

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 122 «Комп'ютерні науки»
(освітня програма «Комп'ютерна графіка та розробка ігор»)
на тему:

**«Система керування завданнями для працівників
організації»**

Виконав студент групи ЗКН-19-1
КУРАЄВ Владислав Леонідович

Керівник роботи:
КВАСНІКОВ Володимир Павлович

Консультант:
ГРАФ Марина Сергіївна

Рецензент:
ЛЕВКІВСЬКИЙ Віталій Леонідович

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ЗАТВЕРДЖУЮ

Завідувач кафедри комп'ютерних наук

Марина ГРАФ

«05» січня 2023 р.

ЗАВДАННЯ

на кваліфікаційну роботу освітнього ступеня «бакалавр»
за спеціальністю 122 «Комп'ютерні науки»
(освітня програма «Комп'ютерна графіка та розробка ігор»)

Здобувач вищої освіти: **КУРАЄВ Владислав Леонідович**

Керівник роботи: **КВАСНІКОВ Володимир Павлович**

Тема роботи: **«Система керування завданнями для працівників організації»**,
затверджена наказом закладу вищої освіти від **«05» січня 2023 р., №11/с**

Термін здачі закінченої роботи: **«10» червня 2023 р.**

Вихідні дані роботи (зазначається предмет і об'єкт дослідження): об'єкт дослідження – веб-орієнтовані системи для керування завданнями в межах організації. Предмет дослідження – реалізація функціоналу по обробці інформаційних процесів щодо керування завданнями для працівників організації.

Консультанти випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Дата	
		Завдання видав	Завдання прийняв
1	ГРАФ Марина	02.02.2023 р.	02.02.2023 р.
2	ГРАФ Марина	03.04.2023 р.	03.04.2023 р.
3	ГРАФ Марина	15.05.2023 р.	15.05.2023 р.

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі. Пошук, огляд та аналіз аналогічних розробок. Формулювання технічного завдання. Опрацювання літературних джерел.	2 лютого 2023 – 5 березня 2023	Виконано
2	Проектування структури системи	6 березня 2023 – 25 березня 2023	Виконано
3	Написання програмного коду	26 березня 2023 – 26 квітня 2023	Виконано
4	Тестування системи	27 квітня 2023 – 14 травня 2023	Виконано
5	Оформлення пояснювальної записки	15 травня 2023 – 1 червня 2023	
6	Попередній захист	1 червня 2023	Виконано

Здобувач вищої освіти **Владислав КУРАЄВ**

Керівник **Володимир КВАСНІКОВ**

РЕФЕРАТ

Кваліфікаційна робота складається з вихідного коду, SQL-файлу для імпорту структури БД та тестових даних, пояснювальної записки. Пояснювальна записка до випускної роботи має 69 сторінок друкованого тексту і містить 20 рисунків та 10 таблиць. Список використаних джерел містить 14 найменувань.

Мета роботи: здійснити розробку веб-орієнтованої інформаційної системи з реалізацією базового функціоналу для керування завданнями для працівників організації.

Здійснено огляд аналогічних систем, визначено варіанти використання та базові функціональні особливості системи. Описано деталі технічного завдання. Зроблено вибір інфраструктури для розробки: інструментів, мови програмування, фреймворку, модулів, бібліотек тощо.

Спроектовано та описано необхідні UML-діаграми: Use Case, Activity, Sequence, Classes. Наведено типові кроки щодо розробки одного з модулів системи.

Розроблено інструкцію користувача. Наведено кроки по клонуванню та впровадженню проекту. Проведено тестування системи, вирішено всі помилки та недоліки.

КЛЮЧОВІ СЛОВА: ОРГАНІЗАЦІЯ, ПРАЦІВНИКИ, ЗАВДАННЯ

					КН.КР.Б – 122 – 23 – ПЗ							
Змн.	Арк.	№ докум.	Підпис	Дата								
Розроб.	В.Л. КУРАЄВ				Система керування завданнями для працівників організації Пояснювальна записка				Літ.	Арк.	Аркушів	
Керівник	В.П. КВАСНІКОВ										3	64
Рецензент	В.Л. ЛЕВКІВСЬКИЙ								Житомирська політехніка, група ЗКН-19-1			
Н. контр.												
Зав. каф.	М.С. ГРАФ											

ABSTRACT

The bachelor degree work consists of the source code, an SQL file for importing the database structure and test data, and an explanatory note. The explanatory note has 69 pages and contains 20 figures and 10 tables. The list of used sources contains 14 names.

The purpose of the work is to develop a web-oriented information system with the implementation of basic functionality for managing tasks for the employees in organization.

An overview of similar systems was carried out, use cases, advantages and basic functional features of the system were determined. The details of the technical task are described. The infrastructure for development has been selected: tools, programming language, framework, modules, libraries, etc.

Designed and described the necessary UML diagrams: Use Case, Activity, Sequence, Classes. Typical steps for developing one of the system modules are given.

A user manual has been developed. The steps for cloning and implementation of the project are given. The system has been tested, all errors and shortcomings have been resolved.

KEY WORDS: ORGANIZATION, EMPLOYEES, TASKS.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		5

ЗМІСТ

ВСТУП	8
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ВЕБ-ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ІНФОРМАЦІЙНИХ СИСТЕМ КЕРУВАННЯ ЗАВДАННЯМИ.....	12
1.1. Постановка задачі	12
1.2. Огляд аналогічних систем і продуктів	15
1.3. Вибір інфраструктури розробки: платформа, веб-сервер, мова програмування, фреймворк, бібліотеки.....	19
Висновки до розділу 1	22
РОЗДІЛ 2. ПРОЕКТУВАННЯ І РОЗРОБКА СИСТЕМИ КЕРУВАННЯ ЗАВДАННЯМИ ДЛЯ ПРАЦІВНИКІВ В ОРГАНІЗАЦІЇ	23
2.1. Базові функції та варіанти використання системи.....	23
2.2. UML-діаграми діяльності, послідовностей, класів	25
2.3. Структура бази даних системи.....	31
2.4. Написання програмного коду модуля системи	35
Висновки до розділу 2	40
РОЗДІЛ 3. ОПИС ВЕБ-ІНТЕРФЕЙСУ СИСТЕМИ. ІНСТРУКЦІЯ КОРИСТУВАННЯ І ВСТАНОВЛЕННЯ. ТЕСТУВАННЯ СИСТЕМИ КЕРУВАННЯ ЗАВДАННЯМИ.....	41
3.1. Керівництво встановлення системи.....	41
3.2. Опис веб-інтерфейсу. Інструкція користування системою	43
3.3. Тестування веб-системи для керування завданнями	48
Висновки до розділу 3	51
ВИСНОВКИ.....	52
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	54
ДОДАТКИ.....	56

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

СКБД – Система керування базами даних

PM – Project Management – проектний менеджмент

TMS – Task Management System – система керування завданнями

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми. В сучасних компаніях більшість форм діяльності відбуваються у формі проектів. Деякі проекти завершуються успішно, інші проекти провалюються. Опитування, проведене компанією PwC, показало, що понад 60 відсотків невдач проектів пов'язані з внутрішніми проблемами проекту (наприклад, порушення термінів виконання, недостатні ресурси). Виділено три головні причини провалів проектів – погані оцінки/пропущені терміни, зміни обсягу та недостатні ресурси — це внутрішні фактори проекту, і лише ці три категорії складають 50 відсотків причин провалу проекту.

Існує ряд програмних рішень для керування завданнями та проектами. Наявність такого рішення для керування проектами, яке допомагає проектам на всіх етапах життєвого циклу, дає організаціям внутрішню перевагу над конкурентами та підтримує їх організованість.

Інструменти управління завданнями дають компаніям достатній вплив, щоб максимізувати прибуток і правильно організувати робочий час, зокрема для розширення свого ринку. Ефективне застосування програмного забезпечення для управління завданнями дає організаціям перевагу перед компаніями, які ще не впровадили такі системи.

Постійне розширення бізнесу та технічний розвиток підтвердили важливість програмного забезпечення для управління завданнями та контролю над проектами. Наведемо обґрунтування доцільності використання програмних засобів для управління завданнями.

1. Моніторинг виконання завдань

Одним із ключових аспектів задоволення вимог проекту є моніторинг прогресу виконання завдань проекту. Такий моніторинг дає можливість якісно організувати інші процеси: управління запасами, планування та обліку робочого часу, система закупівель тощо. Програмні інструменти для управління проектами дозволяють організаціям відстежувати хід і ефективність проекту.

2. Спрощення взаємодій працівників

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

Командна робота відіграє першорядну роль у будь-якій організації. Проте індивідуальний внесок становить більшу частину великих проектів. Інструменти управління завданнями прокладають шлях до кращої командної роботи, допомагаючи кожному працівнику підтримувати стандарти. Програмне забезпечення для керування завданнями забезпечує інтерфейси для часових шкал, оновлень у реальному часі та керування документами.

Наприклад, програмне забезпечення управління завданнями може забезпечити регулярні оновлення та спілкування для досягнення необхідної ефективності. Система автоматичних сповіщень інформує працівників та менеджерів про хід виконання певних завдань, щоб забезпечити швидші зв'язки та спілкування між членами команди.

3. Внутрішній і зовнішній зв'язки

Незалежно від організаційної ієрархії, недостатність комунікації можуть мати негативний вплив для будь-якої компанії. Одна з ключових цілей програмного забезпечення для управління завданнями – допомогти працівниками співпрацювати в режимі реального часу. Завдяки ефективним інструментом управління завданнями організації отримують вигоду від ефективного спілкування між користувачами, командами, відділами та зацікавленими сторонами. Усі учасники мають доступ до стану завдань в режимі реального часу, таким чином проектами керують краще і ефективніше.

4. Управління бюджетом проекту

Проекти, як правило, пов'язані з витратами, які залежать від правильності планування бюджету. Інструменти управління проектами допомагають планувати та контролювати бюджет, керуючи вартістю та витратами. Упорядкування рахунків-фактур і документації може бути досить виснажливим, особливо у великому масштабі; у більшості випадків створення та оновлення простих електронних таблиць не допомагає.

5. Оптимізований тайм-менеджмент і планування

Управління часом є невід'ємною частиною структури організації. Продуктивність і організація типового робочого місця можуть бути визначені

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		9

наявною системою планування. Використання інструменту управління завданнями має велике значення для організації та планування. Більшість програмних рішень для управління проектами розроблені, щоб допомогти компаніям втілювати найскладніші завдання, окреслюючи ієрархію важливості кожного завдання та контролюючи управління змінами. Вони також мають важливе значення для надання деталей про часовий проміжок, виділений кожному члену команди, і встановлення нагадувань про крайні терміни. Однією з основних переваг інструментів управління завданнями в цьому варіанті залишається те, що існує підтримка методів управління часом і мінімізація непорозумінь у процесі.

6. Більш продуктивний процес прийняття рішень

Процес прийняття рішень є вимагає точності, акуратності та ретельного планування. Належні інструменти управління завданнями відіграють важливу роль в організації цього аспекту.

Таким чином основними цілями системи керування завданнями є:

- дотримання термінів і вчасне завершення завдань і проектів
- контроль бюджету
- покращення командної співпраці та комунікації
- підвищення продуктивності
- керування очікуваннями зацікавлених сторін
- оптимальне використання ресурсів проекту
- підвищення ефективності командних зустрічей
- створення культури, керованої даними
- спрощення спілкування та співпраці з клієнтами
- постійне вдосконалення і розвиток з використанням нових практик

Метою роботи є програмна реалізація функцій системи управління завданнями для працівників в організації у відповідності з цілями по ефективній реалізації проектів, розумним використанням ресурсів, контроль бюджету, покращення командної роботи та спілкування з клієнтами.

Завдання кваліфікаційної роботи:

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

- визначити тему, мету, цілі та функціональні можливості програмних засобів керування завданнями;
- сформулювати об'єкт і предмет дослідження;
- сформулювати задачу щодо розробки системи;
- визначити варіанти використання, здійснити опис базових функціональних вимог
- розглянути аналогічні системи, порівняти їх функціональні можливості
- деталізувати технічне завдання, здійснити проектування бази даних
- розробити UML-діаграми варіантів використання, діяльностей, послідовностей, класів
- здійснити проектування системи ролей та дозволів
- написати програмний код системи
- здійснити тестування системи
- зробити висновки про результати розробки і дослідження
- написання пояснювальної записки та захист роботи

За **об'єкт дослідження** покладемо веб-орієнтовані системи для керування завданнями в межах організації.

Предметом дослідження є реалізація функціоналу по обробці інформаційних процесів щодо керування завданнями для працівників організації.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ВЕБ-ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ІНФОРМАЦІЙНИХ СИСТЕМ КЕРУВАННЯ ЗАВДАННЯМИ

1.1. Постановка задачі

При розробці веб-орієнтованих систем для керування завданнями розробники мають вирішити наступний ряд задач:

Організація роботи з завданнями: розробка системи для створення і ефективної обробки завдань має вирішальне значення. Розробникам необхідно створити структуру, яка дозволить користувачам класифікувати, пріоритезувати та ефективно призначати завдання. Це передбачає розробку інтерфейсів користувача для створення, редагування та видалення завдань, а також впровадження таких функцій, як мітки завдань, теги, терміни та залежності.

Дозволи користувача та контроль доступу: програмне забезпечення для керування завданнями часто залучає кількох користувачів із різними ролями та дозволами. Реалізація функцій автентифікації, авторизації та контролю доступу важлива для того, щоб користувачі могли переглядати, редагувати та керувати завданнями на основі призначених їм ролей і дозволів. Це завдання вимагає ретельного розгляду ролей користувачів, рівнів дозволів і заходів захисту даних.

Відстеження завдань і моніторинг прогресу: система для керування завданнями повинна надавати користувачам можливість перегляду статусів і прогресу завдань. Це передбачає впровадження таких функцій, як списки завдань, дошки Kanban, діаграми Ганта чи інші візуальні представлення завдань. Розробники повинні обробляти оновлення завдань, сповіщення та функції співпраці в режимі реального часу, щоб інформувати користувачів про зміни та прогрес завдань.

