

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра комп'ютерних систем та мереж

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавра

(ступінь вищої освіти)

на тему РОЗРОБКА САФМ СИСТЕМИ ДЛЯ ПОКРАЩЕННЯ ОБСЛУГОВУВАННЯ
ІНФРАСТРУКТУРИ ПІДПРИЄМСТВА

(назва теми)

Виконав(ла): студент(ка) 4 курсу, групи КНТ-520

Спеціальності 123 Комп'ютерна інженерія

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерна інженерія

ДУДНИК А.В.

(ПРИЗВИЩЕ та ініціали)

Керівник КИРИЧЕК Г.Г.

(ПРИЗВИЩЕ та ініціали)

Рецензент КОЗИНА Г.Л.

(ПРИЗВИЩЕ та ініціали)

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

Факультет Комп'ютерних наук і технологій

Кафедра «Комп'ютерні системи та мережі»

Ступінь вищої освіти бакалаврський

Спеціальність 123 Комп'ютерна інженерія

(код і найменування)

Освітня програма (спеціалізація) _____

Комп'ютерна інженерія

(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри КУДЕРМЕТОВ Р.К.

Р.К. Кудерметов

“ ” _____ 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ДУДНИКА Андрія Володимировича

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Розробка САФМ системи для покращення обслуговування інфраструктури підприємства

Керівник проекту (роботи) к.т.н., доцент, КИРИЧЕК Галина Григорівна

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом закладу вищої освіти від « 01 » квітня 2024 року
№ 105

2. Строк подання студентом проекту (роботи) 27 травня 2024 року

3. Вихідні дані до проекту (роботи) Технічне завдання на розробку системи. Технології розробки: MySQL, PHP, JavaScript. Програмне забезпечення: PhpStorm, Open Server, Windows 11.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Вибір технологій та програмних засобів. 3. Основні рішення з реалізації системи. 4. Тестування та аналіз системи. 5. Керівництво оператора

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Вимоги до моделі системи. Діаграма станів заявки. Алгоритм роботи заявки. Діаграма класів.

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	КИРИЧЕК Г.Г., доцент		
Нормоконтроль	ПОЛЬСЬКА О.В. ст. викл.		

7. Дата видачі завдання « 1 » квітня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Постановка завдання	1 тиждень	Завдання, ТЗ
2	Дослідження предметної області	2 тиждень	Розділ 1
3	Аналіз технології розробки	3 тиждень	Розділ 2
4	Розробка алгоритму роботи заявок	4тиждень	Розділ 3
5	Створення баз даних	4 тиждень	Розділ 3
6	Розробка програми	5 тиждень	Розділ 3
7	Тестування та аналіз роботи	5 тиждень	Розділ 4
8	Керівництво оператора	5 тиждень	Розділ 5
9	Оформлення графічної роботи	6 тиждень	Додатки
9	Оформлення пояснювальної записки та документів до неї. Проходження нормконтролю	6 тиждень	Додатки
10	Захист роботи	6 тиждень	

Студент(ка)



(підпис)

Андрій ДУДНИК

(прізвище та ініціали)

Керівник проекту (роботи)



(підпис)

Галина КИРИЧЕК

(прізвище та ініціали)

РЕФЕРАТ

ПЗ: 61 с., 15 рис., 1 дод., 23 джерел, 9 табл.

ВЕБ-ДОДАТОК, УПРАВЛІННЯ ІНФРАСТРУКТУРОЮ,
АВТОМАТИЗАЦІЯ, БАЗА ДАНИХ, ПІДПРИЄМСТВО, СИСТЕМА.

Об'єктом є процес розробки автоматизованої системи CAFM

Предметом є моделі, алгоритми та програмні засоби розробки та використання CAFM систем на підприємстві та її вплив на покращення управлінських процесів у підприємстві

Мета роботи – розробка, впровадження CAFM системи для підвищення ефективності управління інфраструктурою підприємства та покращення взаємодії між підрозділами, та підрядними організаціями

Матеріали, методи та технічні засоби: структурне та об'єктно орієнтоване програмування, технологія та пристрій PHPStorm, Laravel, JavaScript, Open Server, персональний комп'ютер з процесором Intel i5-12500H під управлінням операційної системи Microsoft Windows 11.

Результати. Створено систему, яка забезпечує автоматизацію процесів прийому та обробки заявок. Проведено тестування системи, яке підтвердило стабільну та ефективну роботу.

Висновки. Завдяки впровадженню системи може покращитись управління інфраструктурою підприємства, також це допомагає оперативності виконання заявок та взаємодія між підрозділами

Галузь використання –торгові центри, медичні та освітні заклади, офісні будівлі, промислові підприємства та інші об'єкти

Економічна ефективність. Знижаються витрати на технічне обслуговування та ремонт інфраструктури, зменшення часу простою.

ЗМІСТ

Вступ.....	9
1 Аналіз технічного завдання.....	10
1.1 Опис предметної області.....	10
1.2 Аналіз подібних систем	13
1.3 Модель системи	15
1.4 Постановка завдання.....	18
2 Вибір технологій та програмних засобів	19
2.1 Вибір серверної мови програмування	19
2.2 Вибір фреймворку	23
2.3 Аналіз СУБД	27
2.4 Вибір клієнтської мови програмування	31
2.5 Вибір засобів розробки	36
3 Основні рішення з реалізації системи	36
3.1 Розробка бази даних	36
3.2 Алгоритм роботи заявки	40
3.3 Проектування баз даних.....	42
4 Тестування та аналіз роботи	46
4.1 Характеристики програми.....	46
4.2 Запуск застосунку.....	46
4.3 Загальні відомості про програму	47
4.4 Структура програми.....	47
4.5 Тестування роботи	48
5 Керівництво оператора	51
5.1 Призначення програми	51
5.2 Умови виконання програми	51
5.3 Експлуатація програми	52

Висновки	59
Перелік джерел посилання	60

Перелік графічних матеріалів:

Плакат 1 – Вимоги до моделі даних;

Плакат 2–Алгоритм станів заявки;

Плакат 3 – Проходження заявки;

Плакат 4 – Діаграма класів.

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

CAFM – Computer-Aided Facility Management;

UI – User Interface;

FM – Facility Management;

SQL – Structured Query Language;

CMMS – Computerized Maintenance Management System;

PHP – Hypertext Preprocessor;

HTTP – HyperText Transfer Protocol;

HTML – HyperText Markup Language;

PEAR – PHP Extension Application Repository;

API – Application Programming Interface;

MVC – Model-View-Controller;

CMN – Content Management System;

CDN – Content Delivery Network;

CSS – Cascading Style Sheets;

БД – База Даних;

СУБД – Система Управління Базами Даних.

ВСТУП

У наш час світ переживає етап глобалізації, який характеризується взаємозалежністю та інтеграцією технологій у різних сферах бізнесу. Глобалізація також загострює конкуренцію на міжнародних ринках, змушуючи компанії знаходити нові шляхи підвищення ефективності та оптимізації своїх ресурсів. Одним з таких аспектів є галузь управління інфраструктурою підприємства, в якій питання щодо оптимізації управління будівлями, обладнанням і ресурсами стало вирішальним. Одним із найефективніших інструментів вирішення цих проблем є комп'ютерне управління інфраструктурою.

Система CAFM – це програмний комплекс, який автоматизує процеси управління інфраструктурою підприємства, особливо технічне обслуговування, ремонт, управління простором, енергоспоживанням та інші важливі аспекти. Ці системи дають можливість інтегрувати різні дані та процеси в єдине інформаційне середовище, тим самим суттєво підвищуючи ефективність і точність управління.

Окремо слід відзначити взаємодію підприємств з підрядними організаціями, процеси яких зав'язані на роботі з архівами та паперами. У цьому випадку значна частина часу та ресурсів витрачається на пошук, обробку та зберігання документів, що значно уповільнює робочі процеси та знижує ефективність управління.

В Україні розвиток CAFM систем набирає обертів та вже зараз можна спостерігати позитивні зміни в підприємствах, які почали впроваджувати ці технології. Враховуючи міжнародний досвід та сучасні тенденції, розробка та впровадження систем CAFM на українських підприємствах є надзвичайно актуальним завданням, яке сприятиме підвищенню конкурентоспроможності та ефективності управління інфраструктурою.

1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

1.1 Опис предметної області

Міжнародні компанії дуже часто залишають не основні сфери свого бізнеса на аутсорсинг. Якщо беремо за приклад управління інфраструктурою, то на аутсорсингу працюють підрядні організації, які займаються ремонтом будівель, технічним обслуговуванням інженерних мереж, прибиранням, тощо. Інформація про виконану роботу зберігається в окремих комп'ютерних системах або у паперових архівах. Комунікація та обмін даними між різними відділами та організаціями проходить за допомогою дзвінків або особистих зустрічей, через що виникають ситуації, коли робочий процес на деякий час призупиняється, та його продовження стає складним через неактуальність інформації, перешкоди з технічних та організаційних сторін. Також не слід забувати, що у відділах обслуговування інфраструктури працює невелика кількість людей, через що непомічені дефекти на великому підприємстві можуть перетворитися у величезні.

