складності, величезна кількість професійних шаблонів.
- Недоліки CMS:
- час завантаження сторінок сайту длительней, ніж у звичайних сайтів;
- складнощі у відновленні працездатності сайту в разі його падіння.[22]

3.2 Розроблення архітектури програмної системи

По-перше, я коротко представлю низьку архітектуру, яка добре розміщена на кістках уіі. Я не думаю, що нам потрібно говорити про спеціальну версію уіі. Тут найголовніше, як його загалом готують. Планова архітектура - це системна архітектура, в якій система складається з ряду впорядкованих систем, які називаються шарами.

1. У кожному стовпці нічого не відомо про властивості іншого (і навіть його присутності) наступних (вищих) шарів.

2. Кожен шар може спілкуватися з нижнім шаром раніше прийнятим способом, не знаючи нічого про його структуру чи внутрішню реалізацію. Тому в програмних системах низького рівня кожен персонаж може виконувати власний витяг даних. Співвідношення між шарами обмежується шляхом перенесення значень параметрів у кожному стовпці в сусідній нижній шар і надсилання результатів із нижнього шару у верхній. Вам заборонено використовувати глобальні дані на інших шарах. Якщо ви не розглядали основні рівні мереж та серверів, давайте розглянемо структуру самого програмного забезпечення. Програмне забезпечення може мати три шари. (рис. 3.1).

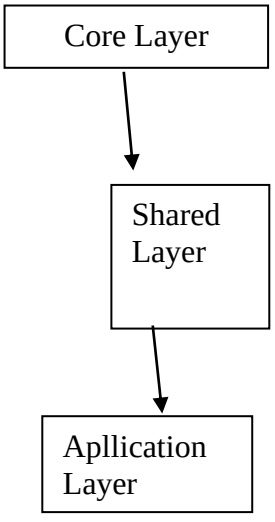

Рисунок 3.1 – Шари технічної бази

Переглянемо їх:

- Core Layer. Існує мало бібліотек і систем. Наприклад, yii.
- Shared Layer - рівень модулів, які виконують певні системні функції і можуть бути використані пізніше в інших проектах.
- Aplication Layer - Рівень програми та його конкретні модулі.

Важливим є те, що модулі можуть бути легко переміщені з програмного рівня на загальний рівень. Причина дуже проста: вам не потрібно переписувати число двічі, тому що вам потрібно думати про майбутнє, можливо, вийде подібний продукт, і вам знадобиться подібний модуль. І той факт, що ви можете легко перенести модулі у спільний масив, свідчить про відсутність залежностей, що завжди добре. Розглянемо файлову структуру стовпця програми на основі цієї класифікації.[23-24]

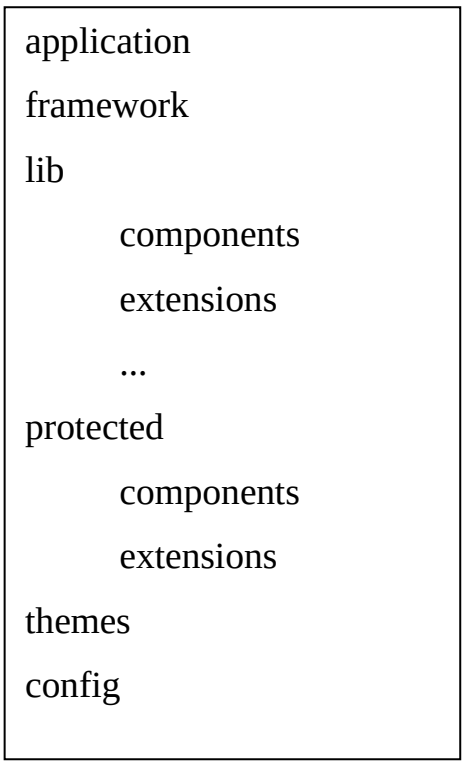

Рисунок 3.2 –Файлова організація застосунку

Типова програма містить формат, показаний на малюнку 3.2. index.php - це розділ для входу, де подається домен і створюється модель WebApplication (існує також ConsoleApplication). Структура файлів об’єднує файли, які отримують спільний архівний ресурс. Шлях до цього файлу також застарів. Усі дії можуть починатися лише з контролера 35 і включати опис фактичної дії

контролера. Адміністратор обробляє вхідні дані, маніпулює віджетами, моделями даних та налаштовує вихідні дані HTML та інші відповіді, json, xml та все інше, що потрібно користувачеві.

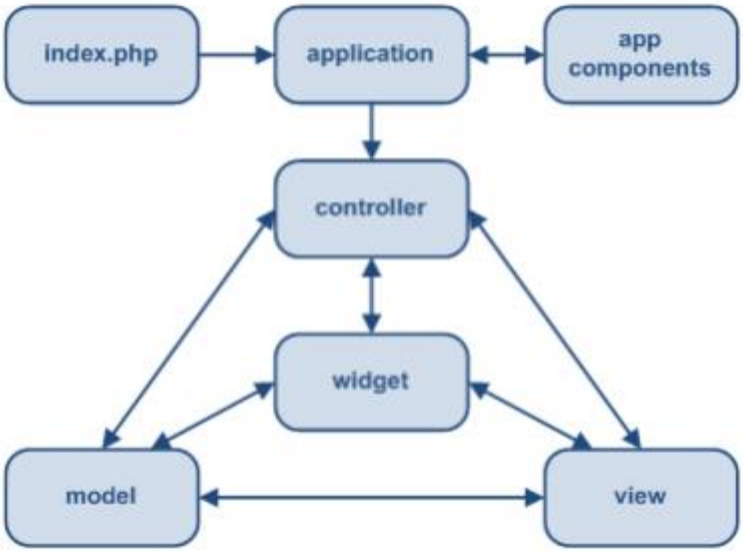

Рисунок 3.3 – Будова типового дії застосунку

Весела роль у прикладі. Це повністю взято з джерела даних. Ви можете запускати зразки, чистити, перевіряти та будь-яку СУБД.

В основі віджету - міні-клас із класами, які визначають його поведінку та шаблон експорту. Найповсякденнішими деталями віджетів є бар’єри в Інтернеті, такі як навігаційні панелі та фотографії готелів. Стандартна операційна система показана на малюнку 3.4. Наслідуючи цей приклад, ви швидко зможете побачити, що має система та які типи завдань вона може вирішити.