Інтеграція з електронною поштою та месенджерами: багато систем керування завданнями потребують інтеграції з застосунками, такими як електронна пошта, чат або платформи для спільної роботи над проектами. Це дозволяє користувачам без проблем спілкуватися та співпрацювати в програмному забезпеченні для керування завданнями. Розробникам потрібно

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

реалізувати такі функції, як сповіщення, обмін повідомленнями та інтеграційні API, щоб з'єднатися із зовнішніми інструментами зв'язку.

Звітування та аналітика. Програмне забезпечення для керування завданнями часто містить функції звітності та аналітики для відстеження продуктивності команди, рівня виконання завдань та інших ключових показників. Розробникам необхідно розробити та впровадити функції звітності для створення корисних звітів і візуалізацій, які надають уявлення про хід проекту, продуктивність команди та розподіл ресурсів.

Мобільна та крос-платформна підтримка: щоб задовольнити потреби різноманітної бази користувачів, програмне забезпечення для керування завданнями часто потребує мобільних додатків або крос-платформної сумісності. Розробники повинні проектувати та розробляти мобільні програми для iOS та Android або гарантувати, що програмне забезпечення доступне та реагує на різні пристрої та платформи.

Продуктивність і масштабованість. Оскільки кількість завдань і користувачів має властивість зростати, система для керування завданнями повинна справлятися зі збільшенням обсягу даних і користувачів без втрати продуктивності. Розробники повинні оптимізувати запити до бази даних, реалізувати механізми кешування та розробити масштабовану архітектуру, щоб програмне забезпечення залишалось оперативним та ефективним.

Безпека та захист даних: інформаційна система для керування завданнями часто має справу з конфіденційними та конфіденційними даними, включаючи деталі завдань, інформацію про користувача та дані проекту. Впровадження надійних заходів безпеки, таких як шифрування даних, безпечна автентифікація користувачів і регулярне резервне копіювання, має вирішальне значення для захисту даних користувачів від несанкціонованого доступу або втрати даних.

Взаємодія з користувачем і зручність використання: забезпечення інтуїтивно зрозумілої та простої взаємодії з користувачем є важливим у системі для керування завданнями. Розробникам необхідно зосередитися на зручності використання, створювати естетично привабливі інтерфейси та проводити

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						13
Змн.	Арк.	№ докум.	Підпис	Дата		

тестування користувачів, щоб гарантувати, що користувачі можуть легко орієнтуватися в програмному забезпеченні та взаємодіяти з ним.

Кастомізація та розширюваність: система для керування завданнями має бути достатньо гнучкою, щоб адаптувати різні робочі процеси та параметри налаштування. Розробники враховують необхідність задавати параметри налаштування, такі як налаштування полів завдань, визначені користувачем робочі процеси та інтеграцію з інструментами сторонніх розробників, щоб задовольнити унікальні потреби різних користувачів і організацій.

Таким чином ми описали типові проблеми і завдання при створенні системи для керування завданнями. Здійснимо формулювання загальної задачі.

Формулювання задачі. Основною задачею розробки системи керування завданнями є створення веб-орієнтованого програмного застосунку, що дозволяє користувачам ефективно організовувати, відстежувати та керувати завданнями в індивідуальних або командних умовах. Мета полягає в тому, щоб надати комплексне рішення, яке дозволяє користувачам створювати, призначати, визначати пріоритети та відстежувати завдання, одночасно сприяючи співпраці, спілкуванню та продуктивності. Система керування завданнями повинна забезпечувати функції для

- створення та редагування завдань
- призначення та делегування завдань
- відстеження прогресу
- керування термінами виконання
- сповіщення та нагадування
- звітування та аналітику
- інтеграцію з електронною поштою, месенджерами та іншими програмними додатками.

Система має бути зручною, інтуїтивно зрозумілою та налаштовуваною відповідно до конкретних потреб і робочих процесів різних користувачів і організацій. Безпека, захист даних і оптимізація продуктивності також є важливими міркуваннями при розробці системи керування завданнями.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						14
Змн.	Арк.	№ докум.	Підпис	Дата		

Отже, метою даної роботи є розробка веб-орієнтованої інформаційної системи управління завданнями для працівників в організації для досягнення цілей ефективної реалізації проєктів, раціонального використання ресурсів, контролю бюджету, покращення командної роботи та спілкування з клієнтами. Визначено опис типових задач, що ставляться перед розробниками таких систем, і сформульовано основну задачу дослідження.

1.2. Огляд аналогічних систем і продуктів

Здійснимо огляд і порівняння функціональних можливостей декількох веб-орієнтованих систем керування завданнями: Todoist, Microsoft To Do, Google Tasks, Wunderlist.

Todoist

Функції системи: створення завдань, терміни виконання, пріоритети, мітки завдань і основні параметри співпраці.

Інтерфейс користувача: Todoist має простий та інтуїтивно зрозумілий інтерфейс із акцентом на мінімалістичний дизайн.

Платформи: застосунок доступний на різних платформах, включаючи веб, мобільні (iOS, Android) і настільні (Windows, macOS).

Інтеграція: система інтегрується з іншими інструментами, такими як Google Calendar, Dropbox і Zapier, що дозволяє користувачам оптимізувати робочі процеси.

Ціни: пропонується безкоштовний план із обмеженими можливостями, а також платні плани з додатковими функціями.

Microsoft ToDo

Функції системи: створення завдань, терміни виконання, нагадування та категоризація завдань.

Інтерфейс користувача: система має елегантний і сучасний інтерфейс, добре інтегрований з екосистемою Microsoft.

Платформи: застосунок доступний в Web, на мобільному пристрої (iOS, Android) і настільному комп'ютері (Windows).

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дата		

Інтеграція: система інтегрується з іншими службами Microsoft, такими як Outlook і Microsoft 365, що дозволяє легко синхронізувати завдання.

Ціни: використання системи є безкоштовним.

Google Tasks

Функції: Google Tasks пропонує просте створення завдань, терміни виконання, списки завдань і підзавдань.

Інтерфейс користувача: система має мінімалістичний і зрозумілий інтерфейс користувача, який добре підходить до інших програм Google.

Платформи: Завдання Google доступні в Інтернеті, на мобільних пристроях (iOS, Android) і інтегруються з іншими службами Google.

Інтеграція: Google Календар, Gmail і Google Keep, що полегшує керування завданнями разом з іншими інструментами Google.

Ціни: Google Tasks можна використовувати безкоштовно.

Wunderlist

Функції: створення завдань, терміни виконання, нагадування, підзавдання та основні функції співпраці.

Інтерфейс користувача: інтуїтивно зрозумілий і візуально привабливий інтерфейс, що робить керування завданнями приємним.

Платформи: доступна для Веб, мобільних пристроїв (iOS, Android) і комп'ютерів (Windows, macOS).

Інтеграція: інтегрується з різними програмами сторонніх розробників, такими як Dropbox, Slack і Microsoft Outlook, підвищуючи продуктивність і співпрацю.

Ціни: безкоштовний, але платна версія пропонує додаткові функції.

Варто зазначити, що Microsoft To Do замінила Wunderlist, і Microsoft представила нову систему керування завданнями під назвою Microsoft Lists, яка пропонує більш розширені функції, придатні для командної співпраці та керування проектами. Однак для простих потреб у управлінні завданнями

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		16

вищезазначені системи забезпечують зручні для користувача інтерфейси та важливі функції для організації та ефективного відстеження завдань.

Здійснивши огляд поширених сучасних систем керування завданнями, можна виділити і згрупувати ряд типових функцій системи, які спрямовані на підвищення продуктивності, співпраці та організації завдань. Наведемо загальні функції сучасних систем керування завданнями та коротке порівняння їхніх можливостей:

1. Створення та організація завдання:

- *Базове створення завдань:* усі системи керування завданнями дозволяють користувачам створювати завдання, додавати описи, встановлювати терміни виконання та призначати їх членам команди.
- *Пріоритезація завдань:* більшість систем пропонують можливість призначати пріоритети завданням, наприклад високий, середній або низький, щоб допомогти користувачам зосередитися на критичних завданнях.
- *Категоризація завдань:* користувачі можуть організовувати завдання за різними категоріями або проектами, що полегшує їх окреме керування та відстеження.

2. Відстеження завдань і моніторинг прогресу:

- *Відстеження прогресу:* сучасні системи забезпечують візуальні індикатори або шкали прогресу, щоб показати статус завдань, дозволяючи користувачам відстежувати їх виконання.
- *Залежності завдань:* деякі системи дозволяють користувачам визначати залежності між завданнями, гарантуючи, що завдання виконуються в певному порядку.
- *Відстеження часу:* вдосконалені системи керування завданнями дозволяють користувачам відстежувати час, витрачений на кожне завдання, допомагаючи розподіляти ресурси та аналізувати продуктивність.

3. Співпраця та спілкування:

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						17
Змн.	Арк.	№ докум.	Підпис	Дата		

- *Призначення та делегування завдань:* користувачі можуть призначати завдання конкретним членам команди та делегувати обов'язки, забезпечуючи чітке володіння.
- *Коментарі та обговорення:* системи часто включають функції залишати коментарі, проводити обговорення або прикріплювати файли до завдань, полегшуючи співпрацю та обмін інформацією.
- *Оновлення в режимі реального часу:* багато систем пропонують оновлення в режимі реального часу, сповіщаючи користувачів про зміни, внесені в завдання, гарантуючи, що всі залишаються в курсі подій.

4. Нагадування та сповіщення:

- *Нагадування:* користувачі можуть встановлювати нагадування або отримувати сповіщення про майбутні або прострочені завдання, щоб не відставати від своїх обов'язків.
- *Інтеграція електронної пошти:* деякі системи інтегруються з електронною поштою, дозволяючи користувачам отримувати пов'язані із завданнями сповіщення та оновлення безпосередньо у папку «Вхідні».

5. Візуалізація та організація завдання:

- *Дошки Kanban:* популяризовані гнучкими методологіями, дошки Kanban забезпечують візуальне представлення завдань, що дозволяє користувачам відстежувати їхній прогрес на різних етапах.
- *Діаграми Ганта:* ці діаграми пропонують часову шкалу завдань і їхніх залежностей, допомагаючи користувачам візуалізувати розклади проектів і кінцеві терміни.

6. Звітність і аналітика:

- *Аналітика завдань:* сучасні системи керування завданнями можуть надавати аналітику та звіти про рівень виконання завдань, продуктивність команди та показники продуктивності, допомагаючи користувачам оцінювати прогрес і приймати рішення на основі даних.

- *Настроювані звіти:* деякі системи пропонують можливість створювати спеціальні звіти на основі певних критеріїв, що дозволяє користувачам генерувати індивідуальні ідеї.

7. Мобільна та кросплатформна підтримка:

- *Мобільні програми:* багато систем керування завданнями пропонують мобільні програми для пристроїв iOS і Android, що дозволяє користувачам отримувати доступ до завдань і керувати ними на ходу.
- *Сумісність із різними платформами:* системи можуть підтримувати веб-доступ, забезпечуючи сумісність із різними операційними системами та пристроями.

8. Інтеграція зі сторонніми інструментами:

- *Можливості інтеграції:* сучасні системи керування завданнями часто інтегруються з іншими інструментами, такими як програмне забезпечення для керування проектами, комунікаційні платформи або служби обміну файлами, оптимізуючи робочі процеси та централізуючи інформацію.

Важливо зазначити, що функції та можливості систем керування завданнями можуть значно відрізнятися залежно від конкретного програмного забезпечення та його цільової аудиторії. Різні системи можуть надавати перевагу певним функціям над іншими, і діапазон функцій може відрізнятися між базовими інструментами керування завданнями та більш просунутими платформами керування проектами.

1.3. Вибір інфраструктури розробки: платформа, веб-сервер, мова програмування, фреймворк, бібліотеки

В таблиці 1.1 наведемо перелік інструментів розробки а також необхідного програмного забезпечення для реалізації програмного продукту:

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		

Стек та інструменти розробки системи керування завданнями

Назва проекту	Система керування завданнями для працівників в організації
Тип проекту	Веб-орієнтований застосунок
Стек	LAMP: Linux, Apache, MySQL, PHP
Операційна система	Linux Ubuntu 20.04
Веб-сервер	Apache/2.4.54
СКБД	MySQL 8.0
Мова розробки	PHP 8.1
Серверний фреймворк	Yii2
Мови клієнтської розробки	HTML5, CSS3, JavaScript
CSS-фреймворк	Bootstrap5
Адміністративний модуль	AdminLTE
Програмне забезпечення	OpenServer, Composer, Git, PHPStorm

Обґрунтуємо використання даної інфраструктури та інструментів для розробки веб-системи керування завданнями для працівників:

Відкритий вихідний код: усі компоненти стеку LAMP є відкритими: вони вільно доступні, мають широку підтримку та мають велику спільноту розробників, які роблять внесок у їх розвиток. Це забезпечує доступ до великої кількості ресурсів, документації та підтримки спільноти.

Стабільність і безпека: Linux відомий своєю стабільністю та безпекою, що робить його надійним вибором для серверних операційних систем. Apache, як веб-сервер, має довгу історію та велику увагу приділяє безпеці. Ця комбінація забезпечує надійну основу для безпечного розміщення веб-додатків.

PHP: PHP — це широко використовувана серверна мова сценаріїв, яка спеціально розроблена для веб-розробки. Вона має широку підтримку різноманітних фреймворків і бібліотек, що робить її популярною для швидкої розробки веб-орієнтованих систем. PHP також має хороші можливості інтеграції з базами даних, включаючи MySQL.

MySQL: MySQL — це популярна система керування реляційними базами даних із відкритим кодом. Її популярність пояснюється своєю продуктивністю,

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

масштабованістю та простотою використання. MySQL — це стабільний вибір для ефективного зберігання даних, пов'язаних із завданнями, і керування ними.