Якщо розглядати впровадження додаткових систем управління, таких як окремі програмні забезпечення для різних аспектів управління, виникає проблема збільшення незалежних систем, які потребують постійного оновлення та обслуговування. Це не практично, оскільки різні системи можуть мати різні конфігурації, що ускладнює їх інтеграцію та створює додаткові труднощі для користувачів.

Рішенням цієї проблеми може стати система, яка об'єднує усі процеси у єдине інформаційне середовище. Однак важливо приділяти увагу також на інтеграцію та автоматизацію даних, тому що без цього можуть виникнути проблеми з ефективністю. Наприклад, окремі підрядні організації можуть використовувати різні системи обліку та управління, що ускладнює обмін інформацією та створює додаткові бар'єри для прийняття оперативних рішень.

Facility management - це вид сервісного бізнесу, за якого компанія виконавець

бере на себе всі непрофільні види діяльності клієнта, що не належать до його основного виду діяльності. Клієнт таким чином має можливість концентруватися на основному бізнесі.

САФМ виникла наприкінці 1980-х років з використанням персонального комп'ютера (ПК) для автоматизації збору та ведення інформації про управління об'єктами нерухомості. Широке використання систем інформаційних технологій (ІТ) майже у всіх галузях з часом проникло і в будівництво та управління об'єктами нерухомості. Після розвитку систем баз даних на основі Інтернету, використання високотехнологічних інструментів у практиці управління збільшилося і в цьому секторі. Системи САФМ надали керівнику об'єкта інструменти для відстеження, планування, управління та звітування щодо інформації про об'єкти.

Системи САФМ об'єднують та аналізують складні дані для покращення практики управління об'єктами в різних галузях, включаючи уряд, охорону здоров'я, освіту, комерційну діяльність та промисловість. Система САФМ надає особам, які приймають рішення, можливість автоматизувати багато функцій управління об'єктами, що вимагають великих обсягів даних, і, як правило, призводить до постійної економії коштів і поліпшення використання активів протягом усього їхнього життєвого циклу.

Програмне забезпечення САФМ надає керівнику об'єкта адміністративні інструменти та можливість відстежувати, управляти, звітувати та планувати експлуатацію об'єкта. Ця функція асоціюється більше з адміністративною, а не з технічною діяльністю. Наприклад, системи САФМ можуть містити дані, що ідентифікують відділи та персонал, ідентифікують та відстежують, хто має права доступу до приміщень та рівень безпеки, а також інформацію про орендовані приміщення, включаючи інформацію про оренду, доходи та повернення орендарями коштів [2].

Щоб не плутати комп'ютеризовану систему управління експлуатацією (CMMS) з системою САФМ, уявіть собі палату пацієнта в лікарні. Забезпечення того, що система виклику медсестер у палаті «належним чином перевіряється, обслуговується і ремонтується» - це діяльність CMMS. «Знання» про персонал

медичного відділення і конкретних пацієнтів у палаті, вміст палати (телефони, телевізори, ліжка, включаючи їх переміщення з палати в палату) і підключення обладнання (електрика, кисень, зв'язок і т.д.) відносяться до діяльності CAFM. Системи CMMS і CAFM почали продовжувати об'єднуватися в інтегровані системи управління робочими замовленнями (IWOMS) [1].

Хоча не існує ідеальної моделі, придатної для всіх ситуацій, для задоволення конкретних потреб керівника об'єкта, добре розроблена система CAFM часто включає в себе різноманітні функції та можливості. Системи CAFM зазвичай надають і підтримують інформацію про плани поверхів, опис майна, використання площ, споживання енергії, розташування обладнання та інші дані про критично важливу інфраструктуру, які стосуються сектору, який вона обслуговує.

Системи CAFM набули найбільшого поширення у сфері управління нерухомістю. Нерухомість, як найсуттєвіша частина основних фондів підприємств та організацій, а також, як об'єкт бізнесу, наприклад під час девелопменту та оренди, є основним об'єктом для послуг типу facility management і відповідно систем CAFM.

Типові CAFM-системи представляються як поєднання CAD та/або програмного забезпечення СУБД із специфічними можливостями управління інформацією про об'єкти нерухомості. CAFM належить до корпоративних інформаційних систем підприємств та організацій [2].

CAFM - це особливий тип програмного забезпечення для управління об'єктами, який дозволяє користувачам оптимізувати багато клієнтоорієнтованих послуг в організації, таких як бронювання номерів, обслуговування об'єктів, управління простором, управління активами і так далі. Крім того, програмне забезпечення для автоматизованого управління об'єктами (CAFM) дозволяє планувати, виконувати і контролювати різні види діяльності в будівлях, коли вони вступають у фазу технічного обслуговування.

На найпростішому рівні системи CAFM допомагають на етапі планування різноманітних конкретних операцій на об'єкті або в будівлі, включаючи більшість рутинних щоденних операцій. Інший спосіб поглянути на програмне забезпечення

САФМ - як на комплексний інструмент обслуговування будівлі з різноманітними корисними функціями та опціями [3].

Система розробляється для підприємства, яке зав'язане на роботі з підрядними організаціями. Для забезпечення ефективній роботі та координації дій з підрядниками, система має відповідати специфічним параметрам та інтегруватися у структуру підприємства.

1.2 Аналіз подібних систем

На міжнародному ринку існує багато різних САФМ систем, кожна з них має свої особливості та переваги. Функціонал цих систем відрізняється від потреб сфер бізнесу в яких вони використовуються.

Незважаючи на те, що існує величезна кількість різноманітних факторів, які можуть вплинути на вибір конкретного програмного забезпечення для фасіліті менеджменту, все ж таки можна дати загальні рекомендації щодо того, як вибрати правильний тип для вашої компанії. Використовуючи цю логіку, існує три рекомендації:

- визначте, який функціонал вам потрібен. Різні види програмного забезпечення для обслуговування інфраструктури створюють багато плутанини, особливо через те, що їхні функції перетинаються. На щастя, існують певні ознаки, за якими можна розрізнити конкретні типи систем управління експлуатацією;
- визначте головну мету рішення. Точне знання того, для чого ви збираєтеся використовувати програмне забезпечення для фасіліті менеджменту, є важливою складовою вибору в першу чергу. Об'єкти з одним об'єктом зазвичай потребують лише базового програмного забезпечення для управління технічним обслуговуванням, і в цих випадках найкраще підходить CMMS;
- вивчіть можливі варіанти розгортання. Методи розгортання систем управління об'єктами сильно відрізняються один від одного. Існує три основні

способи розгортання систем фасіліті менеджменту: хмарне, локальне та гібридне.

Розгортання хмарних сервісів, як правило, відносно дешеве, причому ціна швидко зростає для великих компаній з великою кількістю співробітників або з великою кількістю налаштувань. З іншого боку, хмарне розгортання є гнучким і доступ до нього можна отримати не лише з настільних комп'ютерів, але й за допомогою телефонів або планшетів. Крім того, оскільки в цьому випадку захист ваших даних покладається на постачальника хмарних послуг, він нерідко пропонує найсучасніші рішення для захисту даних на ринку.

Локальне управління об'єктами використовує ваше власне обладнання як основне сховище для всієї системи. Це, як правило, трохи дорожче, ніж хмарний варіант, але часто це одноразова покупка, з набагато меншою кількістю необхідних додаткових покупок. Локальне розгортання пропонує набагато більше можливостей для кастомізації, ніж хмарний сервіс, але місцезнаходження ваших даних означає, що вам також доведеться турбуватися про питання кібербезпеки.

Гібридне розгортання - це комбінація обох варіантів, і існує багато варіацій гібридного розгортання. Більшість з них дійсно унікальні і залежать від галузі, в якій ви працюєте.

FMX – це програмне забезпечення для обслуговування інфраструктури, яке призначене для робочих процесів з технічного обслуговування. Воно скорочує час вирішення завдань і пропонує уніфіковану інформаційну панель для керування всіма вашими активами з одного місця. FMX допомагає організаціям, підвищуючи продуктивність активів, оптимізуючи процеси та здатність перетворювати корисну інформацію на результати.

Procore є одним із найпопулярніших постачальників програмного забезпечення на будівельному ринку, що пропонує широке рішення для управління ресурсами та фінансами на кожному етапі проекту. Він може діяти як комунікаційна платформа між командами, де кожен учасник проекту був безпосередньо залучений до одного процесу. Procore також пропонує мобільні

версії свого програмного забезпечення, що являється плюсом, тому що це виводить оперативність роботи на інший рівень.

Urkeeper – відомий учасник ринку програмного забезпечення для управління інфраструктурою, який пропонує сучасні рішення для управління активами та обслуговування з численними функціями. Вся система також досить портативна, тому до неї можна отримувати доступ не лише з комп'ютера, а й за допомогою телефону чи планшета. Його можна використовувати для створення робочих процесів, отримування сповіщень у разі виникнення надзвичайних ситуацій та отримувати повідомлення, коли робочі заявки змінюють свій статус. [2]

Оглянувши основні пропозиції ринку готових рішень можна підвести підсумки, що їх головними недоліками є висока вартість впровадження та обслуговування, складність початкового налаштування та адаптації під конкретні задачі підприємства, а також недостатня гнучкість і масштабованість. Плата за використання систем частіше за всього йде підпискою, де чим більше працює користувачів, тим більше ціна.