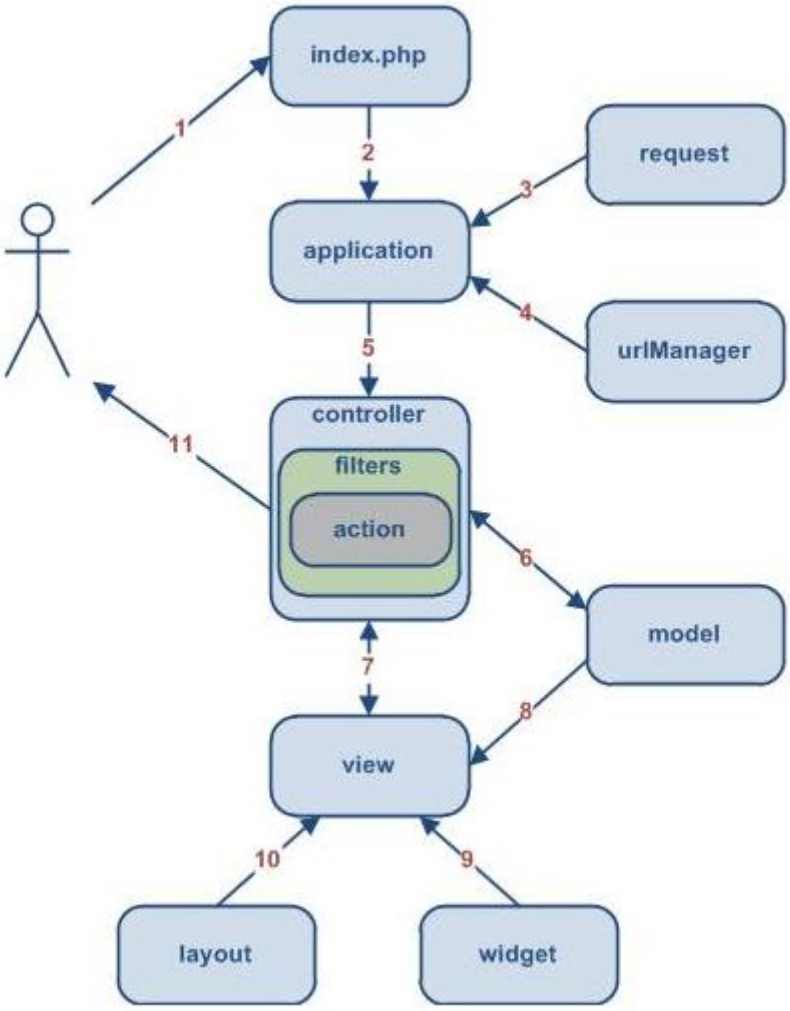

Рисунок 3.4 – Сценарій використання застосунку

1. Коли користувач відправляє запит на адресу http://www.example.com/index.php?r=post/show&id=1, веб-сервер отримує та виконує документ, встановлений у index.php.

2. Документ створює модель програми та запускає її.

3. Додаток отримує всі дані користувача та зберігає їх у розділі запиту.

4. Завдяки розділу urlManager програма розрізняє контролери та дії. Наприклад, Post посилається на клас PostController. Ось дії, реалізовані в цьому класі як публічний підхід.

5. Тепер, коли ви визначили клас, який відповідає вашому запиту, створіть для нього модель. Дія show посилається на метод actionShow цього класу.

6. Введіть ідентифікатор 1 у базі даних Приклад.

7. Використовуйте такі дії, щоб перетворити отримані дані моделі в шаблон.

8. Відобразіть виставкову функцію статті розміщеного прикладу.

9. У поданні є багато віджетів.

10. Результати шоу включені в макет - це один із шаблонів найвищого рівня.

11. Дія завершує доставку і повертає результат користувачеві

4 РОЗРОБКА ІНТЕРНЕТ-ДОДАТКУ З ВИКОРИСТАННЯМ PHP, POSTGRESQL

4.1 Розробка БД PostgreSQL

У розробленому мною програмному забезпеченні БД, подана через базу даних PostgreSQL, відіграє важливу роль. Ми вибрали цей тип баз даних, оскільки система PostgreSQL підтримує діяльність, необхідну для обробки фінансового балансу компанії та балансів працівників.

Діяльність - це група наступних операцій над базою даних, які є логічними одиницями для обробки даних. Такі заходи можуть виконуватися повністю та успішно, беручи до уваги цілісність даних та незалежність інших видів діяльності, які сумісні або ніколи не виконувались і не мають ефекту. Транзакції - це процес створення історії транзакцій і обробляються системою закупівель.

Паралельно розповсюджується низка (звичайних) видів діяльності. Розподілена діяльність усвідомлює використання декількох систем діяльності та вимагає більш складної логіки (наприклад, двофазні зобов’язання). Деякі системи також виконують самостійну діяльність або невелику діяльність, яка є частиною незалежності батьків.

Однією з найпоширеніших вимог до систем транзакцій та закупівель є концентрація кислоти (атомність, консистенція, розділеність, довговічність). В кінці 1970-х Джим Грей широко розробив вимоги до кислоти. Однак існують також спеціальні системи із слабкими купівельними характеристиками.

Атомарність запобігає частковій реєстрації діяльності в системі. Все це робиться за рахунок діяльності чи нічого. Ідея "компенсації" запроваджена, оскільки неможливо виконати цілий комплекс дій під час одночасної та атомної діяльності. Одна з найскладніших та найскладніших послуг квартету ACID. Відповідно до цих вимог, система повинна знаходитись у затвердженому стані

до транзакції та залишатися в узгодженому стані після завершення транзакції. Не плутайте вимоги до положення з вимогами цілісності. Остаточне правило є вузьким і найчастіше стосується реляційних баз даних. Існують вимоги щодо цілісності домену, цілісності домену, цілісності пам’яті та цілісності пристрою, а унікальність реалізації системи не порушує консоль.

Рівність - це широке поняття. Наприклад, банківській системі може знадобитися збалансувати суму, випущену одним рахунком, із сумою, внесеною на інший. Це законодавство про підприємництво і не може бути гарантоване лише аудитом доброчесності. Організатор програми повинен продовжувати, вводячи номер діяльності. Якщо ви не виконуєте операцію з позикою в кінці транзакції, ваша система залишається в поганому стані, а ваші активи стабільності порушені.

Нарешті, інший погляд полягає в тому, що під час транзакції не потрібна жодна позиція. У цьому прикладі скасування боргу та кредит - це, мабуть, дві різні транзакції, і ви побачите несумісні системні твердження між реалізаціями в цій транзакції. Однак майте на увазі, що ця відмінність не застосовується до інших видів діяльності, якщо вимоги до відокремлення виконуються. І атомізм гарантує, що діяльність повністю завершена або не виконується. Тому ця тимчасова двозначність прихована.

ізоляція. Якщо виконується певна діяльність, паралельні дії не повинні впливати на результати. Ця властивість не враховується нижче рівня повторної ізоляції Соми.

Зміни, внесені в успішно завершені дії, незалежно від проблем низького рівня (наприклад, відключення електроенергії або несправності обладнання), слід зберігати після перезавантаження системи. Іншими словами, користувач, який отримує підтвердження від системи про те, що транзакція завершена, може гарантувати, що внесені зміни незворотні. Таким чином були створені таблиці, структура яких представлена на рисунках (Рис. 4.1)