Yii2 Framework: Yii2 — це потужний PHP-фреймворк, який використовує архітектурний шаблон проектування Model-View-Controller (MVC). Він надає повний набір інструментів, бібліотек і функцій для швидкого створення надійних і масштабованих веб-додатків. Yii2 сприяє багаторазовому використанню коду, модульній розробці та слідує передовим практикам сучасної веб-розробки.

Масштабованість і продуктивність: стек LAMP відомий своєю масштабованістю, що дозволяє системі керування завданнями ефективно обробляти зростаючу кількість користувачів і завдань. Linux і Apache мають високу масштабованість, тоді як PHP і Yii2 пропонують методи оптимізації продуктивності для забезпечення ефективного виконання коду.

Гнучкість і налаштування: стек LAMP надає розробникам можливість налаштовувати та адаптувати систему керування завданнями відповідно до конкретних вимог. Відкритий вихідний код компонентів дозволяє легко модифікувати та інтегрувати з іншими інструментами та технологіями.

Економічна ефективність: стек LAMP є економічно ефективним рішенням для веб-розробки, оскільки всі компоненти є відкритими та доступними у вільному доступі. Це зменшує витрати на ліцензування та дозволяє здійснювати бюджетну розробку та розгортання.

Підтримка спільноти: стек LAMP має велику та активну спільноту розробників, що полегшує пошук ресурсів, навчальних посібників і допомоги, коли виникнуть проблеми під час розробки. Наявність керованих спільнотою плагінів, розширень і бібліотек покращує процес розробки.

Враховуючи ці фактори, стек LAMP із Linux, Apache, PHP, MySQL і Yii2 є підходящим вибором для розробки веб-системи керування завданнями для співробітників. Він пропонує стабільність, безпеку, масштабованість, продуктивність, гнучкість і економічну ефективність, а також багатий набір інструментів і ресурсів для ефективної розробки та обслуговування.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

Висновки до розділу 1

В першому розділі обґрунтовано доцільність використання програмних засобів для управління завданнями для працівників організації. Поставлено задачу щодо створення веб-орієнтованого програмного застосунку для ефективного керування завданнями. Сформульовано мету роботи, предмет та об'єкт дослідження. Визначено опис типових задач, що ставляться перед розробниками таких систем, і сформульовано основну задачу дослідження.

Здійснено огляд і порівняння поширених сучасних систем керування завданнями, виділено і згруповано ряд типових функцій системи, які спрямовані на підвищення продуктивності, співпраці та організації завдань.

Наведено та обґрунтовано перелік інструментів розробки та необхідного програмного забезпечення для реалізації програмного продукту.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЕКТУВАННЯ І РОЗРОБКА СИСТЕМИ КЕРУВАННЯ ЗАВДАННЯМИ ДЛЯ ПРАЦІВНИКІВ В ОРГАНІЗАЦІЇ

2.1. Базові функції та варіанти використання системи

Передбачається, що дана система володітиме трьома модулями:

- Модуль працівника (employee);
- Модуль менеджера (manager);
- Модуль адміністратора (administrator).

Модуль працівника являє собою інформаційну панель для користувачів з роллю *employee*:

- перегляд власних завдань;
- групування і пошук завдань по статусу та пріоритетності;
- оновлення власного профілю.

Модуль менеджера являє собою інформаційну панель для користувачів з роллю *manager*:

- створення завдання;
- перегляд власноруч створених завдань;
- оновлення власноруч створеного завдання;
- зміна статусу власноруч створеного завдання на “виконано”.

Модуль адміністратора є інформаційною панеллю для користувачів з роллю *admin*:

- керування структурою організації;
- керування користувачами/працівниками;
- призначення ролей;
- керування довідковими даними: статуси завдань, імена пріоритетів завдань тощо.

Побудуємо UML-діаграми використання, діяльностей, послідовностей та діаграму класів системи.

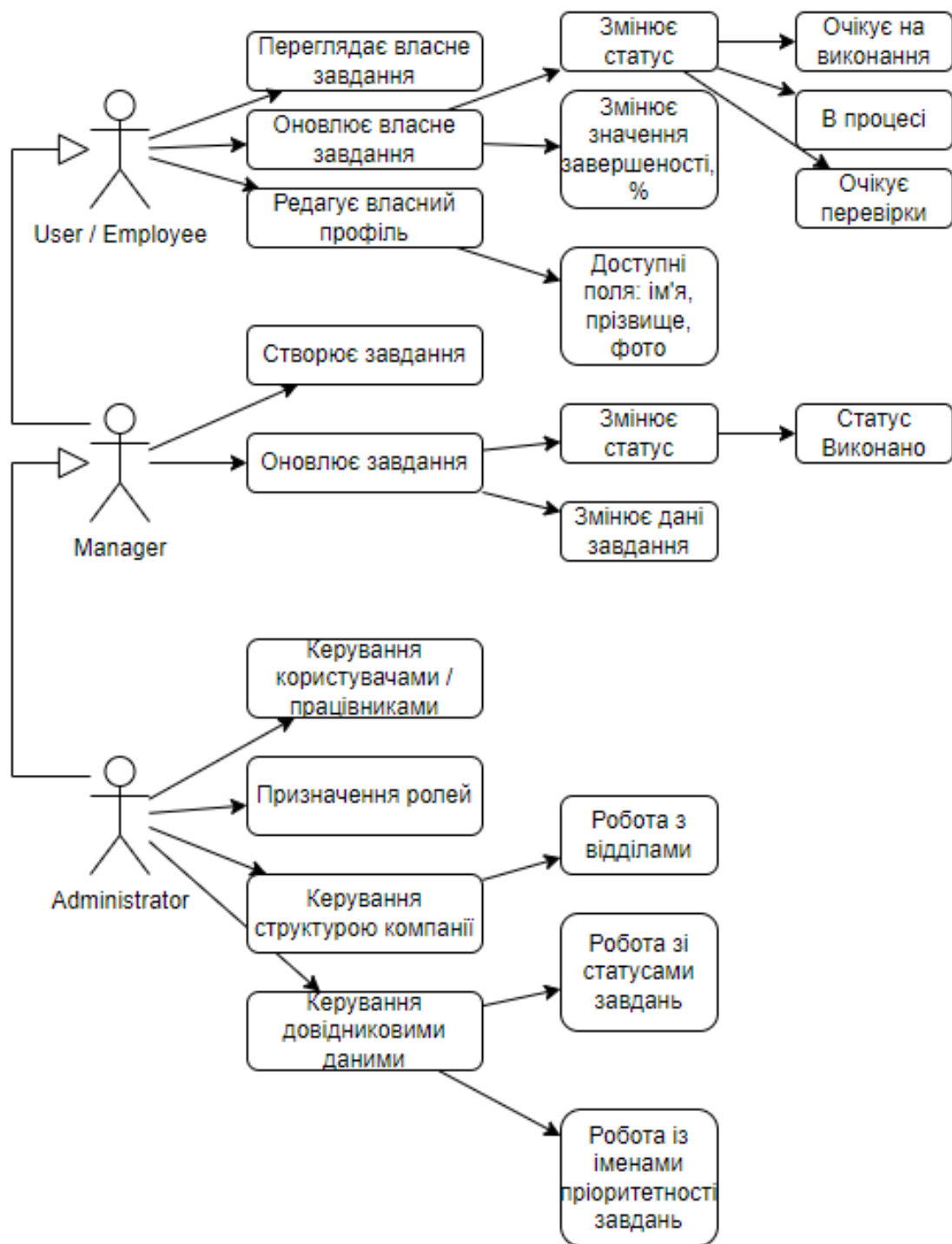


Рисунок 2.1. Діаграма варіантів використання системи керування завданнями для працівників в організації

На наведеній вище діаграмі варіантів використання:

- *Користувач/Працівник* (User/Employee) – головний актор (дійова особа), який представляє співробітників, які взаємодіють із системою керування завданнями. *Працівник* може виконувати такі дії, як перегляд деталей власного завдання, оновлення статусу

завдання з доступними значеннями: “очікує на виконання”, “в процесі”, “очікує перевірки”. Також *працівник* може змінювати значення завершеності власного завдання (у %) та доступні поля власного профілю: ім’я, прізвище, фото.

- *Менеджер* – актор, який є в т.ч. *працівником*, і додатково може виконувати такі дії, як створення завдання, перегляд деталей завдання, оновлення інформації про завдання, позначення завдань як виконаних і призначення завдань іншим співробітникам.
- «Адміністратор» – актор, який в т.ч. є *менеджером* і який має адміністративні привілеї. Адміністратори можуть керувати користувачами, створюючи, редагуючи або видаляючи облікові записи користувачів, призначати ролі користувачам, керувати довідниковими даними.

Зауважимо, що дана діаграма – спрощене представлення варіантів використання і ми при потребі можемо розширити її далі на основі конкретних вимог і додаткових функцій системи керування завданнями.

Відношення ролей:

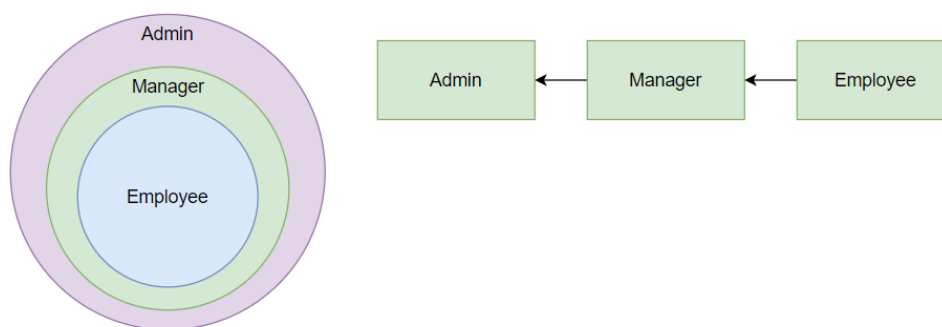


Рисунок 2.2. Відношення ролей

2.2. UML-діаграми діяльностей, послідовностей, класів

Здійснимо проектування діаграми активності системи.

Діаграма активності починається з дії «Запуск програми», яка представляє початок роботи системи керування завданнями.

Далі йде дія «Вхід/автентифікація користувача», яка вказує на процес автентифікації користувача для доступу до системи. Якщо користувач не зареєстрований в системі, він може зареєструватись.

Увійшовши в потрібний модуль системи, користувач може здійснювати власний набір функцій, до яких він має доступ.

В даному випадку діаграма ілюструє логіку створення і виконання завдання. Послідовність дій при цьому наступна:

- менеджер створює завдання та призначає його виконання деякому працівнику;
- працівник отримує завдання, виконує його повністю або частково, оновлюючи його статус (“в ході виконання”, “виконано, потребує перевірки”, “довиконано, потребує перевірки”);
- менеджер, отримавши дані по виконаним завданням зі статусом “потребує перевірки”, здійснює перевірку і підтверджує виконання або відправляє на доопрацювання;
- працівник доопрацьовує при необхідності і змінює статус на “довиконано, потребує перевірки”;
- після роботи із системою, користувач може здійснити вихід;

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						26
Змн.	Арк.	№ докум.	Підпис	Дата		

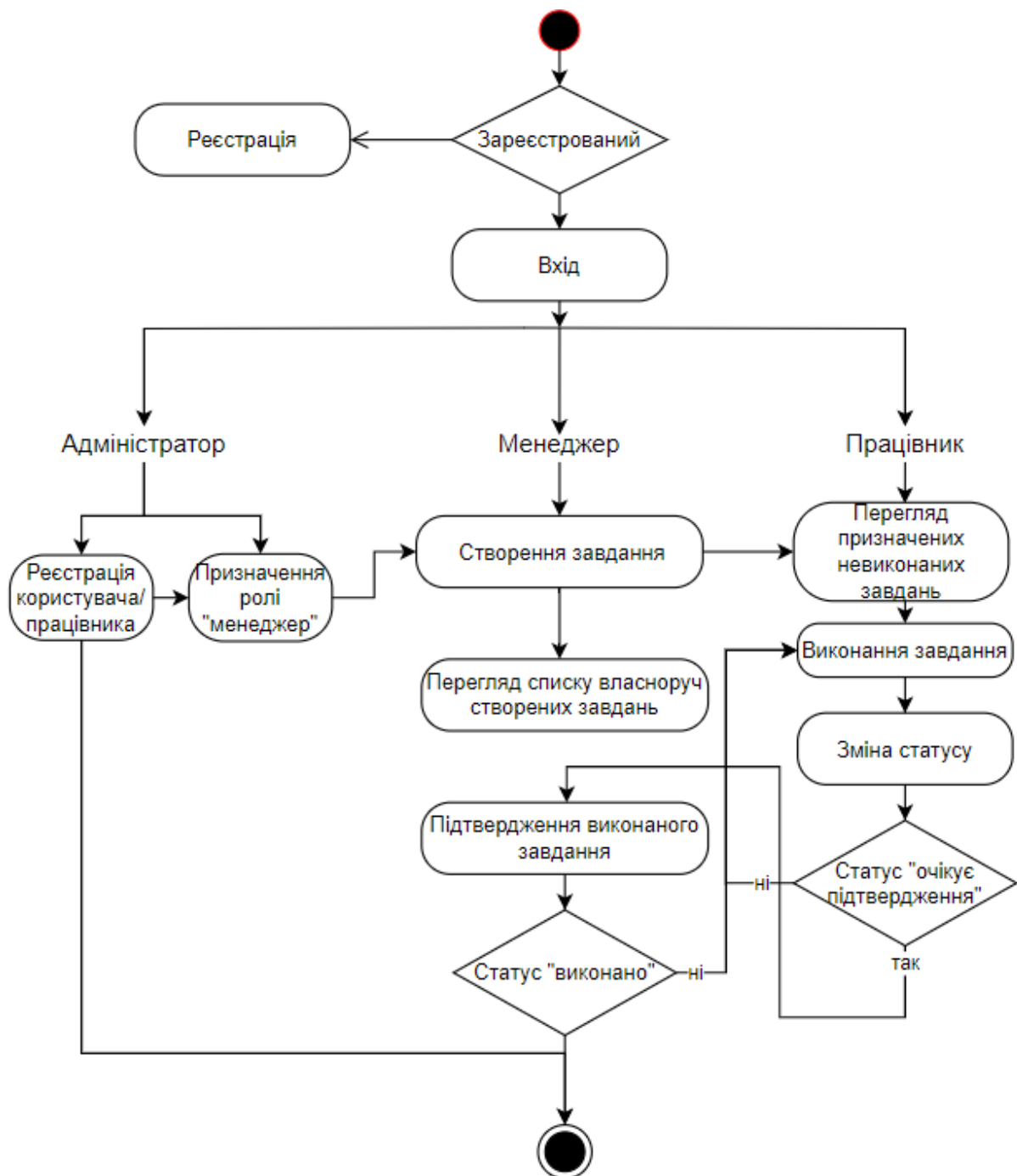


Рисунок 2.3. Діаграма діяльностей системи керування завданнями
Спроектуюмо наступну UML-діаграму – діаграму послідовностей.