1.3 Модель системи

Метою розроблюваної системи є автоматизація робочого процесу обслуговування інфраструктури підприємства. Сформулюємо модель системи для кращого розуміння структури, функціональності та взаємодії компонентів системи.

Модель є представленням реального об'єкту, системи або поняття в деякій формі, відмінній від форми їх реального існування.. Першим кроком є вибір правильної мови моделювання, оскільки в комп'ютерних науках існує декілька мов моделювання. Основна відмінність між цими мовами полягає в об'єктно-орієнтованому або процедурно-орієнтованому моделюванні.

Об'єктно-орієнтовані мови моделювання розбивають проблему на найменші незалежні частини і представляють ієрархічні зв'язки між цими об'єктами.

Процедурно-орієнтовані (алгоритмічно засновані) мови моделювання досягають успіху в окремих кроках, щоб досягти вирішення проблеми. Ця концепція дуже добре працює для чітко визначеної проблеми в чітко визначеному середовищі. Але інформація CAFM та аналіз цих даних мають бути розраховані на середньострокове та довгострокове планування (можливо, на понад 50 років), тому прогнозування майбутніх потреб в аналізі неможливе або можливе лише частково. Як логічний наслідок, здається, що об'єктно-орієнтовані приклади для моделювання даних CAFM є розумною стратегією.

Для того, щоб розробити модель даних для системи CAFM, необхідно визначити процес управління фінансами компанії. Цей процес є основою для всієї системи CAFM. CAFM може лише підтримувати процес фасіліті менеджменту. Моделювання даних для систем CAFM - це робота з інтеграції всіх тем, пов'язаних з управлінням будівлею. Для того, щоб побудувати модель даних для системи CAFM, необхідно об'єднати всі різні розуміння управління в технічному, фінансовому сенсі та з точки зору зручності використання. Фінансове управління будівлею зосереджене на центрах витрат і загальних витратах будівлі. Основною метою є зменшення загальних витрат. Технічний менеджмент зосереджений на функціональності будівлі. Кожне приміщення повинно мати певну температуру, вологість, електрику, телефонні лінії, мережу, підключення до Інтернету тощо. Основна частина технічного менеджменту полягає в тому, щоб підтримувати будівлю в робочому стані. Вся інформація для виконання вищезгаданих обов'язків повинна бути представлена в системі CAFM.

На цьому етапі відбувається генерування альтернатив, тобто ідей та можливих шляхів досягнення визначеної мети з використанням формальних і неформальних методів системного аналізу. Генерування альтернатив є творчим процесом і його результати залежать як від повноти інформації зібраної на першому етапі так і від рівня професійної підготовки експертів, що виконують побудову моделей.

Основою створення як формальних так і неформальних моделей є з'ясування структури системи, а саме основних її елементів і частин, а також зв'язків між

ними, що реалізують відносини між елементами, компонентами, підсистемами і системами в цілому [4].

Ціль полягає в тому, щоб представити всю важливу інформацію для процесу управління об'єктом в узгодженій моделі даних. Процеси, які є важливими для управління, повинні бути представлені в моделі даних. На рисунку 1.1 визначені загальні вимоги до моделей даних.

- структурування даних;
- інтеграція всіх робочих процесів;
- представлення всіх функціональних можливостей та обов'язків.

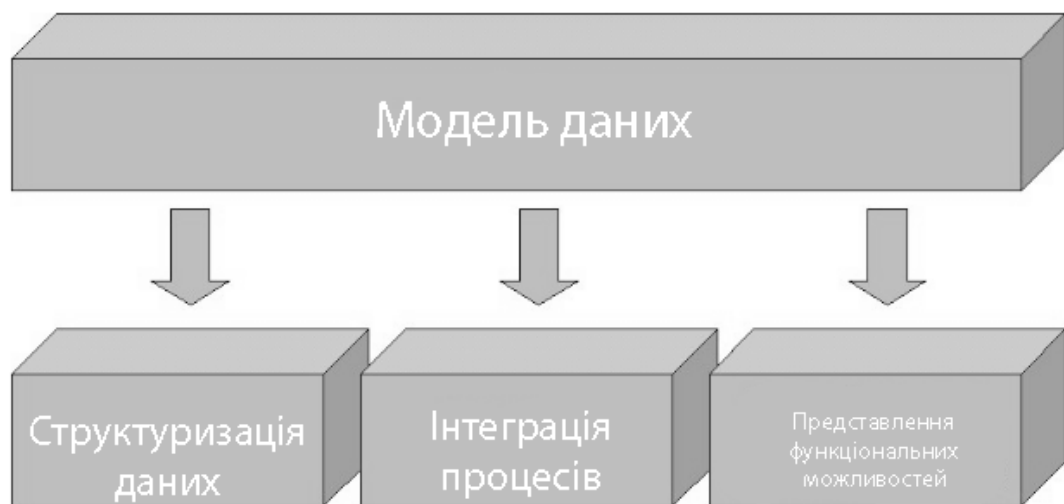


Рисунок 1.1 – Основні вимоги до моделі даних

У кожному програмному проекті збираються та оцінюються вимоги користувачів, джерела даних, функціональні можливості, робочі процеси. Ця база даних є відправною точкою для розробки послідовної та структурованої моделі даних. Ця модель даних інтегрує зібрані робочі процеси та базову функціональність. Інформація, необхідна для робочих процесів, також є частиною моделювання [5].

Для входу у систему необхідно пройти форму авторизації. Користувачі повинні ділитися на декілька типів для коректної працездатності роботи заявок. Заявки повинні мати декілька станів, які зображені у діаграмі станів заявок (рис. 1.1),

для коректної роботи. Комунікація між користувачами може бути створена за допомогою модулем для зберігання та управління коментарями до заявок.



Рисунок 1.2 – Алгоритм станів заявки

1.4 Постановка завдання

Виходячи з моделі системи та визначивши основні вимоги до системи та напрямки її роботи можемо сформулювати основні завдання.

Зробивши висновки з огляду існуючих рішень, для зручності під потреби розробимо веб-додаток.

До основних функцій веб-додатку відносяться:

- автоматизація процесу прийому та обробки заявок;
- моніторинг статусу виконання заявок в режимі реального часу ;
- управління персоналом;
- облік заявок.

Система повинна мати декілька типів користувачів, таких як:

- гість;
- адміністратор системи;
- керівник підрядної організації;
- робітник підрядної організації.

Користувачі цієї системи повинні мати можливість створювати заявки, перезначати їх та писати коментарі щодо робочих процесів. Користувачу має бути зрозумілий інтерфейс застосунку. На головній сторінці має бути наглядна форма де можна переглянути які заявки відправив користувач, та які заявки адресовані до нього.

У робочій заявці, яка адресована до робітника підрядної організації повинна бути кнопка-таймер, яка розраховує час потрачений ним на роботу та прорахунок кошторису його робіт.

2 ВИБІР ТЕХНОЛОГІЙ ТА ПРОГРАМНИХ ЗАСОБІВ

2.1 Вибір серверної мови програмування

Архітектура веб-додатків описує взаємозв'язок між компонентами веб-додатку, їхню структуру та спосіб взаємодії. Вона визначає, як розподіляються функції та обов'язки між різними частинами додатку, щоб забезпечити його ефективну роботу та легкість розширення.

Одним з найпоширеніших підходів до архітектури веб-додатків є клієнт-серверна модель. У цій моделі, веб-додаток складається з двох основних

компонентів: клієнта та сервера. Клієнт (зазвичай веб-браузер) відповідає за відображення інтерфейсу користувача та збирання введених даних, тоді як сервер обробляє запити клієнта, виконує бізнес-логіку та надає необхідні дані.

Веб-додатки складаються з різних компонентів, які співпрацюють між собою для забезпечення функціональності додатку. Основні компоненти включають:

- клієнтська частина (Front-end): Це інтерфейс, який відображається користувачеві у веб-браузері. Вона включає HTML, CSS та JavaScript, які використовуються для створення інтерактивних елементів та забезпечення користувацького досвіду;

- серверна частина (Back-end): Це сервер, який обробляє запити клієнта, зберігає та обробляє дані, виконує бізнес-логіку та повертає відповіді клієнту. Вона може включати різні технології, такі як веб-сервери, бази даних, мови програмування тощо.

У архітектурі веб-додатків використовуються різні шаблони, які допомагають організувати та структурувати компоненти додатку. Деякі з найпоширеніших архітектурних шаблонів включають монолітну архітектуру, клієнт-серверну архітектуру, розподілену архітектуру та мікросервісну архітектуру [6].