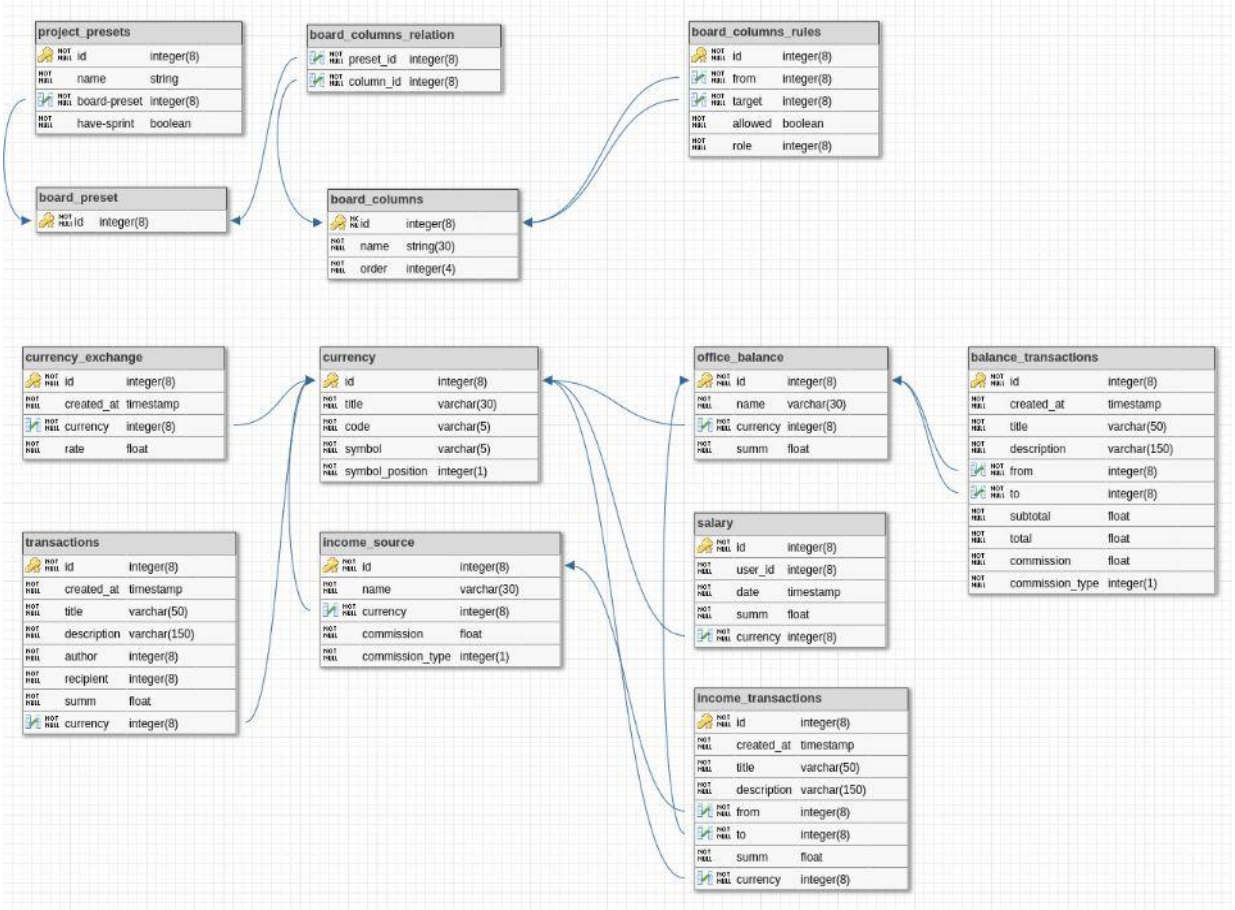

Рисунок 4.1 — Структура зв'язків таблиці відповідаючи за фінансову частину.

4.2 Розробка інтернет додатку

Умовно додаток можна поділити на дві основні частини:

– адміністративна частина;
– інтерфейс користувача.

4.2.1 Адміністративна частина

Адміністративна частина призначена для керування робочим процесом. У цій частині можна додавати, видаляти клієнтів, додавати та видаляти проекти, керувати нарахуванням заробітної плати робітникам. Для доступу до адміністративної частини потрібно бути зареєстрованим на сайті, додати до URL адреси додати /admin/ та ввести логін та пароль.

В адміністративній частині в інтерфейс були внесені наступні сторінки:

1. головна сторінка (Рис 4.2);

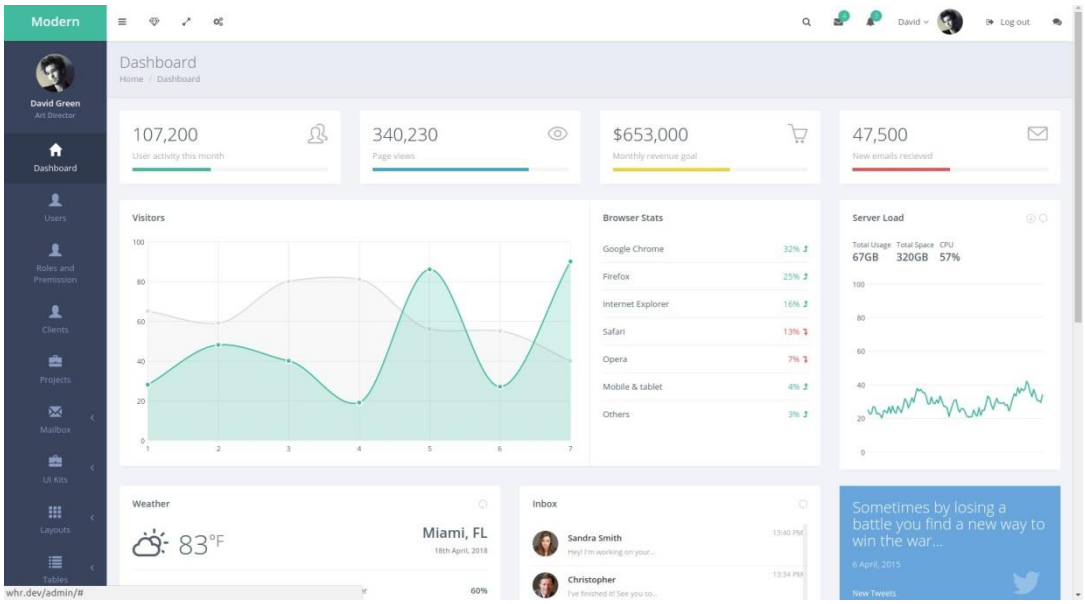

Рисунок 4.2 — Головна сторінка

2. сторінка зі списком прав для різних ролей (Рис. 4.3);

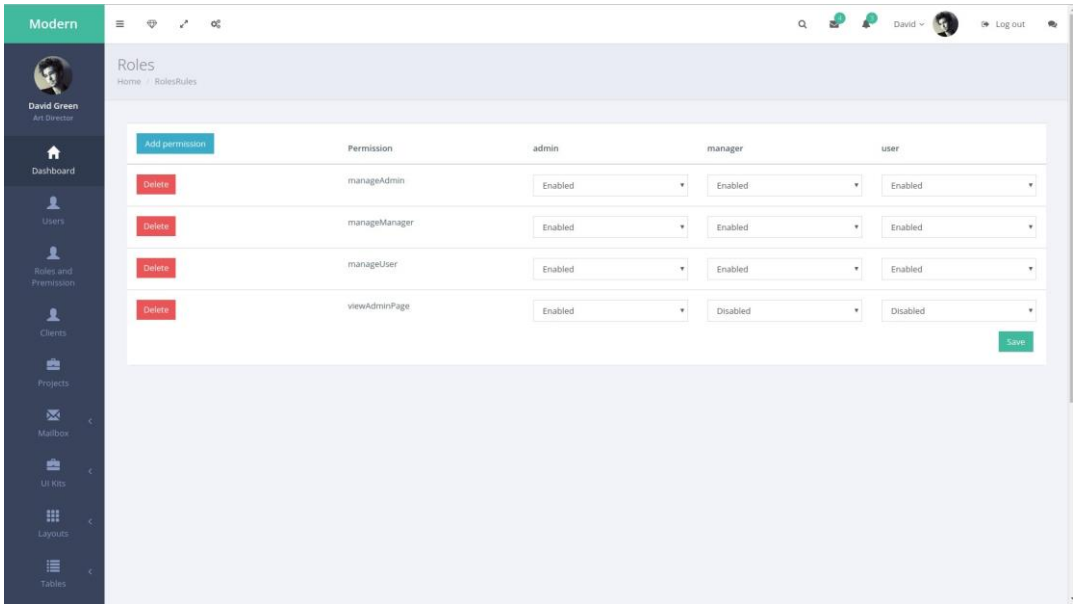

Рисунок 4.3 — Сторінка зі списком прав для різних ролей

3. сторінка зі списком користувачей (Рис. 4.4);