- 1) Діаграма послідовності починається з надсилання менеджером запиту на створення нового завдання до системи керування завданнями.
- 2) Система отримує запит і витягує список завдань із бази даних.
- 3) Система повертає список завдань менеджеру.

- 4) Менеджер призначає завдання працівнику та надсилає запит на призначення в систему, включаючи деталі завдання та інформацію про працівника.
- 5) Співробітник отримує список завдань із системи.
- 6) Співробітник вибирає призначене завдання та запитує деталі завдання.
- 7) Система надсилає деталі завдання співробітнику.
- 8) Співробітник починає працювати над завданням і періодично оновлює статус завдання.
- 9) Співробітник виконує завдання та оновлює статус завдання як «виконано, потребує перевірки».
- 10) Система отримує оновлення про виконання завдання та позначає завдання як виконане в базі даних.
- 11) Про виконання завдання система сповіщає менеджера.
- 12) Менеджер отримує сповіщення та отримує доступ до системи керування завданнями.
- 13) Менеджер отримує список завдань і перевіряє статус виконання завдання.
- 14) Менеджер перевіряє деталі виконаного завдання.
- 15) При необхідності менеджер може дати відгук або закрити завдання.
- 16) Послідовність закінчується, представляючи завершення процесу керування завданнями.

Таким чином, діаграма послідовності демонструє взаємодію між менеджером та співробітником і системою керування завданнями, ілюструючи потік дій та обмін інформацією протягом усього процесу керування завданнями.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						28
Змн.	Арк.	№ докум.	Підпис	Дата		

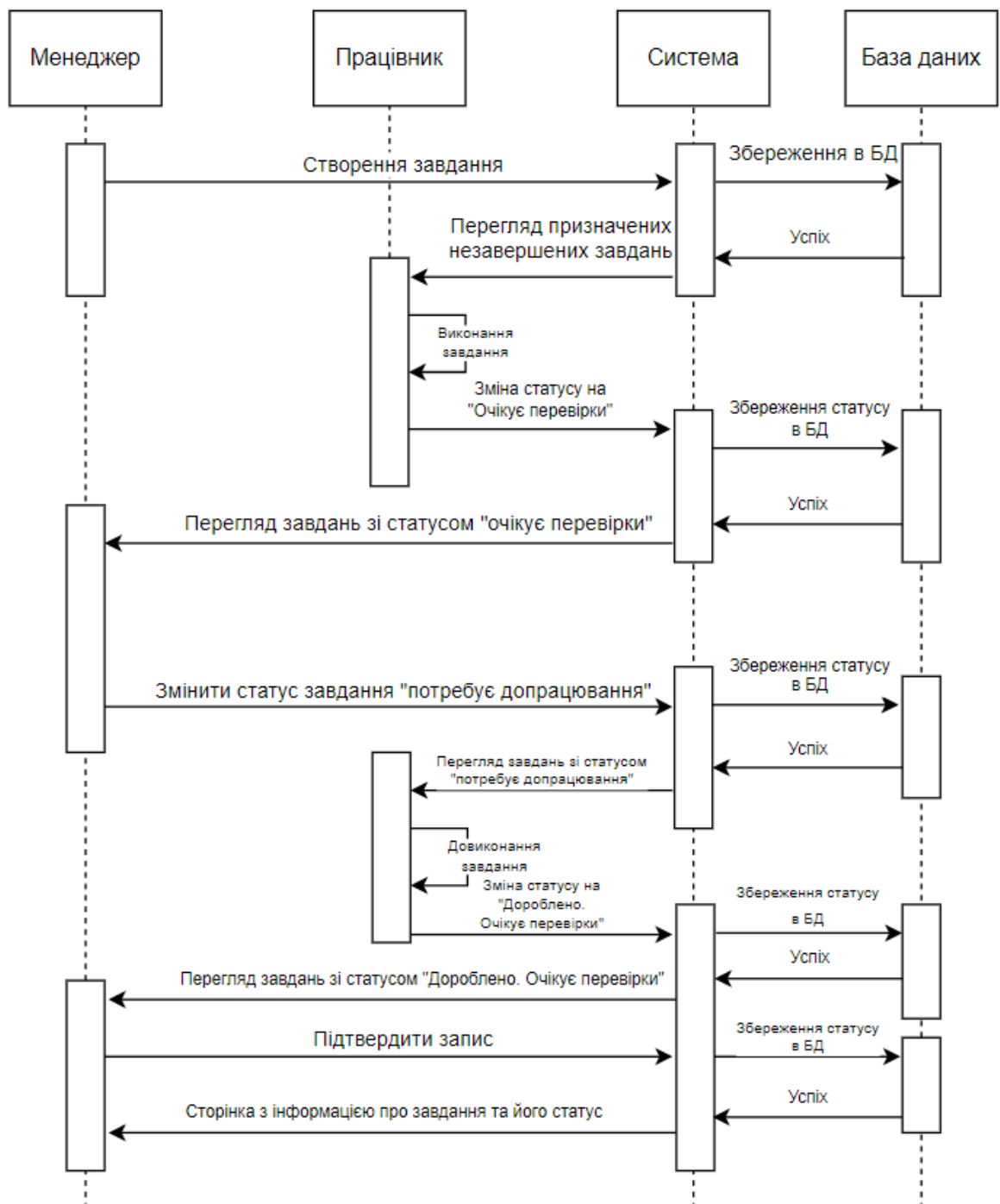


Рисунок 2.4. Діаграма послідовностей для ілюстрації роботи менеджера, працівника і системи по створенню і обробці завдання.

Здійснимо проектування діаграми класів – статичної структури даних, що відображає сутності та їх відношення: агрегації, асоціації, композиції

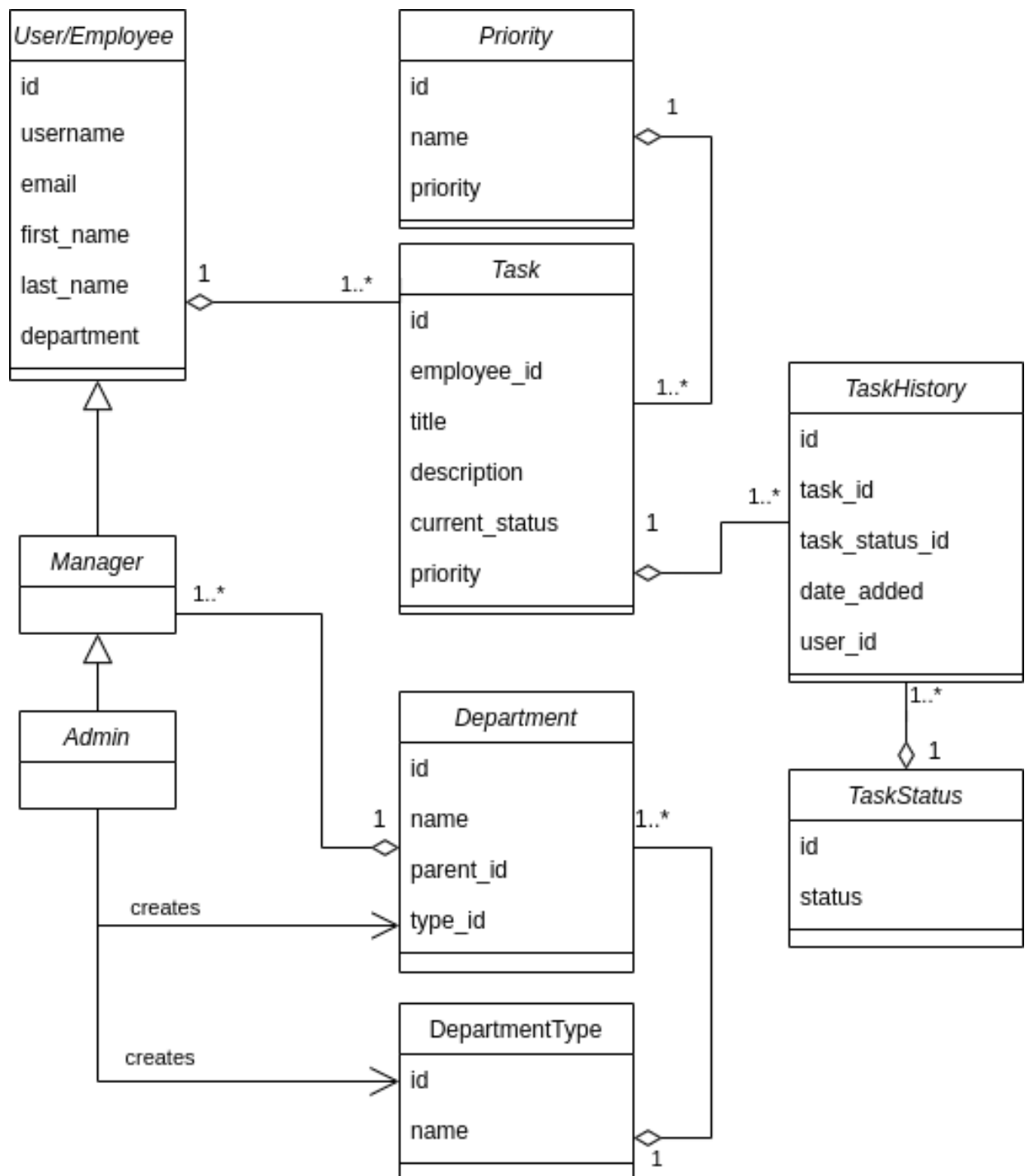


Рисунок 2.5. Діаграма класів системи керування задачами для працівників організації.

В даній діаграмі присутні такі користувачі та сутності:

User/Employee – клас роботи з базовим функціоналом користувача, таким як аутентифікація, вхід, реєстрація, тощо, а також для реалізації функціоналу працівника в межах свої дозволів: перегляд власних завдань, робота з профілем, формування та перегляд історії поточного завдання.

Manager – користувач, що має функції по створенню завдань та

призначенню їх виконання працівникам відділу, керівником якого він є. Якщо користувач має роль *Manager*, це означає, що він є керівником відділу в якому він працює. Даній ролі також належить роль *Employee*.

Admin – роль користувача, що має функції по створенню і редагуванню відділів організації, реєстрації користувачів, редагуванню їх профілів як працівників, а також по роботі з довідниковими даними: довідник статусів, довідник назв пріоритетності завдань тощо. Даній ролі також належить роль *Manager*.

Department – відділи як структурні одиниці організації.

DepartmentType – типи відділів.

Priority – назви пріоритетів завдання: критичне, надважливе, середній пріоритет, низький пріоритет, термінове, рутинне.

Task – завдання. Призначається певному працівнику, має поточний статус та історію статусів, має пріоритет.

TaskHistory – історія статусів завдання.

TaskStatus – назви статусів завдання: “очікує виконання”, “в процесі”, “очікує перевірки”, “відправлено на доопрацювання”, “доопрацьовано”, “виконано”.

2.3. Структура бази даних системи

Розробимо схему реляційної бази даних. В першій версії розробки система працюватиме з 11 таблицями даних:

Таблиця 2.1.