У веб-розробці розробники користуються багатьма різноманітними мовами програмування, кожна з яких має свої переваги. Для серверної частини веб-розробки часто використовують такі мови, як: PHP, Python, Ruby, Java. Порівняємо їх для найкращого вибору під наші потреби (таблиця 2.1).

Таблиця 2.1 – Порівняння мов програмування

Критерій	PHP	Python	Java	C#
Типізація	Динамічна	Динамічна	Статична	Статична
Швидкість розробки	Висока	Висока	Середня	Середня
Документування коду	PHPDoc	Sphinx, Docstring	Javadoc	XML
Розгортання	На будь-яких хостингах	Віртуальні середовища Docker	Інтеграція з CI/CD інструментами	Інтеграція з CI/CD інструментами

У ході порівняння ми вияснили, що PHP є найкращим варіантом для веб-розробки завдяки своїй простоті у використанні, широкій підтримці баз даних та великій кількості фреймворків, таких як Laravel і Symfony.

Сьогодні PHP є найпоширенішою мовою програмування в Інтернеті: понад 40 відсотків усіх веб-додатків написано на PHP. Вона встановлена на понад 22 мільйонах серверів доменів (джерело: Netcraft), як показано на Рисунку F-1. Кількість розробників, які використовують PHP, зараз досягла понад 2,5 мільйона. Спільнота, що розробляє PHP, дуже вражає: понад 450 CVS-комітерів, які написали загалом 1 мільйон рядків коду. Спільнота PHP є однією з найбільш динамічних, тисячі людей діляться кодом, розповідають про PHP, підтримують один одного і створюють багато проектів, таких як Вікіпедія, Mambo, PHP-Nuke, FUDforum, SugarCRM і Horde, і це лише деякі з них.

Майбутнє PHP виглядає дуже яскраво. Провідні постачальники платформ, такі як IBM, Oracle, MySQL, Intel і, зовсім недавно, Red Hat, підтримали його. Новий проект співпраці, ініційований Zend Technologies, об'єднує багато провідних компаній та членів спільноти навколо нових ініціатив з відкритим вихідним кодом, спрямованих на виведення PHP на новий рівень шляхом створення промислового рівня, де-факто стандартного середовища для розробки та розгортання веб-додатків на PHP. Перші дві відкриті ініціативи проекту - це Zend PHP Framework

Це фреймворк для веб-додатків, який прискорить і покращить розробку та розгортання критично важливих веб-додатків на PHP.

Також, не слZend приєднується до Eclipse Foundation в якості стратегічного розробника. У співпраці з партнерами вона розроблятиме PHP IDE на основі платформи Eclipse.

PHP - це проста, але потужна мова, призначена для створення HTML-контенту. При використанні PHP ви не обмежуєтесь виведенням лише HTML або інших текстових файлів, ви можете створювати документи будь-якого формату. На даний час він підтримується майже всіма хостинг-компаніями і є одним із лідерів серед мов програмування, які застосовуються для написання динамічних веб-сторінок.

Основними областями застосування PHP є:

- написання сценаріїв на стороні сервера; написання сценаріїв у командному рядку;
- клієнтські програми з графічним інтерфейсом.

Однією з найважливіших особливостей PHP є його широка підтримка баз даних. PHP підтримує всі основні бази даних (включаючи MySQL, PostgreSQL, Oracle, Sybase і інші бази даних), і навіть багато маловідомих. За допомогою PHP створювати веб-сторінки з динамічним вмістом з бази даних напрочуд просто [7].

PHP - це проект з відкритим вихідним кодом від Apache Software Foundation, який є найпопулярнішим додатковим модулем веб-сервера Apache, близько 53% HTTP-серверів Apache мають можливості PHP. PHP особливо добре підходить для веб-додатків баз даних завдяки своїм інструментам інтеграції для веб-середовищ і баз даних. Зокрема, гнучкість вбудовування скриптів у HTML-сторінки дозволяє легко інтегрувати HTML-презентації та код. Підтримка інтеграції з базами даних також є чудовою: більше 15 бібліотек доступні для взаємодії майже з усіма популярними серверами баз даних. У цій книзі ми представляємо всебічний огляд PHP, а також низку потужних розширень, що надаються репозиторієм, відомим як PEAR.

PHP - це найбільш широко підтримувана і використовувана мова веб-скриптів і чудовий інструмент для створення веб-додатків баз даних. Це не означає, що інші скриптові мови не мають чудових можливостей.

Ця мова має відкритий код, завдяки чому зусилля спільноти по його підтримці та вдосконаленню не обмежені комерційними імперативами.

Він є дуже гнучким для інтеграції з HTML. Один або декілька PHP-скриптів можна вбудувати у статичні HTML-файли, що полегшує інтеграцію на рівні клієнта. З іншого боку, це може призвести до змішування скриптів з презентацією.

Завдяки вбудованому скриптовому движку Zend, виконання PHP-скриптів відбувається швидко, а всі компоненти працюють в межах основної пам'яті PHP (на відміну від інших скриптових фреймворків, в яких компоненти знаходяться в окремих модулях). Наші експерименти показують, що для завдань принаймні

помірної складності PHP працює швидше, ніж інші популярні скриптові інструменти [8].

Нарешті, PHP надає бібліотеку PHP-коду для виконання поширених завдань, таких як абстрагування від баз даних, обробка помилок і т.д., за допомогою репозиторію розширень і додатків PHP (PEAR). PEAR - це фреймворк і система розповсюдження багаторазових PHP-компонентів. [6].

Основними перевагами PHP є:

- вбудована підтримка баз даних;
- простий та зрозумілий синтаксис;
- відмінна масштабованість.

2.2 Вибір фреймворку

Фреймворк представляє собою набір бібліотек, інструментів що допомагають розробникам створювати програмне забезпечення ефективніше і стандартизованіше.

Легко зрозуміти, чому вигідно використовувати окремі компоненти, або пакети, які доступні PHP-розробникам. З пакетами хтось інший відповідає за розробку і підтримку ізольованого фрагмента коду, який має чітко визначену роботу, і, теоретично, ця людина має глибше розуміння цього окремого компонента, ніж ви маєте час [9].

Як правило, традиційний метод проектування фреймворків занадто простий, що призводить до значних обмежень, забирає багато часу та інших проблем, для вирішення таких проблем у цій статті представлено метод проектування та реалізації веб-сайту на основі фреймворку Laravel, Laravel робить процес розробки стандартизованим, автоматично обробляючи деякі не пов'язані з бізнес-логікою зв'язки, що дозволяє програмістам зосередитися на реалізації бізнес-логіки. Експериментально та імітаційним моделюванням доведено, що веб-дизайн,

заснований на фреймворку Laravel, має масштабованість та надійність масштабування, що дозволяє підвищити ефективність розробки [10].

Такі фреймворки, як Laravel, а також Symfony, Lumen і Slim, містять набір сторонніх компонентів разом із власним «класом» фреймворку, таким як конфігураційні файли, постачальники послуг, задані структури каталогів і завантажувальні стрічки додатків. Отже, перевага використання фреймворку в цілому полягає в тому, що хтось прийняв рішення не лише про окремі компоненти за вас, але й про те, як ці компоненти повинні працювати разом [6].

Найпопулярніші фреймворки наведені на рисунку 2.1

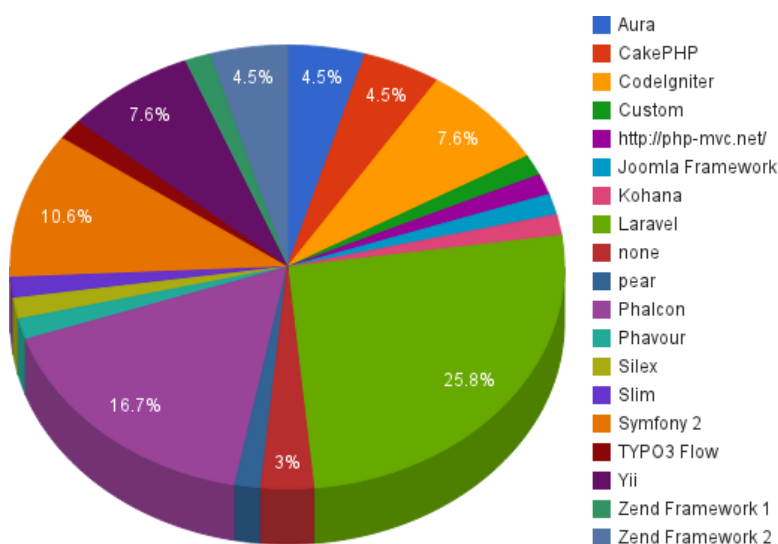


Рисунок 2.1 – Популярність фреймворків для PHP

Завдяки діаграмі ми можемо побачити що найпопулярнішим фреймворком серед веб-розробників на PHP є Laravel. Для точності нашого вибору нам потрібно порівняти два найпопулярніші фреймворки веб-розробки для PHP: Laravel, Simfony. Порівняння наведені в таблиці 2.2.