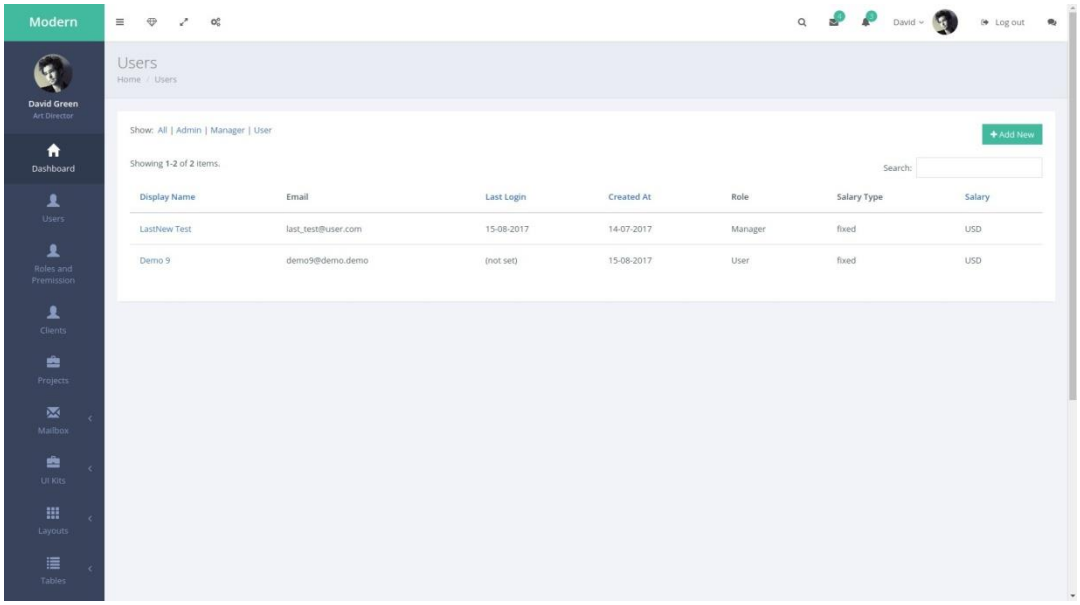

Рисунок 4.4 — Список користувачів

4. сторінка клієнтів (Рисунок 4.5);

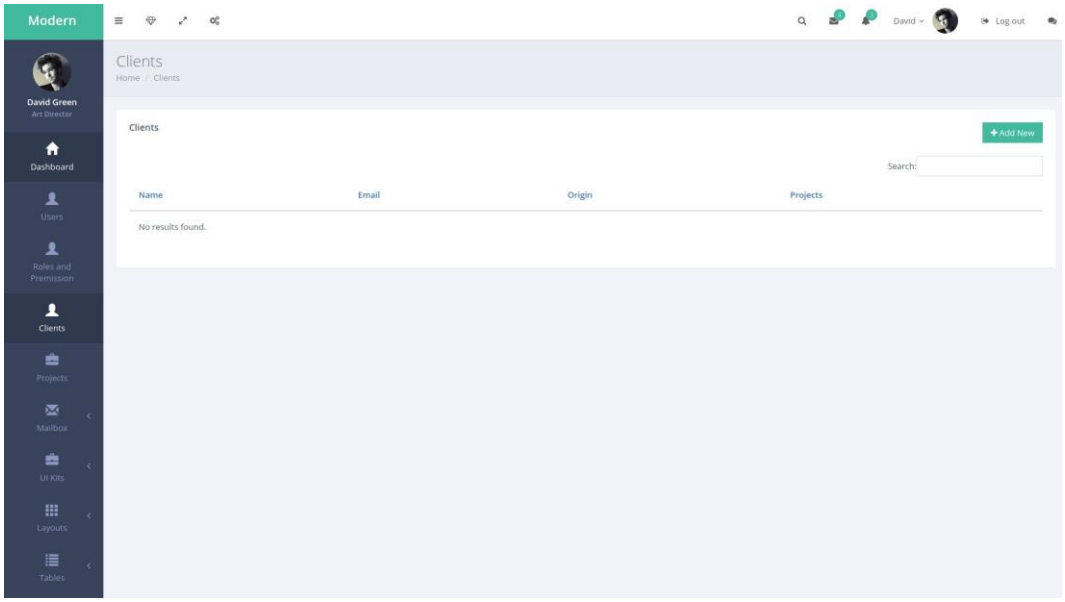

Рисунок 4.5 — Список клієнтів

5. сторінка проектів(Рисунок 4.6);

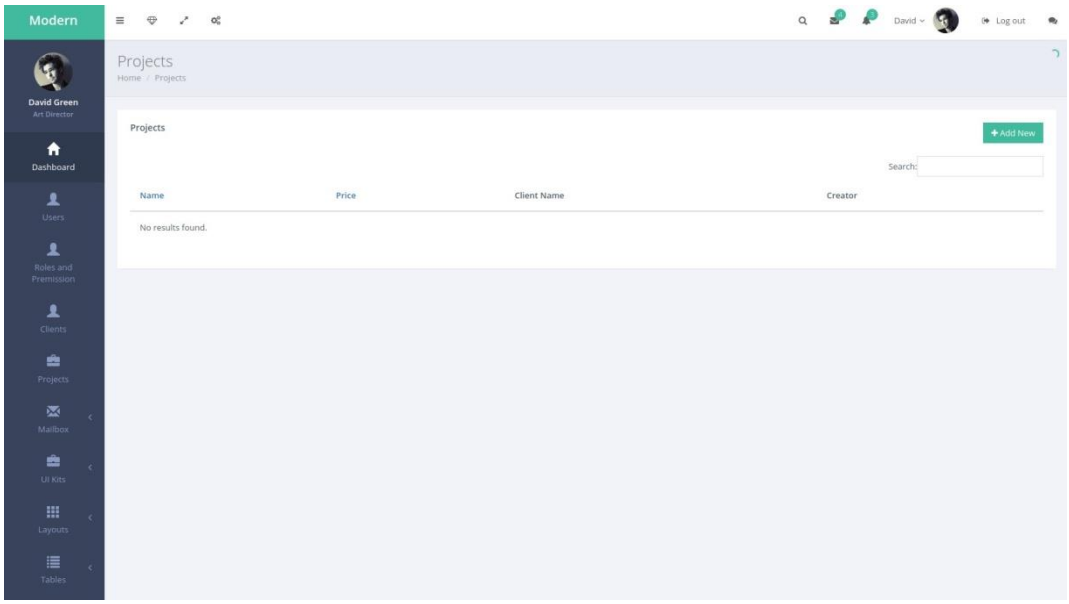

Рисунок 4.6 — Список проектів

6. календар(рис. 4.7);

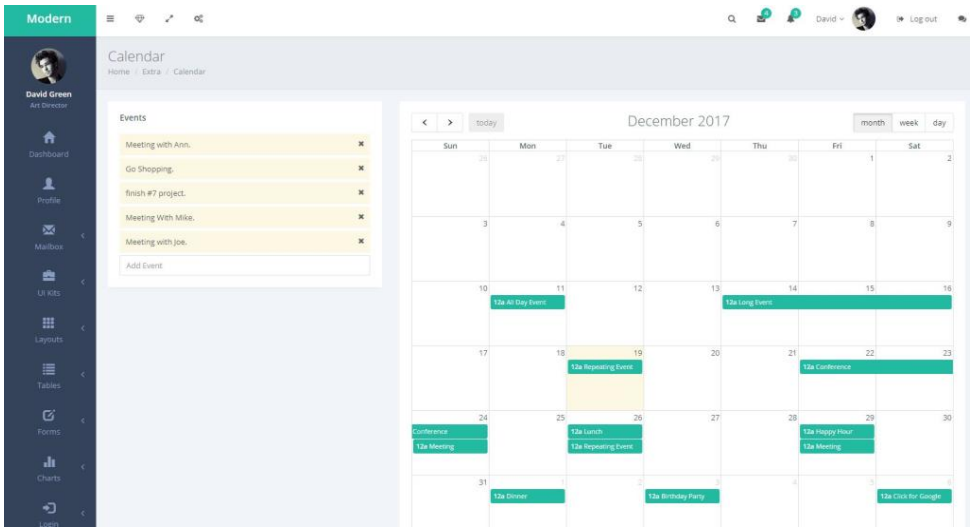

Рисунок 4.7 — Список проектів

4.2.2 Інтерфейс користувача

Ця частина сайту призначена для робітників. У цій частині можна отримати звіт з виконаної роботи за одним проектом, подивитись історію за весь час отримати інформацію за об'ємом напрацьованих годин та заробітної платні. Для доступу до цієї частині сайту потрібно бути зареєстрованим на сайті, перейти за головною URL адресом сайту та ввести логін та пароль. В інтерфейсі користувача були внесені наступні сторінки:\