Таблиці реляційної бази даних системи

Таблиця	Сутність
user	Користувачі / працівники
task	Завдання
task_history	Історія статусів завдань
task_status	Довідник назв статусів завдань

task_priority	Довідник назв пріоритетів для завдань
department	Структурні одиниці організації (відділи)
department_type	Типи структурних одиниць (типи відділів)
auth_item	Ролі та дозволи
auth_assignment	Призначення ролей та дозволів користувачам
auth_item_child	Відношення ролей та дозволів
auth_rule	Правила в системі ролей та дозволів

Схему бази даних з ілюстраціями таблиць, полів, типів даних та зв'язків наведемо на рисунку 2.6:

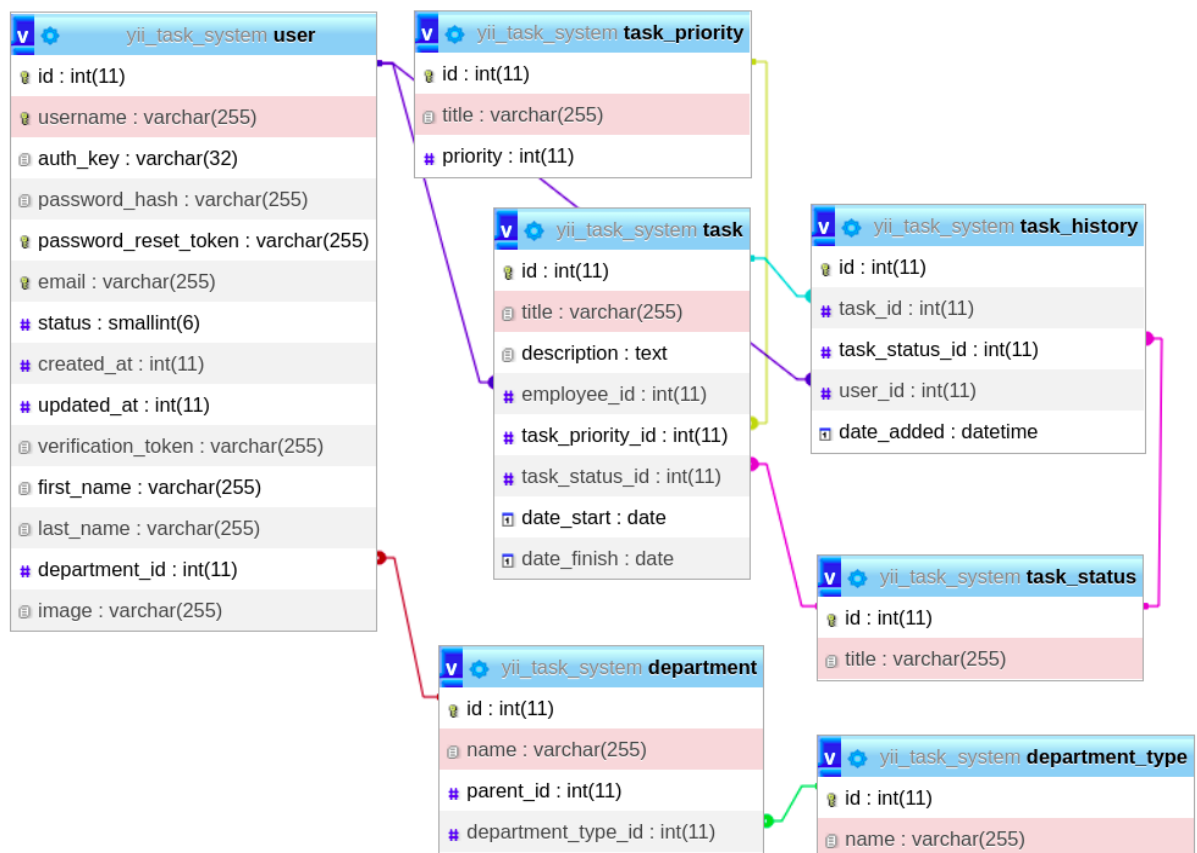


Рисунок 2.6. Схема бази даних системи керування завданнями для працівників

Наведемо перелік всіх таблиць системи керування завданнями з назвами та описом полів, типами даних, індексами.

Таблиця 2.2

Таблиця *user* – дані про працівників (zareєстрованих користувачів)

Поле	Тип	Індекс	Пояснення
id	int(11)	Первинний	Ідентифікатор працівника
username	varchar(255)	Унікальний	Логін
email	varchar(255)	Унікальний	Email
first_name	varchar(255)		Ім'я
last_name	varchar(255)		Прізвище
department_id	int(11)	Індекс	Відділ
image	varchar(255)		Зображення (фото)

Таблиця 2.3

Таблиця *task* – завдання для працівника

Поле	Тип	Індекс	Пояснення
id	int(11)	Первинний	Ідентифікатор завдання
title	varchar(255)		Заголовок завдання
description	text		Опис завдання
employee_id	int(11)	Індекс	Ідентифікатор працівника
task_priority_id	int(11)	Індекс	Пріоритет
task_status_id	int(11)	Індекс	Поточний статус
date_start	date		Дата початку завдання
date_finish	date		Дата завершення завдання

Таблиця 2.4

Таблиця *task_priority* – довідник імен пріоритетів завдань

Поле	Тип	Індекс	Пояснення
id	int(11)	Первинний	Ідентифікатор пріоритету
title	varchar(255)		Назва пріоритету
priority	int(11)		Числове значення пріоритету для сортування

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						33
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 2.5

Таблиця *department* – структурні одиниці організації, відділи

Поле	Тип	Індекс	Пояснення
id	int(11)	Первинний	Ідентифікатор відділу
name	varchar(255)		Назва
parent_id	int(11)	Індекс	Підпорядкування
department_type_id	int(11)	Індекс	Тип відділу

Таблиця 2.6

Таблиця *department_type* – довідник типів відділів

Поле	Тип	Індекс	Пояснення
id	int(11)	Первинний	Ідентифікатор
name	varchar(255)		Назва типу

Таблиця 2.7

Таблиця *task_status* – довідник з даними про статуси завдань.

Поле	Тип	Індекс	Пояснення
id	int(11)	Первинний	Ідентифікатор
title	varchar(255)		Статус

Таблиця 2.8

Таблиця *task_history* – ведення історії статусів виконання завдань

Поле	Тип	Індекс	Пояснення
id	int(11)	Первинний	Ідентифікатор статусу
task_id	int(11)	Індекс	Ідентифікатор завдання
task_status_id	int(11)	Індекс	Статус
user_id	int(11)	Індекс	Користувач, що додав статус
date_added	datetime		Дата і час додавання статусу

2.4. Написання програмного коду модуля системи

В даному розділі продемонструємо деякі практичні кроки по написанню програмного коду системи.

Етап 1. На першому етапі необхідно встановити модуль AdminLTE як основний веб-інтерфейс системи. Даний модуль матиме розділені блоки з функціями для користувачів з ролями Адміністратор, Менеджер та Працівник. Для встановлення модуля AdminLTE в файлі `composer.json` в розділі *require* пропишемо:

```
"hail812/yii2-adminlte3": "~1.1"
```

Після чого виконаємо команду в корені проекту для оновлення залежностей:

```
composer update
```

Після встановлення модуля AdminLTE, можна перенести всі шаблони з папки `vendor/hail812/yii2-adminlte3/src/views` в папку `views`:

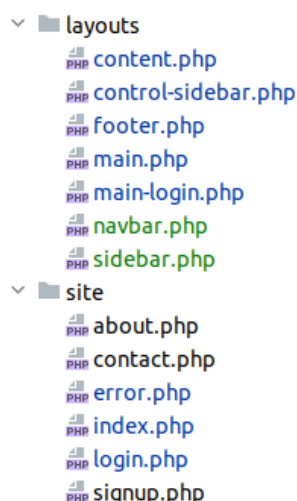


Рисунок 3.1.Шаблони модуля AdminLTE

Після чого веб-інтерфейс системи зміниться на AdminLTE:

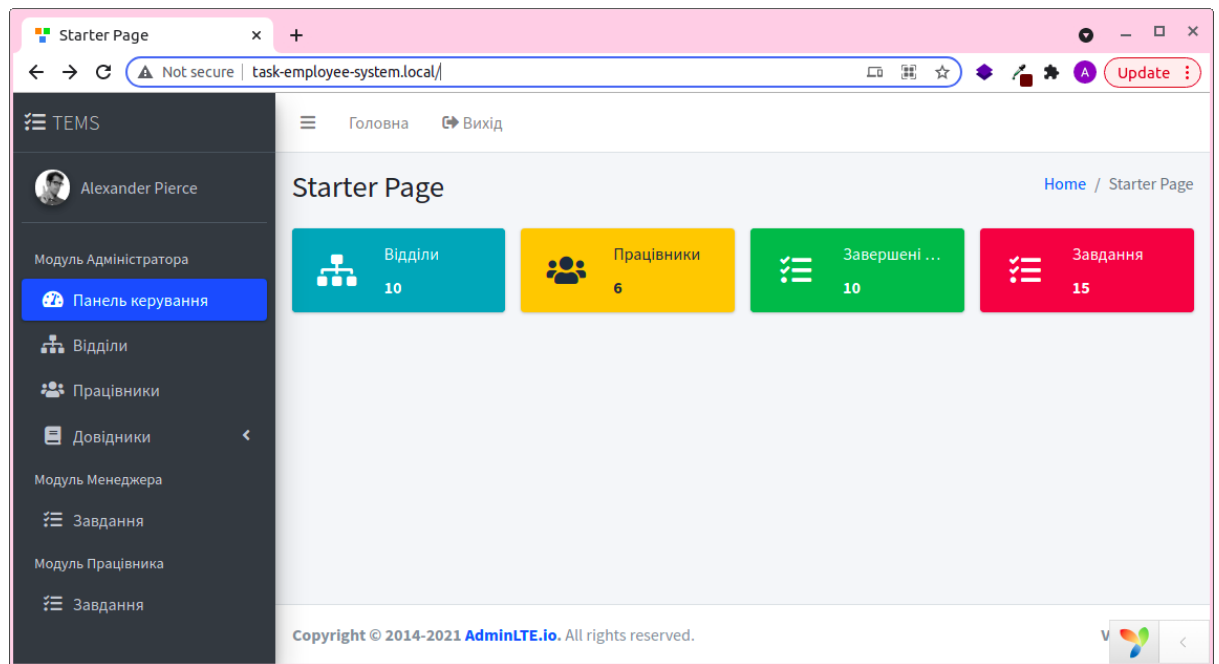


Рисунок 3.2. Вигляд веб-інтерфейсу системи після встановлення модуля AdminLTE.

Етап 2. На другому етапі здійснимо розробку системи ролей. В даній системі будуть три підпорядковані ролі: Employee, Manager та Admin. Скористаємось механізмом для створення ролей, що надає фреймворк Yii2.

- 1) Сконфігуруємо компонент `authManager`, щоб дані зберігались в БД. Для цього в файли конфігурації `config/console.php` та `config/web.php` в масив `components` додамо компонент `authManager` з посиланням на клас `yii\rbac\DbManager`:

```
'authManager' => [
    'class' => 'yii\rbac\DbManager',
    // uncomment if you want to cache RBAC items hierarchy
    // 'cache' => 'cache',
],
```

- 2) Виконаємо міграцію для створення таблиць системи авторизації в БД:

```
yii migrate --migrationPath=@yii/rbac/migrations
```

Після чого в БД створяться 4 таблиці:

- `auth_item` – таблиця для зберігання елементів авторизації;
- `auth_item_child` – таблиця для зберігання ієрархії елементів авторизації;
- `auth_assignment` – таблиця для зберігання призначень елементів авторизації;

- *auth_rule* – таблиця для зберігання правил.

3) Створимо клас *app\commands\RbacController* і в ньому реалізуємо метод дії *actionInit()*:

```
<?php
namespace app\commands;

use Yii;
use yii\console\Controller;

class RbacController extends Controller
{
    public function actionInit()
    {

    }
}
```

Даний контролер являє собою консольні команди, які можна виконувати через термінал. Кожен метод дії (action) являє собою окрему команду.

5) Реалізуємо метод дії *actionInit()*:

```
public function actionInit()
{
    $auth = Yii::$app->authManager;
    $auth->removeAll();

    $employee = $auth->createRole('Employee');
    $manager = $auth->createRole('Manager');
    $admin = $auth->createRole('Admin');

    $auth->add($employee);
    $auth->add($manager);
    $auth->add($admin);

    $auth->addChild($manager, $employee);
    $auth->addChild($admin, $manager);

    $auth->assign($admin, 1);
    $auth->assign($manager, 2);
    $auth->assign($employee, 3);
    $auth->assign($employee, 4);
    $auth->assign($manager, 5);
}
```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						37
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    $auth->assign($employee, 6);
}

```

В даному методі, ми отримуємо `$auth` – змінна-делегат для роботи із системою авторизації. Після чого робимо повну очистку таблиць авторизації. Далі створюємо (`createRole`) і зберігаємо (`add`) в БД три ролі: `Employee`, `Manager`, `Admin`. Метод `addChild()` задає інформацію про відношення ролей та дозволів. В даному випадку, ми підпорядковуємо роль `Manager` ролі `Admin`, а роль `Employee` – ролі `Manager`.

Наявним вже зареєстрованим користувачам потрібно призначити дані ролі. Для цього викличемо метод `assign()` з параметрами ролі та ідентифікатора користувача.

б) виконаємо метод дії як консольну команду:

```
yii yii rbac/init
```

Перевіримо дані в БД:

- в таблиці *AuthItem* повинні бути три створені ролі (тип 1):

name	type	description	rule_name	data	created_at	updated_at
Admin	1	NULL	NULL	NULL	1686753643	1686753643
Employee	1	NULL	NULL	NULL	1686753643	1686753643
Manager	1	NULL	NULL	NULL	1686753643	1686753643

Рисунок 3.3. Дані таблиці для зберігання ролей

- в таблиці *AuthItemChild* повинні бути дані про відношення ролей:

parent	child
Admin	Manager
Manager	Employee

Рисунок 3.4. Дані таблиці для зберігання відношень між ролями

- в таблиці *AuthAssignment* повинні бути дані про призначення ролей заданим користувачам по ID:

item_name	user_id	1	created_at
Admin	1	1686753643	
Manager	2	1686753643	
Employee	3	1686753643	
Employee	4	1686753643	
Manager	5	1686753643	
Employee	6	1686753643	

Рисунок 3.5. Дані таблиці для призначення ролей відповідним користувачам

7) При реєстрації (створенні) працівників наша система повинна автоматично призначати їм роль Employee. Цей крок можна реалізувати з допомогою “магічного” методу-хуку *afterSave()* моделі *User*:

```
public function afterSave($insert, $changedAttributes)
{
    if ($insert) {
        $auth = Yii::$app->authManager;
        $employee = $auth->getRole("Employee");
        $auth->assign($employee, $this->getId());
    }

    parent::afterSave($insert, $changedAttributes);
}
```

Даний хук спрацює автоматично одразу після додавання нового запису в таблицю *users*. В ньому перевіряється параметр *\$insert*, який має значення *true* у випадку саме додавання нового запису (не оновлення). Якщо *\$insert* має значення *true*, отримуємо компонент *authManager* (змінна-делегат *\$auth*), отримуємо роль *Employee* та призначаємо її даному щойно зареєстрованому користувачеві за його ID.

Для здійснення перевірки, необхідно зареєструвати нового користувача та перевірити в таблиці *auth_assignment*, чи додався новий запис з призначенням ролі *Employee* користувачу за його ID:

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						39
Змн.	Арк.	№ докум.	Підпис	Дата		

Employee	7
Employee	8

Рисунок 3.6. Результат перевірки автоматичного призначення ролі Employee щойно зареєстрованим користувачам

Висновки до розділу 2

При проектуванні системи було сформульовано функціональні можливості системи, визначено основні актори, що володіють доступом до даних функцій. Побудовано UML діаграми: варіантів використання, активностей, послідовностей та класів. Наведено структуру реляційної бази даних, що складається з 11 таблиць, 7 з яких використовуються для роботи з завданнями, структурою організації, та користувачами (працівниками), а також довідниками. 4 таблиці використовуються для обробки системи авторизації: створення ролей, дозволів, їх підпорядкування, призначення користувачам.

Наведено опис етапів розробки деяких компонентів системи, а саме особливості встановлення і використання модуля веб-інтерфейсу AdminLTE, а також механізм створення, збереження та призначення ролей зареєстрованим користувачам як працівникам організації.

РОЗДІЛ 3. ОПИС ВЕБ-ІНТЕРФЕЙСУ СИСТЕМИ. ІНСТРУКЦІЯ КОРИСТУВАННЯ І ВСТАНОВЛЕННЯ. ТЕСТУВАННЯ СИСТЕМИ КЕРУВАННЯ ЗАВДАННЯМИ

3.1. Керівництво встановлення системи

Для розгортання системи локально потрібно виконати ряд кроків:

Крок 1. Клонування репозиторію і створення проекту. Зайдемо в папку *projects*. Створимо в ній папку *task-employee-management-system*. Склонуємо в неї репозиторій.