Виходячи з аналізу, більш всього для нашого проекту підходить фреймворк Laravel завдяки своїми інструментами для розробки, а також шаблонізацією Blade, що дозволяє нам створювати динамічні веб-сторінки з мінімальним обсягом коду.

Таблиця 2.2 – Порівняння фреймворків

Критерій	Laravel	Symfony	Slim	Phalcon
Архітектура	MVC	MVC	Мікрофреймворк	MVC
Вимоги до серверу	PHP 7.3+, Composer	PHP 7.2.5+, Composer	PHP 7.4+ Composer	PHP 7.4 + Composer
Швидкість розробки	Висока	Середня	Висока	Висока
Шаблонізація	Blade	Twig	-	Volt

Виходячи з аналізу, більш всього для нашого проекту підходить фреймворк Laravel завдяки своїми інструментами для розробки, а також шаблонізацією Blade, що дозволяє нам створювати динамічні веб-сторінки з мінімальним обсягом коду.

Перша бета-версія Laravel 1 була випущена в червні 2011 року і була написана повністю з нуля. Вона мала власне об'єктно-реляційне відображення, маршрутизацію на основі закриття, систему модулів для розширення а також помічники для форм, валідації, автентифікації та багато іншого. Ранні версії Laravel розвивалися швидко, і Laravel 2 і 3 були випущені в листопаді 2011 року і лютому 2012 року відповідно. Вони представили контролери, модульне тестування, інструмент командного рядка, контейнер інверсії управління, зв'язки Eloquent та міграції.

У Laravel 4 був переписан весь фреймворк з нуля. На той час Composer, повсюдний менеджер пакетів PHP, вже демонстрував ознаки того, що стане галузевим стандартом, і Тейлор побачив цінність переписування фреймворку як набору компонентів, що розповсюджуються та об'єднуються разом за допомогою Composer.

Було організовано новий набір компонентів під кодовою назвою Illuminate і в травні 2013 року вийшов Laravel 4 з абсолютно новою структурою. Замість того, щоб завантажувати більшу частину свого коду, Laravel тепер витягував більшість своїх компонентів з Symfony, а компоненти Illuminate - через Composer.

У Laravel 4 також з'явилися черги, поштовий компонент, фасади та наповнення баз даних. І оскільки Laravel тепер покладається на компоненти Symfony, було оголошено, що Laravel буде повторювати (не точно, але незабаром після цього) шестимісячний графік випусків, якого дотримується Symfony [8].

По-перше, Laravel - це фреймворк для швидкої розробки додатків. Це означає, що він орієнтований на неглибоку (легку) криву навчання і на мінімізацію кроків між запуском нового додатка і його публікацією. Всі найпоширеніші завдання при створенні веб-додатків, від взаємодії з базами даних до автентифікації, черг, електронної пошти та кешування, спрощуються за допомогою компонентів, які надає Laravel. Але компоненти Laravel не просто чудові самі по собі; вони забезпечують узгоджений API і передбачувані структури у всьому фреймворку.

Це не закінчується на самому фреймворку. Laravel надає цілу екосистему інструментів для створення та запуску додатків. У вас є Sail, Valet і Homestead для локальної розробки, Forge для управління серверами, Envoyer і Vapor для розширеного розгортання. Існує також набір додаткових пакетів: Cashier для платежів і підписок, Echo для WebSockets, Scout для пошуку, Sanctum і Passport для аутентифікації API, Dusk для тестування фронтенду, Socialite для соціального входу, Horizon для моніторингу черг, Nova для створення адміністративних панелей і Spark для перезавантаження вашого SaaS. Laravel намагається позбавити розробників повторюваної роботи, щоб вони могли робити щось унікальне [4].

Laravel — сучасний, гнучкий та ефективний фреймворк, який увібрав в себе найкращі практики програмування на PHP. Його витончений синтаксис в поєднанні з потужними функціями ідеально підходять для створення високонавантажених вебдодатків. Сьогодні він — один з найпопулярніших в усьому світі.

Laravel має гнучку модульну архітектуру "модель-вид-контролер" (MVC). Архітектурна схема MVC — це спосіб організації коду, який дозволяє краще розуміти, як працює сайт і як він взаємодіє з базою даних та користувачами. Кожен з цих компонентів (модель, вид і контролер) виконує свою власну роботу: модель працює з даними, вид показує їх на екрані, а контролер керує тим, як вони взаємодіють один з одним.

По суті, така архітектура MVC визначає, як зберігаються дані та як їх бачитиме користувач вашого вебпродукту: чи у вигляді відкритого онлайн-каталогу товарів, чи як закриті від сторонніх платні курси, чи як складну

логістичну систему з різнорівневим авторизованим доступом. Така архітектура подобається розробникам своєю логікою та зручна для тестування.

Laravel надає цілі пакети інструментів та готових компонентів для розробки різних видів вебдодатків. Окремо його цінують за сет вдалих рішень для електронної комерції, як “кошик”, обробка платежів, управління замовленнями, система автентифікації та облік користувачів. Також з Laravel легко реалізувати багатомовність та багатовалютність, що дозволяє спростити керування різними версіями контенту і надає інструменти локалізації для взаємодії з користувачами з різних країн [11].

Основні переваги Laravel:

- модульна архітектура;
- простий та зрозумілий синтаксис;
- вбудовані інструменти для розробників.

2.3 Аналіз СУБД

Термін «база даних» використовується в різних значеннях, але для цілей цієї книги база даних - це сукупність даних, що зберігаються певним чином організовано. Найпростіший спосіб уявити собі базу даних - це уявити її як картотеку.

У світі баз даних файл називається таблицею. Таблиця - це структурований файл, який може зберігати дані певного типу. Таблиця може містити список клієнтів, каталог товарів або будь-який інший список інформації. Ключовим моментом тут є те, що дані, які зберігаються в таблиці, є одним типом даних або одним списком. Ви ніколи не будете зберігати список клієнтів і список замовлень в одній таблиці бази даних. Це ускладнить подальший пошук і доступ до них. Замість цього ви створите дві таблиці, по одній для кожного списку.

Кожна таблиця в базі даних має ім'я, яке її ідентифікує. Це ім'я завжди

унікальне - це означає, що жодна інша таблиця в цій базі даних не може мати таке саме ім'я.

Таблиці мають характеристики та властивості, які визначають спосіб зберігання даних у них. Вони включають інформацію про те, які дані можуть зберігатися, як вони розбиті, як називаються окремі частини інформації та багато іншого. Набір інформації, що описує таблицю, називається схемою, і схему можна використовувати для опису як окремих таблиць у базі даних, так і всієї бази даних (і зв'язків між таблицями в базі даних, якщо такі є) [12].

Бази даних поділяються на реляційні та нереляційні, нам потрібно обрати який саме тип підходить до нашої роботи.

Реляційні бази (SQL) зберігають структуровані дані, які представляють об'єкти реального світу. Скажімо, це можуть бути відомості про людину, або про вміст кошика для товарів в магазині, згруповані в таблицях, формат яких заданий на етапі проектування сховища.

Нереляційні бази (NoSQL) влаштовані інакше. Наприклад, документо-орієнтовані бази зберігають інформацію у вигляді ієрархічних структур даних. Може йтися про об'єкти з довільним набором атрибутів. Те, що в реляційній БД буде розбито на кілька взаємопов'язаних таблиць, в нереляційних може зберігатися у вигляді цілісної сутності [13].

Реляційні бази даних є традиційним вибором для більшості веб-застосунків. Завдяки їхнім добре відомим моделям даних вони прекрасно підходять для нашого проекту. Тепер залишилось лише порівняти найпоширеніші реляційні бази даних для кращого вибору. До основних реляційних СУБД відносяться: MySQL, PostgreSQL, SQLite. Їх порівняння наведені в таблиці 2.3.

В результаті аналізу найкращими вибором буде MySQL, тому що PHP має високу продуктивність при роботі з базами даних MySQL.

MySQL - це система керування базами даних. У реляційній базі даних дані зберігаються не всі скопом, а в окремих таблицях, завдяки чому досягається виграв в швидкості і гнучкості. Таблиці зв'язуються між собою за допомогою відносин, завдяки чому забезпечується можливість об'єднувати при виконанні запиту дані з

декількох таблиць. SQL як частина системи MySQL можна охарактеризувати як мову структурованих запитів плюс найбільш поширений стандартний мова, яка використовується для доступу до баз даних.

Таблиця 2.3 – Порівняння СУБД

Критерій	MySQL	PostgreSQL	SQLite	MongoDB
Індексування	B-Tree, Full-text	B-Tree, GiST, GIN, BRIN	B-Tree	B-Tree
Критерій	MySQL	PostgreSQL	SQLite	MongoDB
Інструменти адміністрування	MySQL Workbench	pgAdmin, DBeaver	DB Browser for SQLite	MongoDB compass
Швидкість	Висока для читання	Висока для складних запитів	Висока для невеликих баз даних	Висока для невеликих даних
Засоби резервного копіювання	mysqldump, Percona XtraBackup	pg_dump, WAL archiving	Копіювання файлів бази даних	mongodump

MySQL - це система керування базами даних. У реляційній базі даних дані зберігаються не всі скопом, а в окремих таблицях, завдяки чому досягається вигравш в швидкості і гнучкості. Таблиці зв'язуються між собою за допомогою відносин, завдяки чому забезпечується можливість об'єднувати при виконанні запиту дані з декількох таблиць. SQL як частина системи MySQL можна охарактеризувати як мову структурованих запитів плюс найбільш поширений стандартний мова, яка використовується для доступу до баз даних.