1. головна сторінка(рис. 4.8);

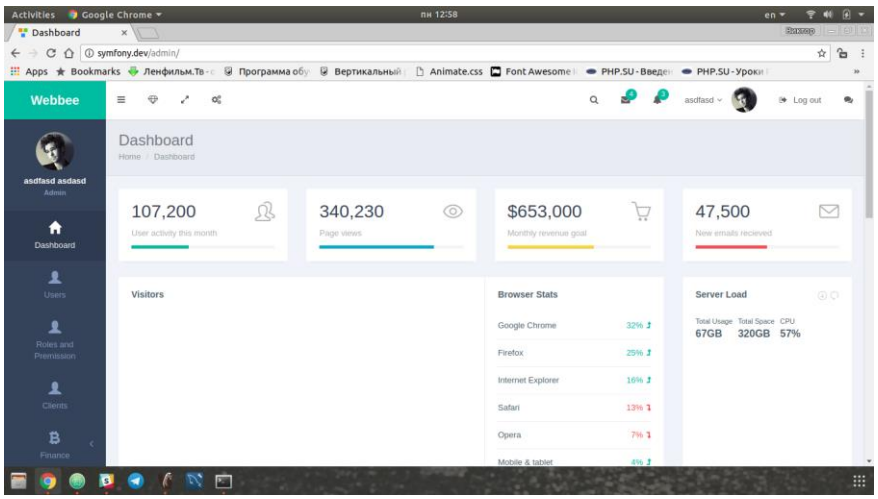

Рисунок 4.8 — Список проектів

2. сторінка To-Do List;

3. сторінка зі звітом по напрацьованим годинам, нарахованій з.п., та списком проектів в яких робітник приймав участь(рис. 4.9);

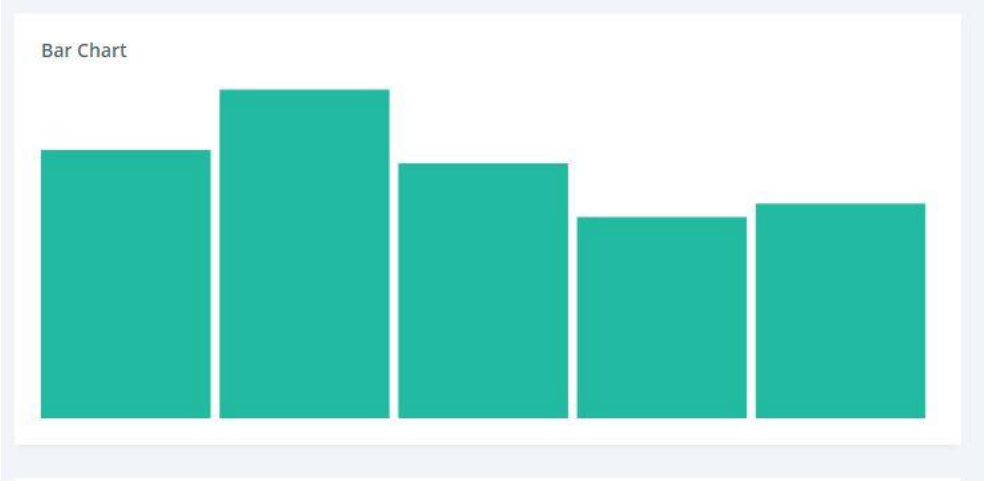

Рисунок 4.9 — Список проектів

4. сторінка з календарем для візуалізації термінів здачі проектів та подій, які можна додати в адмін. панелі(рис. 4.10);

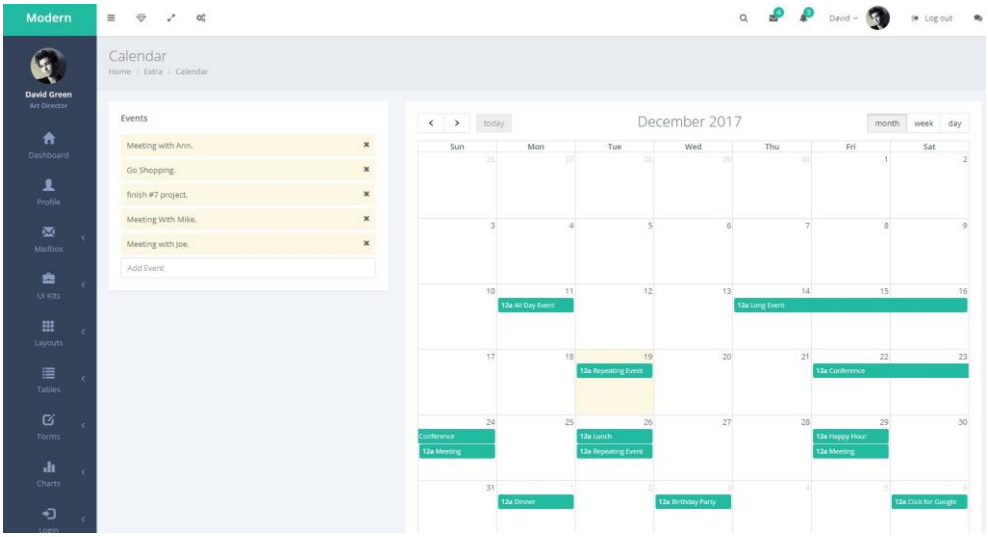

Рисунок 4.10 — Список проектів

5. сторінка ціни часу (рис4.11)