```
cd projects
mkdir task-employee-management-system
cd task-employee-management-system
git clone https://gitlab.com/<username>/task-employee-system
```

Крок 2. Створимо локальне доменне ім'я. Для цього відкриємо файл *httpd-vhosts.conf* та додамо в конфігурацію нового віртуального хоста:

```
cd /opt/lampp/etc/extra
sudo gedit httpd-vhosts.conf

<VirtualHost *:80>
# Set document root to be "basic/web"
DocumentRoot "/opt/lampp/htdocs/task-employee-system/web"
<Directory "/opt/lampp/htdocs/task-employee-system/web">
    # use mod_rewrite for pretty URL support
    RewriteEngine on
    # if $showScriptName is false in UrlManager, do not allow
accessing URLs with script name
    RewriteRule ^index.php/ - [L,R=404]
    # If a directory or a file exists, use the request directly
    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteCond %{REQUEST_FILENAME} !-d
    # Otherwise forward the request to index.php
    RewriteRule . index.php
    # ...other settings...
</Directory>
</VirtualHost>
```

Після чого відредагуємо файл */etc/hosts*, додавши до нього ір-адресу і віртуальний хост:

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						41
Змн.	Арк.	№ докум.	Підпис	Дата		

```
cd /etc
sudo gedit hosts
127.0.1.1 task-employee-system.local
Запустимо веб-сервер Apache (Lampp).
cd /opt/lampp
sudo ./manager-linux-x64.run
```

Виконаємо запит `http://task-employee-system.local`. В браузері повинна з'явитись веб-сторінка системи:

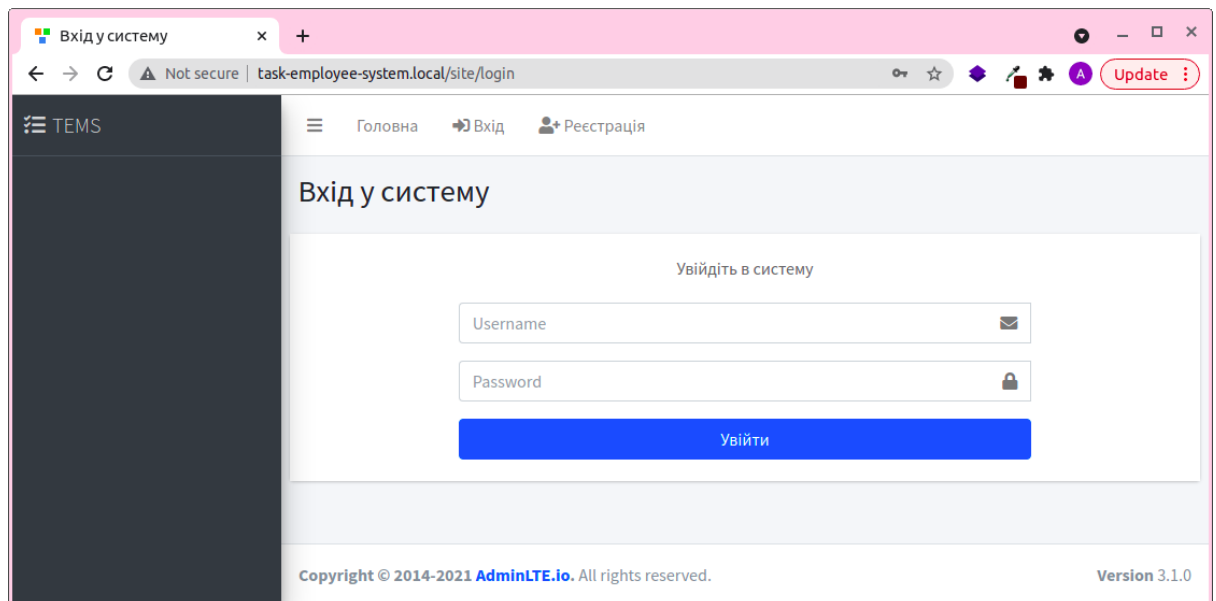


Рисунок 3.7. Сторінка для входу як підтвердження правильного встановлення системи

Крок 3. Створення, імпорт та підключення до БД. Зайдемо в PHPMyAdmin <http://localhost/phpmyadmin/>. Створимо нову базу даних `yii_task_system`. Здійснимо імпорт файлу `yii_task_system.sql`. Таким чином, ми створили БД та імпортували її структуру і демо-дані. Для підключення до БД, необхідно відкрити файл `config/db.php` і внести необхідні параметри підключення:

```
<?php
return [
    'class' => 'yii\db\Connection',
    'dsn' => 'mysql:host=localhost;dbname=yii_task_system',
    'username' => 'root',
    'password' => '',
    'charset' => 'utf8',
];
```

Змн.	Арк.	№ докум.	Підпис	Дата

3.2. Опис веб-інтерфейсу. Інструкція користування системою

Реєстрація (employee/signup). Для користування системою, в ній необхідно зареєструватись та увійти. На сторінці реєстрації користувач заповнює відповідну форму (логін, email, пароль) та відправляє запит на додавання нового користувача:

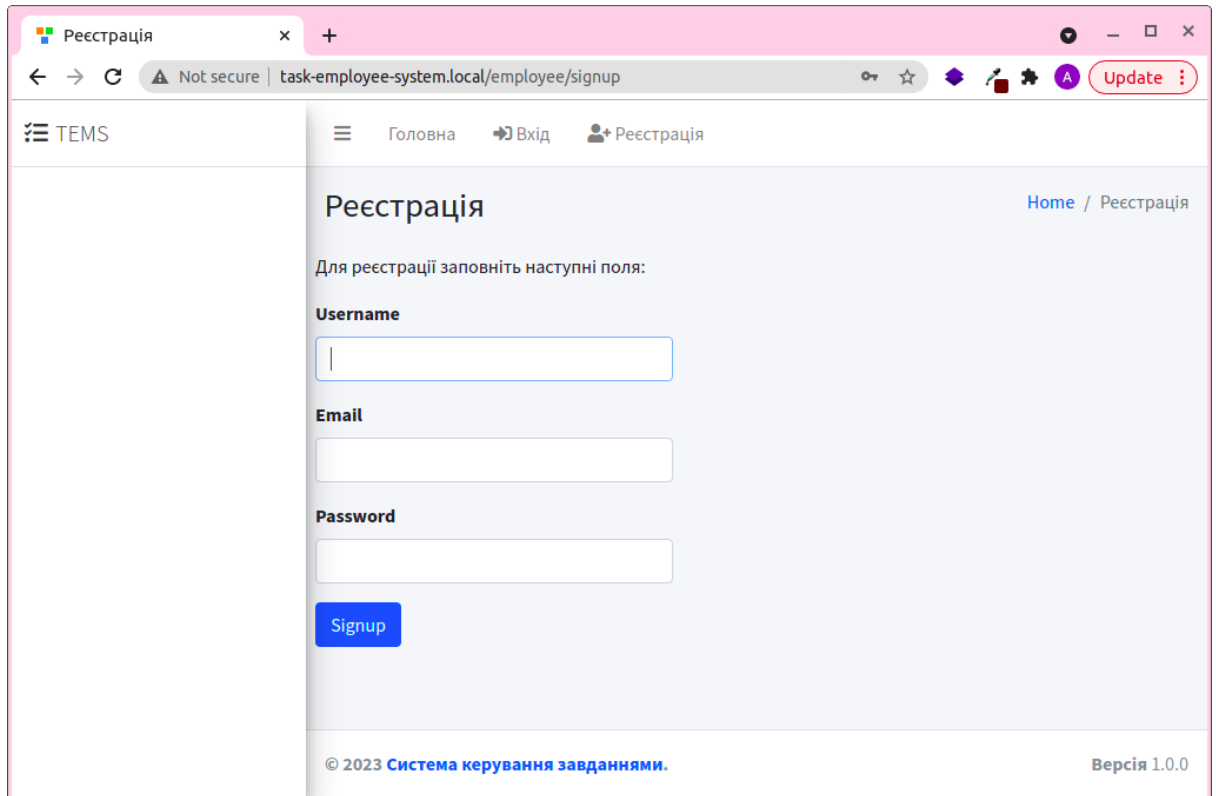


Рисунок 3.8. Реєстрація нового працівника

Вхід та вихід (site/login та /sitelogsout). Для користувача, що ще не здійснив вхід, на головній навігації будуть посилання з доступними діями: *Головна, Вхід, Реєстрація*

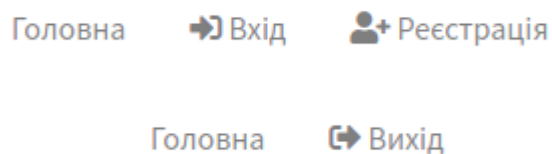


Рисунок 3.9. Верхня (горизонтальна) навігація користувача-гостя та автентифікованого користувача.

Бічне меню користувача, Sidebar. Дане меню містить посилання на веб-сторінки з різними функціями системи. Якщо користувачу призначена роль

Admin, йому доступні посилання для роботи з відділами, працівниками, довідниками (пріоритети завдань, статуси завдань). Якщо користувач має роль Manager, йому доступне посилання “Створені завдання” та “Нове завдання”, з допомогою яких він може створювати, переглядати статус, підтверджувати завдання або повертати на доопрацювання. Якщо користувач має роль Employee, йому доступні посилання: “Завдання для мене” – можливість переглядати, виконувати та змінювати статус призначених йому завдань, “Мій профіль” – можливість переглядати та змінювати власний профіль користувача:

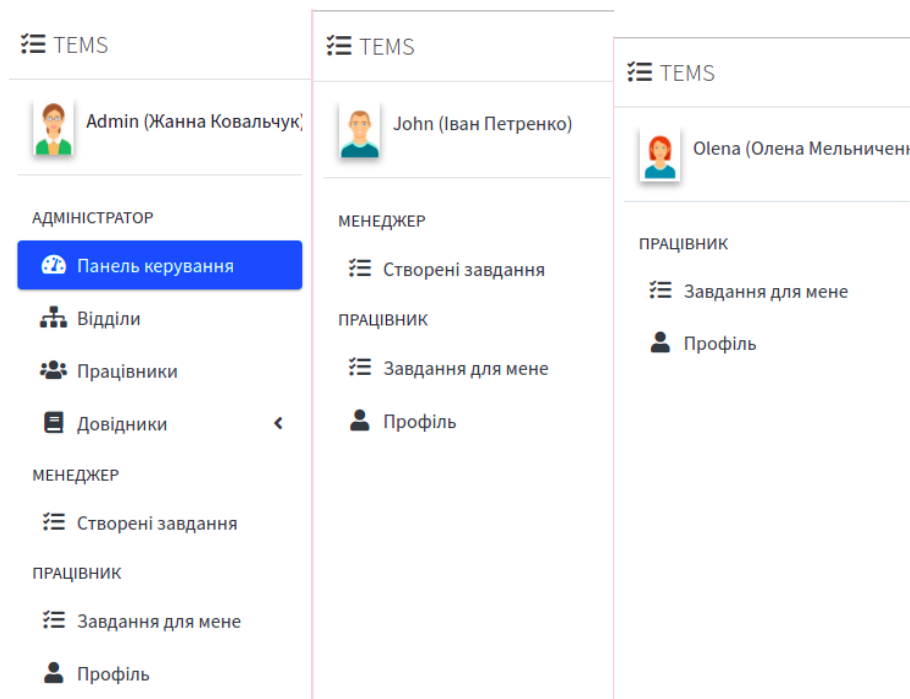


Рисунок 3.10. Бічне меню користувача (модулі “Адміністратор”, “Менеджер” та “Працівник”) та його вигляд для користувачів з відповідними ролями.

Створення завдання. Сторінка з формою для створення завдання доступна користувачам з роллю *Manager* (в т.ч. *Admin*). Користувачі мають ввести заголовок завдання, опис, вибрати працівника з власного відділу, задати пріоритет (звичайний, середній, важливий, надважливий), задати поточний статус (як правило, “очікує виконання”), дату початку і дату завершення.

TEMS

veronika (Вероніка Іщук)

МЕНЕДЖЕР

Керування завданнями

Нове завдання

ПРАЦІВНИК

Завдання для мене

Профіль

Головна

Вихід

Створення завдання

Home / Завдання / Створення завдання

Заголовок

Реалізувати систему ролей та дозволів

Опис

Реалізувати систему ролей та дозволів

Працівник

Michael

Пріоритет

Звичайний

Поточний статус

Очікує виконання

Дата початку

2023-6-1

Дата завершення

2023-6-30

Save

© 2023 Система керування завданнями.

Рисунок 3.11. Сторінка з формою для створення нового завдання. Доступна для користувачів з роллю *Manager*.

Керування завданнями. Доступ до даної сторінки надається користувачам з роллю *Manager*. На ній виводиться список завдань, створених даним користувачем з можливістю фільтрувати та сортувати всіх можливих полях, переглядати інформацію та історію статусів, редагувати, оновлювати статус та видаляти.

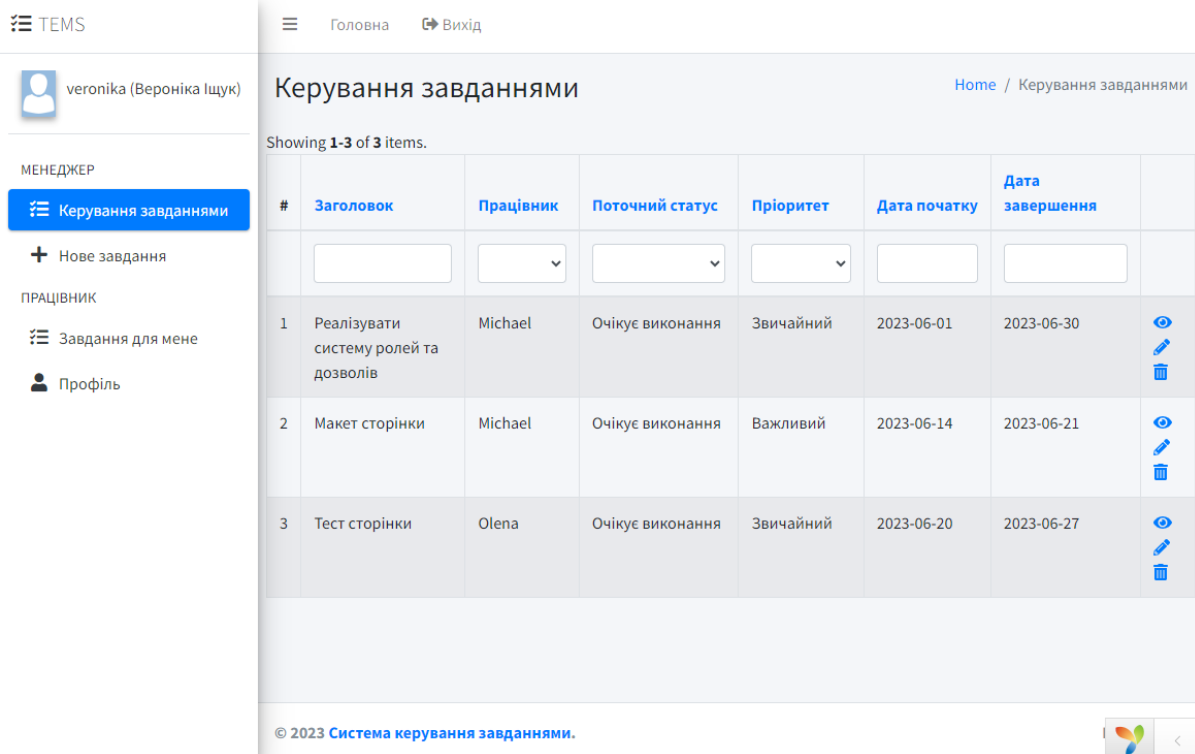


Рисунок 3.12. Сторінка керування завданнями для користувачів з роллю *Manager*.

Мої завдання. Доступна для перегляду користувачам з роллю *Employee*. Виводиться список завдань, призначених для виконання даним користувачем з можливістю пошуку та сортування по списку, переглядати інформацію та історію статусів, оновлювати статус.

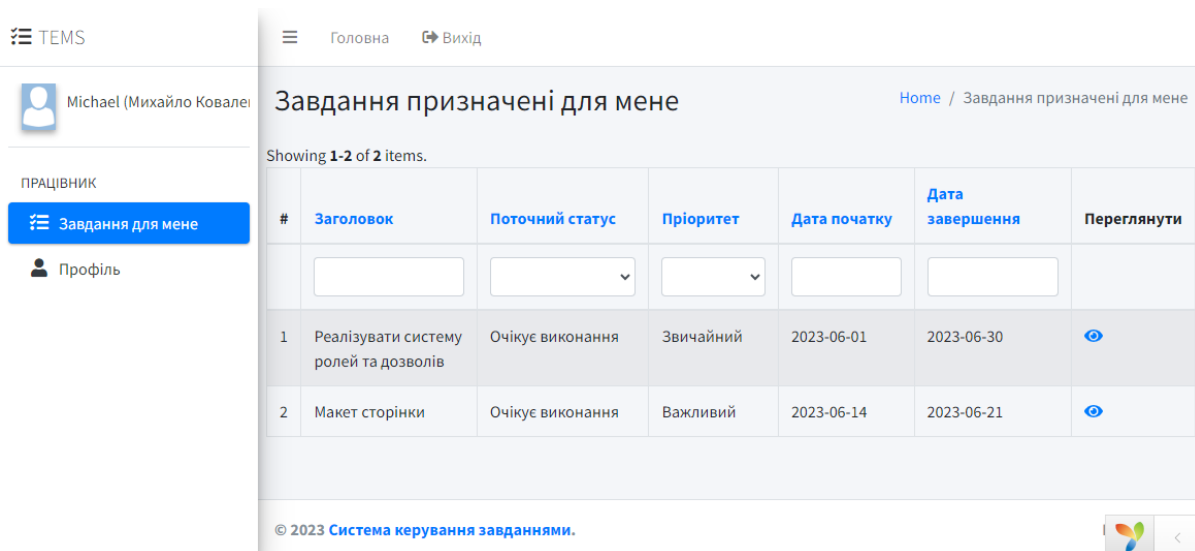


Рисунок 3.13. Сторінка “Мої завдання” для користувачів з роллю *Employee*.

Перегляд детальної інформації про завдання. Дана сторінка доступна для користувача з роллю *Manager*, який керує завданням, та користувача з роллю

Employee, який виконує завдання, а також для користувача з роллю Admin. Вона містить описові поля: заголовок, опис, виконавець, пріоритет, поточний статус, дати початку і завершення. На цій же сторінці виводиться історія статусів даного завдання, в якій зазначається: дата і час зміни статусу, назва статусу, хто змінив статус, примітка.

Завдання #4

[Home](#) / [Tasks](#) / [Завдання #4](#)

Оновити

Оновити статус

Видалити

Опис завдання

Заголовок	Макет сторінки
Опис	Зверстати макет сторінки
Відповідальний	Michael
Пріоритет	Звичайний
Поточний статус	Виконано, підтверджено
Дата початку	2023-06-01
Дата завершення	2023-06-08

Історія виконання завдання

Дата	Статус	Користувач	Примітка
2023-06-15 02:28:36	Очікує виконання	veronika	
2023-06-15 03:10:17	В процесі	Michael	Починаю виконувати 15.06 о 15:00
2023-06-15 03:36:37	Виконано, очікує перевірки	Michael	Чекаю на підтвердження
2023-06-15 03:37:08	Відправлено на доопрацювання	veronika	Є недоліки, потрібно доопрацювати
2023-06-15 03:37:26	Виконано, очікує перевірки	Michael	Недоліки виправлено
2023-06-15 03:38:08	Виконано, підтверджено	veronika	Завдання виконано якісно і вчасно. Дякую!

Рисунок 3.14. Перегляд детальної інформації про завдання

Сторінка з формою для зміни (додавання) статусу завдання. Доступна для менеджера як автора завдання та для працівника як виконавця завдання.

Менеджер має можливість змінювати всі можливі значення статусів. Працівник може змінювати лише статуси з такими значеннями: очікує виконання, в процесі, виконано і очікує підтвердження. Будь-яка зміна статусу записується в історію, в тому числі і повернення на попередні статуси.

Рисунок 3.15. Сторінка з формою для оновлення (додавання) статусу завдання

3.3. Тестування веб-системи для керування завданнями

Наведемо сценарій тестування системи з урахуванням розподілу функцій для створення і роботи з завданнями по ролях Manager та Employee:

Таблиця 3.1.

Сценарій тестування системи

Перевірка функцій адміністратора	
Здійснюємо запит на головну сторінку	Перенаправлення на сторінку для входу
Здійснюємо вхід під користувачем admin	Виводиться головна сторінка системи з меню для всіх модулів системи: Admin, Manager, Employee

Створюємо новий відділ	Перенаправлення на сторінку зі списком відділів з відображенням нового відділу
Реєструємо нового працівника	Перенаправлення на сторінку зі списком всіх працівників, перегляд профілю працівника
Редагуємо профіль та фото працівника	Перенаправлення на сторінку профілю працівника з переглядом оновлених даних
Робота з довідниковими даними: створення, оновлення та перегляд пріоритетів завдань та статусів завдань	Перенаправлення на сторінку списку довідника з оновленими даними
<i>Перевірка функцій менеджера</i>	
Створення нового завдання для працівника свого відділу	Перенаправлення на сторінку зі списком власноруч створених завдань
Спроба створити завдання для працівника не свого відділу	Помилка авторизації з відповідним повідомленням
Оновлення загальної інформації про завдання	Перенаправлення на сторінку з детальною інформацією про завдання
Оновлення (додавання) статусу завдання	Перенаправлення на сторінку з детальною інформацією про завдання і оновленою історією

<i>Перевірка функцій працівника</i>	
Перегляд призначених завдань	Вивід списку завдань, що належать даному користувачу як виконавцю
Оновлення статусу завдання в межах своїх дозволів	Повинні дозволятися всі значення, крім статусу “Підтверджено”
Спроба несанкціонованого доступу до оновлення чи видалення завдання	Помилка авторизації з відповідним повідомленням
Редагування власного профілю та власного зображення	Переадресація на сторінку перегляду профілю з оновленими даними
Спроба відредагувати профіль іншого користувача	Помилка авторизації з відповідним повідомленням

Крім функціонального тестування було проведено також:

- Тестування usability (зручності використання): результати тестування виявили потенційні проблеми зручності використання для внесення необхідних покращень.
- Тестування продуктивності: оцінка швидкості реагування, масштабованості і стабільності системи за різних навантажень і умов. Це гарантує, що система справлятиметься з очікуваними навантаженнями без зниження продуктивності.
- Тестування безпеки: виявлення та усунення потенційних вразливостей у системі керування завданнями.
- Тестування сумісності: тестування сумісності гарантує, що система керування завданнями бездоганно працює в різних браузерях, операційних системах і пристроях.

- Регресійне тестування: виконується після внесення змін або виправлення помилок, щоб переконатися, що зміни не створюють нових проблем і не порушують існуючі функції.
- Відстеження та звітування про помилки: протягом усього процесу тестування було відстежено будь-які виявлені дефекти чи проблеми за допомогою системи відстеження помилок.

Таким чином, варто зазначити, що конкретний процес тестування може відрізнятися залежно від організації, вимог проекту та наявних ресурсів. Гнучкі методології часто інтегрують тестування в цикли розробки, тоді як великі проекти можуть включати спеціальні команди тестування та більш широке тестування. Цей ітераційний процес триває, доки система керування завданнями не відповідатиме необхідним стандартам якості.

Висновки до розділу 3

Розроблено і наведено опис веб-інтерфейсу системи керування задачами і особливості щодо користування системою. Наведено кроки щодо розгортання та налаштування системи для середовища розробки.

Проведено комплексне тестування системи, до якого включено такі типи тестування: функціональне, зручності використання, продуктивності, тестування безпеки, сумісності, регресійне тестування. Процес тестування по суті є ітераційним процесом, що може відрізнятися залежно від організації, вимог проекту та наявних ресурсів, і триває, доки система керування завданнями не відповідатиме необхідним стандартам якості.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						51
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

В дані роботі обґрунтовано доцільність використання програмних засобів для управління завданнями для працівників організації. Поставлено задачу щодо створення веб-орієнтованого програмного застосунку для ефективного керування завданнями. Сформульовано мету роботи, предмет та об'єкт дослідження. Визначено опис типових задач, що ставляться перед розробниками таких систем, і сформульовано основну задачу дослідження.

Здійснено огляд і порівняння поширених сучасних систем керування завданнями, виділено і згруповано ряд типових функцій системи, які спрямовані на підвищення продуктивності, співпраці та організації завдань.

Наведено та обґрунтовано перелік інструментів розробки та необхідного програмного забезпечення для реалізації програмного продукту.

При проектуванні системи було сформульовано функціональні можливості системи, визначено основні актори, що володіють доступом до даних функцій. Побудовано UML діаграми: варіантів використання, активностей, послідовностей та класів. Наведено структуру реляційної бази даних, що складається з 11 таблиць, 7 з яких використовуються для роботи з завданнями, структурою організації, та користувачами (працівниками), а також довідниками. 4 таблиці використовуються для обробки системи авторизації: створення ролей, дозволів, їх підпорядкування, призначення користувачам.

Наведено опис етапів розробки деяких компонентів системи, а саме особливості встановлення і використання модуля веб-інтерфейсу AdminLTE, а також механізм створення, збереження та призначення ролей зареєстрованим користувачам як працівникам організації.

Розроблено і наведено опис веб-інтерфейсу системи керування задачами і особливості щодо користування системою. Наведено кроки щодо розгортання та налаштування системи для середовища розробки.

Проведено комплексне тестування системи, до якого включено такі типи тестування: функціональне, зручності використання, продуктивності, тестування безпеки, сумісності, регресійне тестування. Процес тестування по суті є

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		

ітераційним процесом, що може відрізнятися залежно від організації, вимог проекту та наявних ресурсів, і триває, доки система керування завданнями не відповідатиме необхідним стандартам якості.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						53
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Gamma, E., Helm, R., Johnson, R., Vlissides, J. M. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional. ISBN: 0201633612
2. Документація мови програмування PHP [Електронний ресурс] - Режим доступу до ресурсу: <http://php.net>
3. Yii2 framework. Керівництво користувача [Електронний ресурс] - Режим доступу до ресурсу: <https://www.yiiframework.com>
4. Березін О.В. Управління проектами: навчальний посібник / О.В. Березін, М.Г. Безпарточний. – Суми: Університетська книга, 2014. – 272 с.
5. Schwaber, K. (2004). Agile Project Management with Scrum. Prentice Hall.
6. Shore, J., Warden, S. (2007). The Art of Agile Development. Beijing: O'Reilly. ISBN: 978-0-59652-767-9
7. Bass, L., Clements, P., Kazman, R. (2003). Software Architecture in Practice. Addison-Wesley. ISBN: 9780321154958
8. Benjamin Brandall. Get More Done The Complete Introduction to Task Management. Електронний ресурс. Режим доступу: <https://www.process.st/wp-content/uploads/2016/11/Get-More-Done-%E2%80%94-The-Complete-Guide-to-Task-Management.pdf>
9. Ong, E., Grebner, O. & Riss, U. V. (2007). Pattern-Based Task Management: Pattern Lifecycle and Knowledge Management - Benefits and open issues. In N. Gronau (ed.), 4th Conference on Professional Knowledge Management - Experiences and Visions -, March 28. - 30. 2007, Potsdam, Germany, 03, : GITO Verlag Berlin.
10. Takahara, Y. & Chen, X. (1995). Task Management System.. In F. Pichler, R. Moreno-Díaz & R. F. Albrecht (eds.), EUROCAST (p./pp. 48-61), : Springer. ISBN: 3-540-60748-X
11. Cohn, M. (2006). Agile estimating and planning. Upper Saddle River, NJ, Prentice Hall Professional Technical Reference.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дата		

- 12.Jyothi, N & A, Parkavi. (2016). A study on task management system. 1-6. 10.1109/RAINS.2016.7764421.
- 13.Bolooki, Faisal & Bridge, Adrian. (2023). The guide to the project management body of knowledge (PMBOK Guide) : applicability and contribution to project success in Australia.
- 14.Онищенко Е. Система управління кадровим потенціалом підприємства: підхід до проектування підсистем / Е. Онищенко // Вісник Тернопільського національного економічного університету. – 2012. – № 1. – С. 137-144.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						55
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						56
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг класу TaskController – контролер завдань