Структура MySQL трирівнева: бази даних - таблиці - записи. Бази даних і таблиці MySQL фізично представляються файлами з розширеннями frm, MYD, MYI. Таблиця являє собою сукупність записів. А записи - це сукупність полів різного типу. Ім'я бази даних MySQL унікально в межах системи, а таблиці - в межах бази даних, поля - в межах таблиці. Один сервер MySQL може підтримувати відразу кілька баз даних, доступ до яких може розмежовуватися логіном і паролем. Знаючи ці логін і пароль, можна працювати з конкретною базою даних. Наприклад, можна створити або видалити в ній таблицю, додати записи і т.д. Зазвичай ім'я-ідентифікатор і пароль призначаються хостинг провайдерами, які і забезпечують підтримку MySQL для своїх користувачів [14].

У MySQL оптимізація продуктивності бази даних відома як налаштування даних. Основна мета налаштування даних - знайти і усунути вузькі місця в роботі бази даних. Налаштування даних використовує різні методи, такі як оптимізація запитів, налаштування конфігурації та налаштування обладнання.

Оптимізація запитів - це процес підвищення продуктивності SQL-запитів. Для оптимізації запиту можна змінити структуру запиту, додати або видалити індекси або переписати запит. Налаштування сервера MySQL передбачає коригування його конфігураційних параметрів для підвищення продуктивності. Для оптимізації конфігурації можна використовувати такі параметри, як розмір буфера, паралельність потоків і розмір кешу запитів. Щоб підвищити продуктивність сервера MySQL, апаратна настройка передбачає оновлення апаратного забезпечення сервера. Процесор, пам'ять або сховище можуть потребувати модернізації в рамках апаратного налаштування.

Щоб забезпечити швидкий та ефективний доступ до даних, налаштування даних в MySQL передбачає коригування конфігурації бази даних та продуктивності запитів. Розробники можуть оптимізувати продуктивність своїх баз даних MySQL, використовуючи різні методи та найкращі практики.

Оптимізація запитів є важливою частиною налаштування даних. Це передбачає перегляд планів виконання запитів бази даних, щоб виявити будь-які вузькі місця або потенційні області для оптимізації запитів.

Інструкція MySQL EXPLAIN дозволяє розробникам бачити план виконання запитів і виявляти потенційні області для поліпшення. Індексування стовпців, які часто запитуються, поліпшення об'єднань і підзапитів, а також використання ефективних шаблонів запитів, таких як EXISTS, а не IN - ось кілька типових способів оптимізації пошуку. Розробники можуть змінювати параметри конфігурації сервера MySQL на додаток до оптимізації запитів для підвищення продуктивності. Наприклад, можна змінити розмір буферного пулу, щоб оптимізувати використання пам'яті та підвищити продуктивність запитів. Параметр `max_connections` також можна збільшити, щоб допомогти впоратися з великою кількістю одночасних користувачів.

Нормалізація даних є життєво важливим компонентом налаштування даних. Це передбачає структурування структури бази даних, щоб зменшити дублювання даних і підвищити їхню узгодженість. Розробники можуть підвищити продуктивність запитів і знизити витрати на зберігання, зменшивши надлишковість даних. Нормалізація може бути досягнута шляхом поділу великих таблиць на менші, використання зовнішніх ключів для гарантування посилальної цілісності, уникнення повторюваних груп і уникнення нульових значень.

Нарешті, реплікація може бути використана розробниками для збільшення масштабованості та стабільності додатку. Вони можуть гарантувати, що додаток може обробляти великі обсяги трафіку і продовжуватиме бути доступним у разі відмови сервера, реплікуючи дані між численними серверами [15].

Основні переваги MySQL:

- простота у налаштуванні;
- висока швидкість для операцій читання;
- широка підтримка.

2.4 Вибір клієнтської мови програмування

Під час розробки веб-додатку дуже важливо підібрати правильну комбінацію серверної та клієнтської мови програмування що спростить роботу та ефективно розкриє сильні сторони кожної з них, забезпечуючи високу продуктивність. Порівняємо найвідоміші рішення з вибору клієнтської мови. Порівняння клієнтських мов наведені в таблиці 2.4.

Роблячи висновки з порівняння, можемо надати перевагу JavaScript, так як він легко інтегрується з серверними мовами. JavaScript забезпечить нам якісне створення інтерактивних елементів обробки подій користувача та взаємодію з сервером.

Таблиця 2.4 – Порівняння клієнтських мов

Критерії	TypeScript	JavaScript	Kotlin	Dart
Типізація	Статична	Динамічна	Статична	Статична
Компіляція	Потребує компіляції в JavaScript	Не потребує компіляції	Потребує компіляції в JVM	Потребує компіляції в АОТ
Синтаксис	Подібний до C	Подібний до C	Подібний до Java	Подібний до Java
Модулі та пакети	npm	npm	Maven, Gradle	Pub

JavaScript — це інтерпретована високорівнева мова програмування, яка підтримує імперативний, функціональний і подієво-орієнтовані стилі. Мова програмування з динамічною типізацією. На синтаксис мови програмування JavaScript вплинули мови C/ C++ і Java [16].

З появою на початку поточного десятиліття робочої версії HTML5 цю мову розмітки почали використовувати не тільки як мову визначення вмісту веб-сторінок в мережі Інтернет, але і як мову розмітки візуальних інтерфейсів додатків в платформах розробки на базі мови програмування JavaScript. Прикладами таких платформ програмування є система розробки для магазину додатків Windows 8.1 на базі JavaScript і HTML5, а також відкритий фреймворк PhoneGap, який дозволяє створювати додатки для мобільних пристроїв з використанням мов JavaScript і HTML5 без необхідності використання «рідних» мов програмування під всі мобільні операційні системи (Windows Phone, iOS, Android, Bada і т. д.). В даний час JavaScript є найбільш широко використовуваною мовою програмування. Майже всі, хто має комп'ютер або смартфон, має всі інструменти, необхідні для виконання та створення власних програм JavaScript. Все, що для початкового етапу розробки програм на JavaScript потрібно - браузер і текстовий редактор [17].

Протягом довгого часу не було ніяких засобів зовнішнього збереження даних за допомогою JavaScript. Якщо хотілося зберегти дані, потрібно було відправити форму на веб-сервер і чекати оновлення сторінки. Це перешкоджало процесу створення швидких та динамічних вебдодатків. Однак у 2000 році компанія Microsoft почала поставки технології Ajax (Asynchronous Javascript and XML) [3] у складі Internet Explorer. Незабаром після цього програмну підтримку для всіх

можливостей об'єкта XMLHttpRequest, який лежить в основі цієї технології, було додано і в інші браузери. В 2004 році Google запустила додаток електронної пошти Gmail. Цьому з самого початку аплодували, тому що додаток надавав користувачам майже необмежене сховище для своєї електронної пошти. Gmail також ознаменував важливу революцію в функціональній структурі вебдодатків: пішло в минуле оновлення сторінок. В цілому, Gmail за рахунок використання нової технології Аїах являє собою односторінковий, швидкий і чутливий веб-додаток, який назавжди змінив спосіб проектування додатків цього типу. З тих пір веб-розробники створили додатки майже всіх типів, у тому числі повномасштабні, хмарні офісні пакети, соціальні API, подібні Facebook's JavaScript SDK, і навіть графічно напружені відеоігри.

JavaScript має дуже багаті об'єктно-орієнтовані можливості. Стандарт JSON (JavaScript Object Notation), який використовується майже у всіх сучасних веб-додатках як для комунікацій, так і для збереження даних, є підмножиною чудової нотації об'єктних літералів JavaScript. JavaScript використовує модель прототипного спадкування. Замість класів використовуються прототипи об'єктів. Нові об'єкти автоматично успадковують методи та атрибути свого батьківського об'єкта через ланцюжок прототипів. Прототип об'єкта можна змінювати в будь-який момент, що робить JavaScript дуже гнучкою, динамічною мовою. Прототипне спадкування набагато більш гнучке, ніж класичне спадкування, так що в JavaScript можна імітувати всі можливості моделі та спадкування на основі класів C++ або Java, і в більшості випадків з меншою кількістю коду. Зворотнє не вірно. Всупереч загальноприйнятій думці, JavaScript підтримує такі можливості, як інкапсуляція, поліморфізм, множинне спадкування і композиція.

В той час як кожен додаток, написаний на JavaScript, є унікальним, більшість з них поділяють деякі загальні елементи, такі як інфраструктура хостингу, управління ресурсами, представлення та поведінка користувача інтерфейсу. Далі розглядається, де ці різні елементи знаходяться в додатку і загальні механізми, які дозволяють їм взаємодіяти.