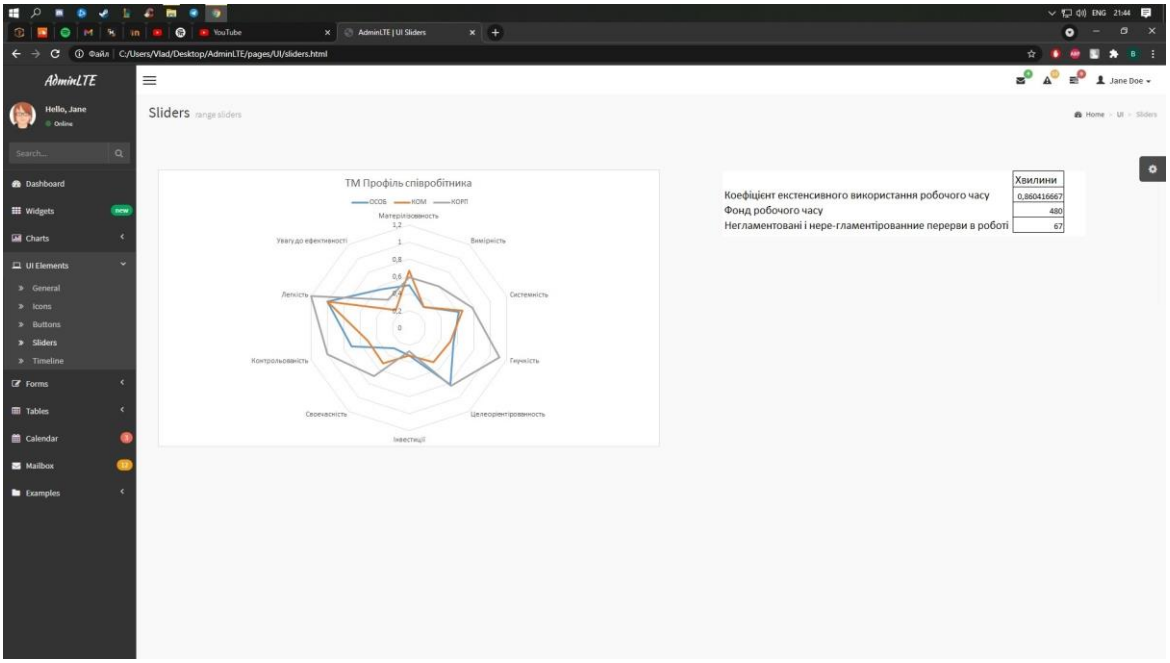

Рисунок 4.11 – Сторінка ціни часу співробітника

ВИСНОВКИ

В результаті виконаної дипломної кваліфікаційної роботи було створено систему підтримки керування проектами. Метою створення DISTRIBUTION SYSTEM OF IT-PROGECT (DSITP) є оптимізація робочого прогресу по розробці програмного забезпечення та веб-додатків на замовлення по офшорній моделі. DSWHC є системою по типу додатків task manager та має стандартний набір функцій такого типу систем для здійснення взаємозв'язків між різними проектними підгрупами. При цьому DSITP має специфічний функціонал, що відповідає потребам замовника. Система має специфічну архітектуру, розроблену спеціально для цього завдання, яка базується на не прямому множинному успадкуванні мається на увазі прийняття методів та властивостей одного класу іншим без прямого успадкування. Реалізується це через спадкування базового класу іншим та створення об’єкту в класі-спадкоємці для роботи з методами реалізованими у базовому класі. запропонована мною реалізація архітектури забезпечує швидкість роботи системи і тим самим знімає навантаження з сервера і бази даних.

Глобально систему можна розділити на три частини:

– підсистема моніторингу роботи програмістів та спілкування з замовником для менеджерів та адміністраторів;

– підсистема моніторингу напрацьованих годин, розподілення завдань для програмістів;

– СКБД;

Розподіл завдань між підгрупами та працівниками виконується згідно їх професійним компетенціям та навичкам та є активним фактором формування зарплатні. DSITP відслідковує результати роботи за кожним активним проектом, та має інструменти з їх оприлюднення. У системі було реалізовано:

– лист «TO DO», на якому міститься перелік активних проектів/завдань з можливістю їх обговорення в окремій гілці;

– сторінка інфорозклади, що містить особові рейтинги виконавців, загальні рейтинги підрозділів, стан виконання кожного активного проекту та його загальний час виконання;

– розрахунок зарплати в залежності від часу виконання проекту та особових рейтингів виконавців та рейтингів підрозділів;

– репозиторій активних та вже закінчених проектів організації з різними ступенями доступу для внутрішнього використання.

Підсистема зберігання даних являє собою базу даних представлену сервером PostgreSQL, який був обраний серед інших некомерційних СКБД завдячуючи своєму функціоналу.

Проект DSITP реалізовано на фреймворку Yii2, який базується на мові PHP. Крім збору й зберігання даних, у проекті реалізоване візуальне відображення даних у вигляді різних таблиць і розкладів у режимі реального часу; складання різних звітів; архівація й захист даних.

Для реалізації даного програмного продукту були використані NW.js та фреймворки Yii2, так як вони значно прискорять створення даного програмного продукту та забезпечать стабільність його роботи на всіх платформах.

Програмне забезпечення працює на 32 та 64 бітних платформах Windows, Mac OS X та Linux. Система має зручний, максимально орієнтований на користувача інтерфейс.

Система була протестована, її відповідність технічному завданню та працездатність підтверджено.

ПЕРЕЛІК ПОСИЛАНЬ

1. **Довгань Л. Є.**, «Керування проектами»: навчальний посібник[Текст]/ Г.А.Мохонько, І.П Малик .К.: КПІ ім. Ігоря Сікорського, 2017. – 420 с.

2. IEEE Std 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology. [Електронний ресурс]. – New York, 1990. – 83 p. – Режим доступу: https://ieeexplore.ieee.org/document/159342//(Дата звернення 04.04.2021)

3. EEE Std 1074-1995, IEEE Standard for Developing Software Life Cycle Processes. [Електронний ресурс]. – New York, 1990. – 83 p. – Режим доступу: https://ieeexplore.ieee.org/document/511093//(Дата звернення 04.05.2021)

4. «Керівництво до зведення знань по програмній інженерії»., , [Текст]/изд. IEEE Computer Society Press – 2004р – 312 стр

5. **David Rubinstein**, «Standish Group Report: There‘s Less Development Chaos Today». [Електронний ресурс] – New York, 2007 – Режим доступу:(http://www.sdtimes.com/content/article.aspx?ArticleID=30247)//(Дата звернення 04.04.2021)

6. **Гудкова К. Ю**. МЕТОДИ ТА ПІДХОДИ ДО ОЦІНКИ ЕФЕКТИВНОСТІ ІТ-ПРОЕКТІВ / К. Ю. Гудкова, А. О. Лях //Економічний вісник Донбасу № 3(45), [Електронний ресурс] – Донбас, 2016 – Режим доступу: http://dspace.nbuv.gov.ua/handle/123456789/113978 //(Дата звернення 04.05.2021)

7. **Листер Т.** Управление рисками в проектах по разработке программного обеспечения.[Текст]/ Т. Листер, Т. ДеМарко – М: Компания p.m. Office, 2005. – 196 с

8. **Paulk, Mark** ., Capability Maturity Model for Software, Version 1.1 (CMU/SEI-93-TR-24). [Текст]/ Изд. Software Engineering Institute Carnegie Mellon University, –1993р. – 435 стр.

9. **Пилип Крачтен**, «Введення в Rational Unified Process», [Текст]/ изд. Вільямс, 2002р – 276с.

10. «MSF, Microsoft, Microsoft Solutions Framework», Відділ MSF, Microsoft . [Електронний ресурс] – California, 2008 – Режим доступу:(https://visualstudio.microsoft.com/ru/ //(Дата звернення 26.04.2021)

11. **M. Pomeroy-Huff,** «The Personal Software Process (PSP) Body of Knowledge», version 1.0,/ J. Mullaney, R. Cannon, M. Sebern, — 2005р [Електронний ресурс] – Edinburgh, 2005 – Режим доступу:(https://visualstudio.microsoft.com/ru) //(Дата звернення 26.04.2021)

12. **Watts S. Humphrey**, «The Team Software Process (TSP)», Technical Report CMU/SEI, [Електронний ресурс] – New York, 2000р. – Режим доступу: (https://www.jetbrains.com/) (Дата звернення 01.04.2021)

13. **Kent Beck,** «Manifesto for Agile Software Development», [Електронний ресурс] – New York, 2001 – Режим доступу: (http://www.agilemanifesto.org/)(Дата звернення 01.04.2021)

14. **З. Макконнелл,** «Залишитися в живих. Керівництво для менеджерів програмних проектів», ,[Текст]/ изд. «Пітер» — 2006р. – 322 стр.

15. **Кастро Э.**, "HTML и CSS для создания web-страниц" ,[Текст]/, изд. НТ Пресс, 2006р. — 144 стр.

16. Самоучитель PHP 5. ,[Текст]/ — Издание 2-е — СПб.: Наука и техника, 2005р. — 576 с.

17. **Харрис Э**. PHP/MySQL для начинающих,[Текст]/. Пер. с англ. — М.: Кудиц — изд. Образ - 2005 — 200 с.

18. **Люк Веллинг**, «Разработка Web-приложений с помощью PHP и MySQL»[Текст]/. Лора Томсон – изд. «Вильямс» – Москва, –2003р – 301 стр.