```

<?php

namespace app\controllers;

use app\models\Employee;
use app\models\Task;
use app\models\TaskHistory;
use app\models\TaskSearch;
use app\models\TaskStatusForm;
use Yii;
use yii\web\Controller;
use yii\web\ForbiddenHttpException;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;

/**
 * TaskController implements the CRUD actions for Task model.
 */
class TaskController extends Controller
{
    /**
     * @inheritdoc
     */
    public function behaviors()
    {
        return array_merge(
            parent::behaviors(),
            [
                'verbs' => [
                    'class' => VerbFilter::className(),
                    'actions' => [
                        'delete' => ['POST'],
                    ],
                ],
            ],
        );
    }

    /**
     * Lists all Task models.
     *
     * @return string
     */
    public function actionIndex()
    {
        if (!Yii::$app->user->can("Admin")) {
            throw new ForbiddenHttpException("Forbidden! You must have an Admin role");
        }
        $searchModel = new TaskSearch();
        $dataProvider = $searchModel->search($this->request->queryParams);

        return $this->render('index', [
            'searchModel' => $searchModel,
            'dataProvider' => $dataProvider,
        ]);
    }

    public function actionManageTodoList()

```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						57
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    {
        if (!Yii::$app->user->can("Manager")) {
            throw new ForbiddenHttpException("Forbidden! You must have an Manager
role");
        }
        $searchModel = new TaskSearch();
        $params = Yii::$app->request->queryParams;
        $params['TaskSearch']['manager_id'] = Yii::$app->user->getId();

        $dataProvider = $searchModel->search($params);

        $manager = Employee::findOne(Yii::$app->user->getId());

        return $this->render('manage-todo-list', [
            'searchModel' => $searchModel,
            'dataProvider' => $dataProvider,
            'manager' => $manager
        ]);
    }

    public function actionMyTodoList()
    {
        if (!Yii::$app->user->can("Employee")) {
            throw new ForbiddenHttpException("Forbidden! You must have an
Employee role");
        }
        $searchModel = new TaskSearch();
        $params = Yii::$app->request->queryParams;
        $params['TaskSearch']['employee_id'] = Yii::$app->user->getId();

        $dataProvider = $searchModel->search($params);

        return $this->render('my-todo-list', [
            'searchModel' => $searchModel,
            'dataProvider' => $dataProvider,
        ]);
    }

    /**
     * Displays a single Task model.
     * @param int $id ID
     * @return string
     * @throws NotFoundHttpException if the model cannot be found
     */
    public function actionView($id)
    {
        return $this->render('view', [
            'model' => $this->findModel($id),
        ]);
    }

    /**
     * Creates a new Task model.
     * If creation is successful, the browser will be redirected to the 'view'
page.
     * @return string|\yii\web\Response
     */
    public function actionCreate()
    {
        if (!Yii::$app->user->can("Manager")) {
            throw new ForbiddenHttpException("Forbidden! You must have a Manager
role");
        }
    }

```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        $model = new Task();
        $manager = Employee::findOne(Yii::$app->user->getId());
        if ($this->request->isPost) {
            if ($model->load($this->request->post()) && $model->save()) {
                return $this->redirect(['view', 'id' => $model->id]);
            }
        } else {
            $model->loadDefaultValues();
        }

        return $this->render('create', [
            'model' => $model,
            'manager' => $manager
        ]);
    }

    /**
     * Updates an existing Task model.
     * If update is successful, the browser will be redirected to the 'view'
page.
     * @param int $id ID
     * @return string|\yii\web\Response
     * @throws NotFoundHttpException if the model cannot be found
     */
    public function actionUpdate($id)
    {
        if (!Yii::$app->user->can("Manager")) {
            throw new ForbiddenHttpException("Forbidden! You must have a Manager
role");
        }
        $model = $this->findModel($id);
        $manager = Employee::findOne(Yii::$app->user->getId());
        if ($this->request->isPost && $model->load($this->request->post()) &&
$model->save()) {
            return $this->redirect(['view', 'id' => $model->id]);
        }

        return $this->render('update', [
            'model' => $model,
            'manager' => $manager
        ]);
    }

    public function actionUpdateStatus($id)
    {
        $model = $this->findModel($id);
        $taskStatusForm = new TaskStatusForm();
        if ($this->request->isPost && $taskStatusForm->load($this->request-
>post())) {
            if ($model->task_status_id != $taskStatusForm->task_status_id) {
                $model->task_status_id = $taskStatusForm->task_status_id;
                if ($model->save()) {
                    $taskHistory = new TaskHistory();
                    $taskHistory->task_id = $model->id;
                    $taskHistory->task_status_id = $model->task_status_id;
                    $taskHistory->user_id = Yii::$app->user->getId();
                    $taskHistory->date_added = date("Y-m-d h-i-s");
                    $taskHistory->note = $taskStatusForm->note;
                    $taskHistory->save();
                    return $this->redirect(['view', 'id' => $model->id]);
                } else {
                    Yii::$app->session->setFlash("danger", "Помилка при
збереженні статусу");
                }
            }
        }
    }

```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						59
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        } else {
            Yii::$app->session->setFlash("danger", "Статус збігається з
поточним");
        }

    }
    return $this->render('status', [
        'taskStatusForm' => $taskStatusForm,
        'model' => $model
    ]);
}

/**
 * Deletes an existing Task model.
 * If deletion is successful, the browser will be redirected to the 'index'
page.
 * @param int $id ID
 * @return \yii\web\Response
 * @throws NotFoundHttpException if the model cannot be found
 */
public function actionDelete($id)
{
    $this->findModel($id)->delete();

    return $this->redirect(['index']);
}

/**
 * Finds the Task model based on its primary key value.
 * If the model is not found, a 404 HTTP exception will be thrown.
 * @param int $id ID
 * @return Task the loaded model
 * @throws NotFoundHttpException if the model cannot be found
 */
protected function findModel($id)
{
    if (($model = Task::findOne(['id' => $id])) !== null) {
        return $model;
    }

    throw new NotFoundHttpException('The requested page does not exist.');
```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг шаблону веб-сторінки для перегляду детальної інформації про завдання

```

<?php

use yii\helpers\Html;
use yii\widgets\DetailView;

/** @var yii\web\View $this */
/** @var app\models\Task $model */

$this->title = "Завдання #" . $model->id;
$this->params['breadcrumbs'][] = ['label' => 'Tasks', 'url' => ['index']];
$this->params['breadcrumbs'][] = $this->title;
\yii\web\YiiAsset::register($this);
?>
<div class="task-view">

    <p>
        <?= Html::a('Оновити', ['update', 'id' => $model->id], ['class' => 'btn btn-primary']) ?>
        <?= Html::a('Оновити статус', ['update-status', 'id' => $model->id], ['class' => 'btn btn-primary']) ?>
        <?= Html::a('Видалити', ['delete', 'id' => $model->id], [
            'class' => 'btn btn-danger',
            'data' => [
                'confirm' => 'Are you sure you want to delete this item?',
                'method' => 'post',
            ],
        ]) ?>
    </p>
    <div class="row">
        <div class="col-12 col-md-6">
            <h4>Опис завдання</h4>
            <?= DetailView::widget([
                'model' => $model,
                'attributes' => [
                    'title',
                    'description:ntext',
                    'employee.username',
                    'taskPriority.title',
                    'taskStatus.title',
                    'date_start',
                    'date_finish',
                ],
            ]) ?>
        </div>
        <div class="col-12 col-md-6">
            <h4>Історія виконання завдання</h4>
            <table class="table table-bordered">
                <tr>
                    <th>Дата</th>
                    <th>Статус</th>
                    <th>Користувач</th>
                    <th>Примітка</th>
                </tr>
                <?php foreach ($model->taskHistories as $historyItem): ?>

```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						61
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        <tr>
            <td><?= $historyItem->date_added ?></td>
            <td><?= $historyItem->taskStatus->title ?></td>
            <td><?= $historyItem->user->username ?></td>
            <td><?= $historyItem->note ?></td>
        </tr>
    <?php endforeach; ?>
</table>

</div>
</div>

</div>

```

Лістинг моделі *Task* – класу для роботи з таблицею для зберігання даних про
завдання

```
<?php

namespace app\models;

use Yii;

/**
 * This is the model class for table "task".
 *
 * @property int $id
 * @property string $title
 * @property string $description
 * @property int $employee_id
 * @property int $task_priority_id
 * @property int $task_status_id
 * @property string $date_start
 * @property string $date_finish
 *
 * @property User $employee
 * @property TaskHistory[] $taskHistories
 * @property TaskPriority $taskPriority
 * @property TaskStatus $taskStatus
 */
class Task extends \yii\db\ActiveRecord
{
    /**
     * {@inheritdoc}
     */
    public static function tableName()
    {
        return 'task';
    }

    /**
     * {@inheritdoc}
     */
    public function rules()
    {
        return [
            [['title', 'description', 'employee_id', 'task_priority_id',
            'date_start', 'date_finish'], 'required'],
            [['description'], 'string'],
            [['employee_id', 'task_priority_id', 'task_status_id'], 'integer'],
            [['date_start', 'date_finish'], 'safe'],
            [['title'], 'string', 'max' => 255],
            [['employee_id'], 'exist', 'skipOnError' => true, 'targetClass' =>
            User::class, 'targetAttribute' => ['employee_id' => 'id']],
            [['task_priority_id'], 'exist', 'skipOnError' => true, 'targetClass'
            => TaskPriority::class, 'targetAttribute' => ['task_priority_id' => 'id']],
            [['task_status_id'], 'exist', 'skipOnError' => true, 'targetClass' =>
            TaskStatus::class, 'targetAttribute' => ['task_status_id' => 'id']],
        ];
    }

    /**
     * {@inheritdoc}
     */
}
```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						63
Змн.	Арк.	№ докум.	Підпис	Дата		

```

public function attributeLabels()
{
    return [
        'id' => 'ID',
        'title' => 'Заголовок',
        'description' => 'Опис',
        'employee_id' => 'Працівник',
        'task_priority_id' => 'Пріоритет',
        'task_status_id' => 'Поточний статус',
        'date_start' => 'Дата початку',
        'date_finish' => 'Дата завершення',
        'employee.username' => 'Відповідальний',
        'taskPriority.title' => 'Пріоритет',
        'taskStatus.title' => 'Поточний статус',
    ];
}

/**
 * Gets query for [[Employee]].
 *
 * @return \yii\db\ActiveQuery
 */
public function getEmployee()
{
    return $this->hasOne(User::class, ['id' => 'employee_id']);
}

/**
 * Gets query for [[TaskHistories]].
 *
 * @return \yii\db\ActiveQuery
 */
public function getTaskHistories()
{
    return $this->hasMany(TaskHistory::class, ['task_id' => 'id']);
}

/**
 * Gets query for [[TaskPriority]].
 *
 * @return \yii\db\ActiveQuery
 */
public function getTaskPriority()
{
    return $this->hasOne(TaskPriority::class, ['id' => 'task_priority_id']);
}

/**
 * Gets query for [[TaskStatus]].
 *
 * @return \yii\db\ActiveQuery
 */
public function getTaskStatus()
{
    return $this->hasOne(TaskStatus::class, ['id' => 'task_status_id']);
}

public function beforeSave($insert)
{
    if ($insert) {
        $this->manager_id = Yii::$app->user->getId();
        $this->task_status_id = 1;
    }
}

```

```

        return parent::beforeSave($insert); // TODO: Change the autogenerated
stub
    }

    public function afterSave($insert, $changedAttributes)
    {
        if ($insert) {
            $taskHistory = new TaskHistory();
            $taskHistory->task_id = $this->id;
            $taskHistory->task_status_id = $this->task_status_id;
            $taskHistory->user_id = Yii::$app->user->getId();
            $taskHistory->date_added = date("Y-m-d h-i-s");
            $taskHistory->save();
        }
        parent::afterSave($insert, $changedAttributes); // TODO: Change the
autogenerated stub
    }
}

```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						65
Змн.	Арк.	№ докум.	Підпис	Дата		