Інфраструктура може мати багато особливостей і може мати багато різних механізмів кешування. Загалом, вона складається з наступних компонентів: сховище даних (data store);

- віртуальна приватна мережа або брандмауер для захисту сховища даних від несанкціонованого доступу;
- рівень сервісу: веб-сервіс Black Box JSON RESTful;
- різні API сторонніх виробників;
- серверна частина додатка або CMS для маршрутизації запитів і доставки сторінок клієнту; Мережа доставки статичного контенту CDN для кешованих файлів (наприклад, зображень, сценаріїв JavaScript, таблиць стилів CSS і клієнтських шаблонів);
- клієнт (браузер).

Більшість з цих компонентів не потребують пояснень, але є деякі важливі моменти, які потрібно знати про зберігання та передачу даних додатків. Сховище даних якраз те, як це звучить: місце для зберігання даних програми. Зараз це, як правило, СУБД з мовою структурованих запитів SQL в якості API. Але в останні роки зростає популярність рішень, які базуються на NoSQL (нереляційних) СУБД. У майбутньому, цілком імовірно, що багато програм будуть використовувати комбінацію реляційних та нереляційних рішень для зберігання даних як окремих програм, так і пов'язаних між собою комплексів різнопланових програм і додатків [16].

Головною особливістю є те, що на JavaScript створюються додатки в браузері і на стороні клієнта. Це дозволяє вивчити одну мову програмування, частково перевикористати готові напрацювання та знання і працювати на frond –end, Back –end, в мобільних додатках, роботобудуванні, для IoT, навіть для машинного навчання, розробляти десктопні додатки. Наприклад, JS застосовується в офісних пакетах Microsoft і OpenOffice, в додатках компанії Adobe навіть.

JS є здебільшого усюди чим ми користуємось, забезпечує нам інтерактивність сайтів. Наприклад, коли заповнюємо форму і натискаємо кнопку «Підписатися», ставимо лайк — миттєва реакція на цю дію зазвичай забезпечується

кодом, написаним на JavaScript.

Так, в першу чергу JavaScript широко використовується у фронтенд-розробці. У найближчому майбутньому JS залишиться основною на цій позиції й надалі. У короткостроковій перспективі нічого тут не замінить JS. Все, що створюється як альтернатива JavaScript, в кінцевому підсумку конвертується в JavaScript.

JavaScript разом з HTML і CSS входить в базовий набір того що вивчають початківці програмісти. Пакетний менеджер npm є найбільшим пакетним менеджером з усіх існуючих. Він забезпечує працю багатьох програм у свою чергу [15].

JavaScript повідомляє веб-сайтам або веб-додаткам, «що зробити». Простими словами, ця мова наказує зображенням анімувати себе, фотографіям — циклічно показувати слайди або автозаповнювати певні частини форм. Розробники, які використовують JavaScript, створюють ці сценарії, а якими такі функції мають виконуватися автоматично [18].

Синтаксис JavaScript зрозумілий всім, хто має досвід роботи з такими мовами, як C++, Java, C# або PHP. Частково популярність JavaScript можна пояснити її близькістю до цих мов, хоча важливо розуміти, що внутрішньо JavaScript реалізується зовсім інакше, ніж компілятори цих мов. JavaScript має дуже багаті об'єктно-орієнтовані можливості. Стандарт JSON, використовуваний майже у всіх сучасних веб-додатках як для комунікацій, так і для збереження даних, є підмножиною чудової нотації об'єктних літералів JavaScript. У сучасних браузерах використовується оперативна компіляція (just-in-time compiling), і більшість коду JavaScript - це відкомпільований, добре оптимізований і виконуваний як машинний код, і тому швидкість виконання близька до програм, написаних на C або C++ [17].

Незважаючи на те, що JavaScript була створена як мова для написання веб-скриптів, вона є мовою програмування загального призначення. Інтерпретатор JavaScript - це машина, яка виконує процеси, описані мовою JavaScript. Перший інтерпретатор JavaScript був реалізований. Свої основні функції JavaScript успадкував від мов програмування Scheme та Self. Від Scheme JavaScript

успадкував свої найфундаментальніші принципи проектування, такі як першокласні функції з лексичним масштабуванням та динамічну типізацію [19].

Основні переваги JavaScript:

- великий вибір бібліотек;
- інтерактивність і динамічність.

2.5 Вибір засобів розробки

Розробка веб-застосунку буде проходити на основі мови PHP через його поширюваність, простоту та вбудовану підтримку баз даних. Він підходить для вирішення широкого кола задач та забезпечує високу швидкість завдяки сучасним оптимізаціям. Велика кількість готових фреймворків, таких як Laravel, надає розробникам чудові інструменти для швидкого створення та масштабованості веб-застосунків. MySQL відомий своєю швидкодією та масштабованістю, що робить його ідеальним вибором для веб-застосунку зі значним навантаженням. JavaScript, як клієнтська мова програмування, значно розширює функціональні можливості веб-застосунку. Завдяки своїй асинхронній природі та широкій підтримці сучасних браузерів, JavaScript дозволяє створювати динамічні та інтерактивні інтерфейси користувача.

3 ОСНОВНІ РІШЕННЯ З РЕАЛІЗАЦІЇ СИСТЕМИ

3.1 Розробка бази даних

Як і інші мови програмування, SQL містить безліч числових, рядкових, датових та інших операторів і функцій. Вирази можуть зустрічатися в декількох місцях в операторі SQL, наприклад, у списку стовпців (значення стовпців можуть

бути результатом обчислення). Але в програмах на PHP/MySQL ви зазвичай використовуєте їх лише в операторах where. Замість того, щоб обчислювати значення стовпців у SQL, простіше просто повернути значення у PHP-програму і виконати будь-які обчислення там. Аналогічно, для значень, які потрібно помістити в рядки, ви захочете виконати всі необхідні обчислення в PHP, а потім передати відповіді в SQL. Таким чином, ви побачите, що багато з того, що розповідається в книгах або курсах по SQL, не буде вам корисним. Вивчаючи SQL, я б зосередився на об'єднаннях і підзапитах (про які ми розповімо пізніше), в яких і полягає справжня сила, і не дуже переймався традиційними виразами мови програмування.

Розвиток SQL з перших днів не зупинився на виразах. Зараз він включає в себе повноцінну мову програмування, завдяки якій ви можете розміщувати процедури всередині бази даних, які будуть виконуватися базою даних. У MySQL це важливо для тригерів [20].

Таблиця - основний об'єкт для зберігання інформації в реляційній базі даних. Вона складається з містять дані рядків і стовпців, займає в базі даних фізичний простір і може бути постійною або тимчасовою. Поле, також зване в реляційній базі даних стовпчиком, є частиною таблиці, за якою закріплений певний тип даних. Кожна таблиця бази даних повинна містити хоча б один стовпець. Рядок даних - це запис в таблиці бази даних, вона включає поля, що містять дані з одного запису таблиці.

Головне в команді створення таблиці - визначення імені таблиці і опис набору імен полів, які зазначаються у відповідному порядку. Крім того, цією командою обмовляються типи даних і розміри полів таблиці.

Ключове слово **NULL** використовується для вказівки того, що в даному стовпці можуть міститися значення NULL. Значення NULL відрізняється від пробілу або нуля - до нього вдаються, коли необхідно вказати, що дані недоступні, опущені або неприпустимі. Якщо вказано ключове слово **NOT NULL**, то будуть відхилені будь-яким спробам помістити значення NULL в даний стовпець. Якщо вказано параметр **NULL**, приміщення значень NULL в стовпець дозволено. За

замовчуванням стандарт SQL передбачає наявність ключового слова NULL [13]. Базами даних керує система управління базами даних (СУБД) або сервер баз даних. Сервер баз даних підтримує мову баз даних для створення та видалення баз даних, а також для управління та пошуку даних. Мова баз даних, яку використовують майже всі сервери баз даних, - це SQL, набір операторів, які визначають дані та маніпулюють ними. Після створення бази даних найпоширенішими операторами SQL є INSERT, UPDATE, DELETE і SELECT, які додають, змінюють, видаляють і шукають дані в базі даних відповідно.

Розроблювана БД повинна містити в собі інформацію про користувачів, інформацію про заявки, зв'язок заявок з користувачем, інформацію про приміщення, коментарі.

Таким чином ми можемо виділити наступні сутності, які будуть зберігатися у БД:

- users;
- requests;
- request_user;
- request_object;
- comments.

Таблиця 3.1 – Атрибути таблиці user

Атрибут	Тип даних	Обов'язковість	Припустиме значення	Ключовий атрибут
id	bigint	+	Bigint max	Первинний ключ
name	varchar	+	255	
email	varchar	+	255	Зовнішній ключ
password	varchar	+	255	
image	varchar		255	
remember_token	varchar		100	
created_at	timestamp		2038-01-19 03:14:07	
updated_at	timestamp		2038-01-19 03:14:07	
role	enum	+	4	

Таблиця users призначена для зберігання інформації щодо користувачів. Опис атрибутів наведено у таблиці 3.1.