19. "Достоинства и недостатки систем управления сайтом CMS" [Електронний ресурс] – New York, 2001 – Режим доступу: (http://www.webstudio55.ru)(Дата звернення 01.05.2021)

20. "Недостатки PHP" [Електронний ресурс] – 2015р. – Режим доступу: (http://comphobby.ru) (Дата звернення 27.04.2021)

21. "Perl: лучший язык программирования" [Електронний ресурс] – 2015р. – Режим доступу:(http://effectiveperl.com) (Дата звернення 25.04.2021)

22. «В помощь Веб-Мастеру» создание сайта, программирование, книги и учебники по php, html, css, java script. [Електронний ресурс] – 2015р. – Режим доступу:(http://wm-help.net/)(Дата звернення 25.05.2021)

24. Yii2 фреймворк документація[Електронний ресурс] – 2015р. – Режим доступу:(http://www.yiiframework.com/doc-2.0)(Дата звернення 25.05.2021)

25 Управління проектом: основні аспекти та функції [Електронний ресурс] – «букліб» - розід 9.1 – Режим доступу:(https://buklib.net/books/34104/)//(Дата звернення 26.04.2021)

Додаток А Технічне завдання

ВСТУП

Розроблюваний програмний продукт призначений для керування робітниками окремої організації.

Система створювалась максимально гнучкою для використання як і в невеликій організації (5-10 чол.) так і в середніх (20-50 чол). База данних та функціонал при внесенні невеликих змін дозволить використовувати ПЗ навіть в великих організаціях.

А.1 Підстава для розробки

А.2 Призначення для розробки

Розроблюваний програмний продукт призначений для оптимізації робочого прогресу по розробці програмного забезпечення та веб-додатків на замовлення по офшорній моделі..

А.3 Основні вимоги до розроблюваної програми

А.3.1 Вимоги до функціональних характеристик

Метод, які мають виконуватися програмним продуктом:

– відстежувати напрацьовані години;
– можливість роботи з фінансами ;
– отримання інформації по кожному проекту;
– планувати зустрічі, події та дата закінчення проекту;
– керування робітниками;
– керування доступом до інформації в залежності від посади;

– створення звітів;
– налаштування застосунка;
– налаштування профілю;
– система бекапів/імпорту/експорту;
– підтримка декількох мов;

А.3.2 Вимоги до надійності

Програма має забезпечувати коректну обробку помилок у випадку виключних ситуацій.

А.3.3 Умови експлуатації

Для роботи з програмою користувач має володіти базовими знаннями користування комп'ютера.

А.3.4 Вимоги до складу і параметрів технічних засобів

Програма повинна працювати на ПЕОМ із складом та параметрами технічних засобів: 2Gb RAM, мінімум 1 Gb вільного місця на жорсткому диску.

А.3.5 Вимоги до інформаційної та програмної сумісності

В склад технічних засобів повинен входити IBM-сумісний персональний комп’ютер. Середовищем функціонування програми є 32 або 64 бітна платформа Windows, Mac OS X та Linux.

А.3.6 Вимоги до програмної документації

Програма має поставлятися з технічним завданням та керівництвом оператора.

Додаток Б Текст програми

```
<?php
	namespace common\models;
	use Yii;
	use yii\base\NotSupportedException;
	use yii\behaviors\TimestampBehavior;
	use yii\db\ActiveRecord;
	use yii\web\IdentityInterface;

	use common\models\UserBalance;
	use common\models\UserMeta;
	/**
	 * User model
	 *
	 * @property integer $id
	 * @property string $username
	 * @property string $first_name
	 * @property string $last_name
	 * @property string $username
	 * @property string $password_hash
	 * @property string $password_reset_token
	 * @property string $email
	 * @property string $auth_key
	 * @property string $company
	 * @property integer $status
	 * @property integer $last_login
	 * @property integer $created_at
```

```
 * @property integer $updated_at
 * @property string $password write-only password
 */
class User extends ActiveRecord implements IdentityInterface
{
    const STATUS_DELETED = 0;
    const STATUS_ACTIVE = 10;

    public $meta = [];

    /**
     * @inheritdoc
     */
    public static function tableName()
    {
        return '{{%user}}';
    }

    /**
     * @inheritdoc
     */
    public function rules()
    {
        return [
            [['first_name', 'last_name', 'email'], 'required'],
            [['first_name', 'last_name', 'email'], 'trim'],
            ['email', 'unique', 'targetAttribute' => 'email', 'targetClass' =>
'\common\models\User', 'when' => function($model){
                return $this->isAttributeChanged('email');
            }],
```

```
            ['status', 'default', 'value' => self::STATUS_ACTIVE],
            ['status', 'in', 'range' => [self::STATUS_ACTIVE, self::STATUS_DELETED]],
            [['salary_type'], 'in', 'range' => ['fixed', 'hourly']]
        ];
    }

    /**
     * @inheritdoc
     */
    public function behaviors()
    {
        return [
            TimestampBehavior::className(),
        ];
    }

    public function getUser_meta()
    {
        return $this->hasMany(UserMeta::className(), ['user_id' => 'id']);
    }

    public function getDisplay_name()
    {
        return $this->first_name . ' ' . $this->last_name;
    }

    public function getProjects()
    {
        return $this->hasMany(Project::className(), ['client_id' => 'id']);
```

```
    }

    /**
     * @inheritDoc
     */
    public function beforeSave($insert) {
        return parent::beforeSave($insert);
    }

    /**
     * @inheritDoc
     */
    public function afterSave($insert, $changedAttributes){
        parent::afterSave($insert, $changedAttributes);

        $this->saveMetaFields($insert);
    }

    /**
     * Save user meta
     *
     * @param bool $insert If `false`, it means the method is called while updating a record.
     * @return void
     */
    public function saveMetaFields($insert)
    {
        $postadata = Yii::$app->request->post('User');
        $meta = $postadata['meta'] ?? [];
```

```
            if( !empty($meta) ) {

                if( !$insert ) {
                    foreach ($meta as $meta_key => $meta_value) {
                        $this->updateMeta($meta_key, $meta_value);
                    }
                } else {
                    foreach ($meta as $meta_key => $meta_value) {
                        $this->meta[] = [
                            'user_id' => $this->id,
                            'meta_name' => $meta_key,
                            'meta_value' => is_array($meta_value)? serialize($meta_value) : $meta_value
                        ];
                    }

                    $userMeta = new UserMeta;

                    Yii::$app->db->createCommand()->batchInsert(UserMeta::tableName(), ['user_id', 'meta_name', 'meta_value'], $this->meta)->execute();
                }
            }
        }

        /**
         * @param string $id [User ID]
         */
        public static function findIdentity($id)
        {
```

```
        return static::findOne(['id' => $id, 'status' => self::STATUS_ACTIVE]);
    }

    /**
     * @param string $token [description]
     * @param string $type
     */
    public static function findIdentityByAccessToken($token, $type = null)
    {
        throw new NotSupportedException('"findIdentityByAccessToken" is not implemented.');
    }

    /**
     * Finds user by email
     *
     * @param string $email
     * @return static|null
     */
    public static function findByEmail($email)
    {
        return static::findOne(['email' => $email, 'status' => self::STATUS_ACTIVE]);
    }

    public static function findByEmailAndRole($email, $role)
    {
        return static::find()
            ->select('user.*')
            ->leftJoin('auth_assignment', 'auth_assignment.user_id = user.id')
```

```
        ->where(['user.email' => $email])
        ->where(['auth_assignment.item_name' => $role])
        ->with('auth_assignment')
        ->one();
}

/**
 * Finds user by password reset token
 *
 * @param string $token password reset token
 * @return static|null
 */
public static function findByPasswordResetToken($token)
{
    if (!static::isPasswordResetTokenValid($token)) {
        return null;
    }

    return static::findOne([
        'password_reset_token' => $token,
        'status' => self::STATUS_ACTIVE,
    ]);
}

/**
 * Finds out if password reset token is valid
 *
 * @param string $token password reset token
 * @return bool
 */
```

```
public static function isPasswordResetTokenValid($token)
{
    if (empty($token)) {
        return false;
    }

    $timestamp = (int) substr($token, strrpos($token, '_') + 1);
    $expire = Yii::$app->params['user.passwordResetTokenExpire'];
    return $timestamp + $expire >= time();
}

/**
 * @inheritdoc
 */
public function getId()
{
    return $this->getPrimaryKey();
}

/**
 * @inheritdoc
 */
public function getAuthKey()
{
    return $this->auth_key;
}

/**
 * @param string $authKey
 */
```

```
    public function validateAuthKey($authKey)
    {
        return $this->getAuthKey() === $authKey;
    }

    /**
     * Validates password
     *
     * @param string $password password to validate
     * @return bool if password provided is valid for current user
     */
    public function validatePassword($password)
    {
        return Yii::$app->security->validatePassword($password, $this->password_hash);
    }

    /**
     * Generates password hash from password and sets it to the model
     *
     * @param string $password
     */
    public function setPassword($password)
    {
        $this->password_hash = Yii::$app->security->generatePasswordHash($password);
    }

    /**
     * Generates "remember me" authentication key
```

```
     */
    public function generateAuthKey()
    {
        $this->auth_key = Yii::$app->security->generateRandomString();
    }

    /**
     * Generates new password reset token
     */
    public function generatePasswordResetToken()
    {
        $this->password_reset_token = Yii::$app->security->generateRandomString() . '_' . time();
    }

    /**
     * Removes password reset token
     */
    public function removePasswordResetToken()
    {
        $this->password_reset_token = null;
    }

    public function getMeta($key)
    {
        $userMeta = new UserMeta();
        return $userMeta->getMeta($this->id, $key);
    }

    public function addMeta($key, $value=null)
```

```
        {
            $userMeta = new UserMeta();
            return $userMeta->addMeta($this->id, $key, $value);
        }

        public function updateMeta($key, $value=null)
        {
            $result = false;
            $meta_field = UserMeta::find()->where(['user_id' => $this->id, 'meta_name' => $key])->one();

            if( !$meta_field ) {

                $meta = new UserMeta;
                $meta->user_id = $this->id;
                $meta->meta_name = $key;
                $meta->meta_value = is_array($value)? serialize($value) : $value;
                $result = $meta->save();

            } else {

                $meta_field->meta_value = is_array($value)? serialize($value) : $value;
                $result = $meta_field->update();

            }

            return $result;
        }
```

```
    public function deleteMeta($key)
    {
        $userMeta = new UserMeta();
        return $userMeta->getMeta($this->id, $key);
    }

    public function getUserBalance()
    {
        return $this->hasOne(UserBalance::className(), ['user_id' => 'id']);
    }

    public function getCurrency()
    {
        return $this->hasOne(Currency::className(), ['id' => 'currency']);
    }
}
<?php
namespace backend\controllers;

use Yii;
use common\models\Currency;
use common\models\UserBalance;
use yii\data\Pagination;
use yii\filters\AccessControl;
use yii\data\ActiveDataProvider;
use yii\web\NotFoundHttpException;
use phpDocumentor\Reflection\Types\Integer;
use yii\web\Controller;
```

```
use yii\web\Response;
use backend\models\UserForm;
use common\models\User;
use common\models\Options;
use common\models\Exercise;
use common\web\BaseController;
use yii\bootstrap\ActiveForm;


/**
 * Site controller
 */
class UserController extends BaseController
{
    /**
     * @inheritdoc
     */
    public function behaviors()
    {
        return [
            'access' => [
                'class' => AccessControl::className(),
                'rules' => [
                    [
                        'actions' => [
                            'index', 'add', 'edit', 'delete'
                        ],
                        'allow' => true,
                        'roles' => ['admin'],
                    ],
```

```
            ],
        ]
    ];
}

/**
 * @inheritdoc
 */
public function actions()
{
    return [
        'error' => [
            'class' => 'yii\web\ErrorAction',
        ],
    ];
}

/**
 * Displays homepage.
 *
 * @return string
 */
public function actionIndex()
{
    $current_user = Yii::$app->user;

    $query = User::find()->where(['!=', 'id', $current_user->id]);
    $search = Yii::$app->request->get('s');
    $filter = Yii::$app->request->get('role');
```

```
        if( $search ) {
            $search = explode(' ', $search);
            $filter = ['or'];

            foreach( $search as $search_word ) {
                $filter[] = ['like', 'first_name', $search_word];
                $filter[] = ['like', 'last_name', $search_word];
                $filter[] = ['like', 'email', $search_word];
            }

            $query->andFilterWhere( $filter );
        }

        if ( $filter ) {
            $usersId = Yii::$app->authManager->getUserIdsByRole($filter);
            $query->andFilterWhere(['id' => $usersId]);
        }
        return $this->render('index', [
            'query' => $query,
        ]);
    }

    public function actionAdd()
    {
        $model = new UserForm();
        $userBalanceModel = new UserBalance;
        $balanceCurrency = Currency::find()->indexBy('id')->all();
        $arrCurrencys = [];
        $userBalanceData = [];
        $newUserId = '';
```

```
            $postdata = Yii::$app->request->post();

            foreach ($balanceCurrency as $value) {
                $arrCurrencys[$value->id] = $value->code;
            }


            if (Yii::$app->request->isAjax && $model->load($postdata)) {
                Yii::$app->response->format = Response::FORMAT_JSON;
                return ActiveForm::validate($model);
            }


            if ($model->load(Yii::$app->request->post()) && $model->save()){
                $newUserId = User::find()->indexBy('id')->where(['email' => $model->email])->one();
                $userBalanceData = [
                    'user'      => $newUserId->id,
                    'currency'  => $model->currency,
                ];
                if ($userBalanceModel->load($userBalanceData) && $userBalanceModel->save()) {
                    return $this->redirect('../user');
                }
            }

            return $this->render('add', [
                'model' => $model,
                'arrCurrencys' => $arrCurrencys
            ]);
```

```
}

public function actionEdit( $id )
{
   $model = new UserForm();
   $user = $model->findOne(['id' => $id]);
   $balanceCurrency = Currency::find()->indexBy('id')->all();
   $arrCurrencys = [];

   foreach ($balanceCurrency as $value) {
      $arrCurrencys[$value->id] = $value->code;
   }

   $model->id = $user->id;
   $model->first_name = $user->first_name;
   $model->last_name = $user->last_name;
   $model->email = $user->email;
   $model->salary_type = $user->salary_type;
   $model->currency = $user->currency;
   $model->salary = $user->salary;

   $postdata = Yii::$app->request->post();
   if (\Yii::$app->request->isAjax && $model->load($postdata)) {
     \Yii::$app->response->format = Response::FORMAT_JSON;
     return ActiveForm::validate($model);
   }

   if ($model->load($postdata) && $model->save()){
   }
```

```
            return $this->render('add', [
                'model' => $model,
                'user' => $user,
                'arrCurrencys' => $arrCurrencys,
            ]);
        }

        public function actionDelete( $id )
        {
            $model = new UserForm();
            $current_user = Yii::$app->user;
            if (Yii::$app->user->can('manage' . ucfirst(key(Yii::$app->authManager->getRolesByUser($id))))){
                if (User::find()->where(['!=', 'id', $current_user->id]) !== $id){
                    $user = $model->findOne(['id' => $id]);
                    $user->delete();
                }
            $this->redirect(['user/index']);
            }
        }
    }
```