Таблиця requests призначена для зберігання інформації щодо заявок. Опис атрибутів наведено у таблиці 3.2.

Таблиця 3.2 – Атрибути таблиці requests

Атрибут	Тип даних	Обов'язковість	Припустиме значення	Ключовий атрибут
id	bigint	+	Bigint max	Первинний ключ
title	varchar	+	255	
description	text	+	255	
object_id	bigint	+	Bigint max	
created_by	bigint	+	Bigint max	
location	text	+	255	
phone	varchar	+	255	
fio	varchar	+	255	
photo	varchar		255	
status	enum	+	4	
created_at	timestamp		2038-01-19 03:14:07	
updated_at	timestamp		2038-01-19 03:14:07	

Таблиця request_user використовується для зв'язку заявок з користувачами. Опис атрибутів наведено у таблиці 3.3.

Таблиця 3.3 – Атрибути таблиці request_user

Атрибут	Тип даних	Обов'язковість	Припустиме значення	Ключовий атрибут
id	bigint	+	Bigint max	Первинний ключ
request_id	bigint	+	Bigint max	Зовнішній ключ
user_id	bigint	+	Bigint max	Зовнішній ключ
created_at	timestamp		2038-01-19 03:14:07	
updated_at	timestamp		2038-01-19 03:14:07	

Таблиця request_object використовується для збереження інформації про приміщення, до яких прив'язуються робочі заявки. Опис атрибутів наведено у таблиці 3.4.

Таблиця 3.4 – Атрибути таблиці request_object

Атрибут	Тип даних	Обов'язковість	Припустиме значення	Ключовий атрибут
id	bigint	+	Bigint max	Первинний ключ
title	varchar	+	255	
description	varchar	+	255	
created_at	timestamp		2038-01-19 03:14:07	
updated_at	timestamp		2038-01-19 03:14:07	

Таблиця comments використовується для збереження коментарів до заявок. Опис атрибутів наведено у таблиці 3.5.

Таблиця 3.5 – Атрибути таблиці comments

Атрибут	Тип даних	Обов'язковість	Припустиме значення	Ключовий атрибут
comment	text	+	255	
user_id	bigint	+	Bigint max	Зовнішній ключ
request_id	bigint	+	Bigint max	Зовнішній ключ
created_at	timestamp		2038-01-19 03:14:07	
updated_at	timestamp		2038-01-19 03:14:07	

3.2 Алгоритм роботи заявки

Робочий процес повинен починатися з користувача з роллю “Гість”. Гостьовий аккаунт призначений для входу звичайних робітників підприємства, які мають доступ лише до створення робочої заявки з можливістю прикріпити фото.

Після того, як заявка створена, вона відправляється до адміністратора системи.

Коли адміністратор системи ознайомився з заявкою, у нього є вибір підтвердити її або скасувати, якщо вона не дійсна. При підтвердженні заявки адміністратор системи вибирає на кого її переназначити: на керівника підрядної організації з прибирання або ж на керівника підрядної організації з ремонту.

Після того як заявка приходить до керівника підрядної організації, він може надіслати заявку назад на уточнення до адміністратору, або ж прийняти її та вибрати працівника, який буде її виконувати.

Працівник підрядної організації бачить список заявок які до нього прийшли та інформацію про них. При перегляді інформації в заявці він має кнопку “Почати роботу” яку він має натиснути прийшовши на місце виникнення проблеми. Після того як проблема усунена працівник має натиснути на кнопку “Закінчити роботу” та система повинна прорахувати кількість затраченого часу та перевести час у грошовий еквівалент, який зазначений у договорі підприємства та підрядної організації. Після виконаної роботи заявка залишається в історії. На рисунку 3.1 продемонстровано процес проходження заявки

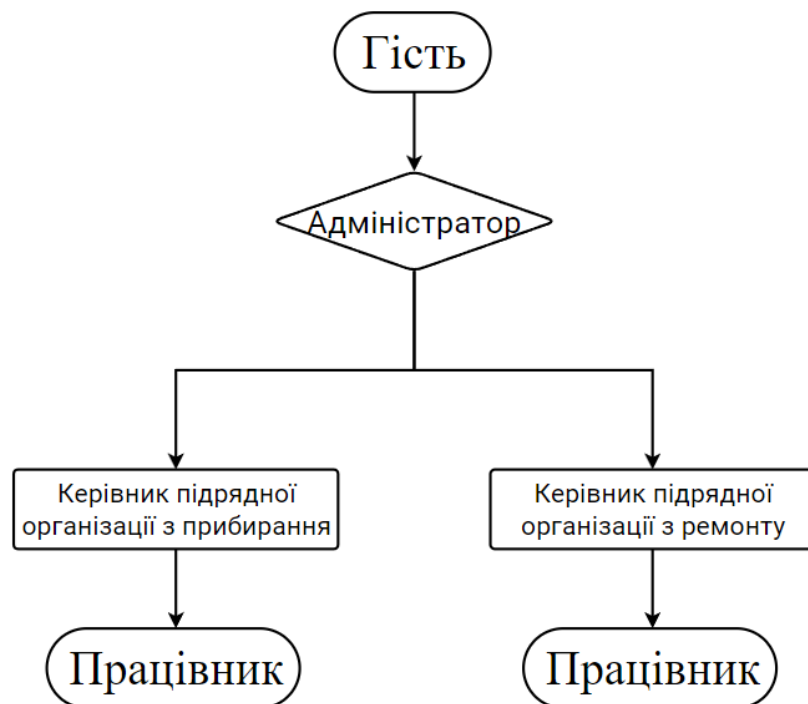


Рисунок 3.1 – Проходження заявки

3.3 Проектування баз даних

Проектування баз даних було створено за допомогою вбудованого підходу у фреймворку Laravel під назвою міграції. Цей підхід дозволяє керувати структурою бази даних за допомогою коду, завдяки чому стає простіше її змінювати та відтворювати.

Міграції більш всього нагадують контроль версій вашої бази даних, що дозволяє визначати та надавати спільний доступ до визначення схеми бази даних програми. Фасад Laravel Schema забезпечує агностичну підтримку бази даних для створення та маніпулювання таблицями в усіх підтримуваних системах баз даних. Як правило, міграції використовують цей фасад для створення та зміни таблиць і стовпців бази даних.

Ви можете скористатися командою `'make:migration Artisan'` для створення міграції бази даних. Нова міграція буде розміщена у вашому каталозі баз даних/міграцій. Кожен файл міграції містить мітку часу, яка дозволяє Laravel визначити порядок міграцій.

Laravel буде використовувати назву міграції, щоб спробувати вгадати назву таблиці і визначити, чи буде міграція створювати нову таблицю чи ні. Якщо Laravel зможе визначити назву таблиці за назвою міграції, Laravel попередньо заповнить згенерований файл міграції вказаною таблицею. В іншому випадку ви можете просто вказати таблицю у файлі міграції вручну.

Якщо ви бажаєте вказати власний шлях до згенерованої міграції, ви можете використати опцію `'-path'` при виконанні команди `'make:migration'`. Вказаний шлях має бути відносно базового шляху вашої програми [21].

Міграції мають декілька переваг над іншими підходами управління базами даних. Завдяки використанню SQL-скриптів та руному управлінню базами даних міграції є гарним вибором.

Для кращого розуміння структури нашої системи сформулюємо UML діаграму класів. Діаграма класів використовується для візуального представлення структури системи на рівні класів і їхніх взаємозв'язків.

В останні роки уніфікована мова моделювання (UML) стала стандартом де-факто для представлення діаграм програмної інженерії. Хоча вона була прийнята як стандарт членами OMG у 1997 році перегляд різних текстів, в яких використовується UML, або дослідження які використовують UML, або дослідження сучасних CASE-засобів, виявляє ряд нотаційних варіацій. Як стандарт, UML все ще розвивається, і в кожній версії стандарту існує багато семантично еквівалентних нотаційних варіацій, з яких автори або видавці можуть вибирати, відповідно до своїх особистих уподобань. Здається, немає жодних підстав для вибору одного варіанту нотації над іншим. Можливо, одна нотаційна варіація легша для сприйняття, ніж інша. Дослідження, про яке йдеться в цій статті, намагається визначити, чи існують якісь відмінності у сприйнятті між п'ятьма різними нотаційними системами.

Діаграми класів UML використовуються для опису статичного вигляду програми. Основними складовими є класи та їхні зв'язки.

Клас - це опис концепції, який може мати атрибути та операції, пов'язані з нею. Класи зображуються у вигляді прямокутників. Зв'язок між двома класами зображується лінією. Відносини успадкування вказують на те, що атрибути та операції одного класу («суперкласу») успадковуються іншими класами («підкласами»), не потребуючи явного представлення в самих підкласах [22].

На рисунку 3.2 зображено діаграму класів, яка відображає структуру основних компонентів, а також їх взаємозв'язки. На діаграмі зображено класи, які відповідають основним сутностям системи.

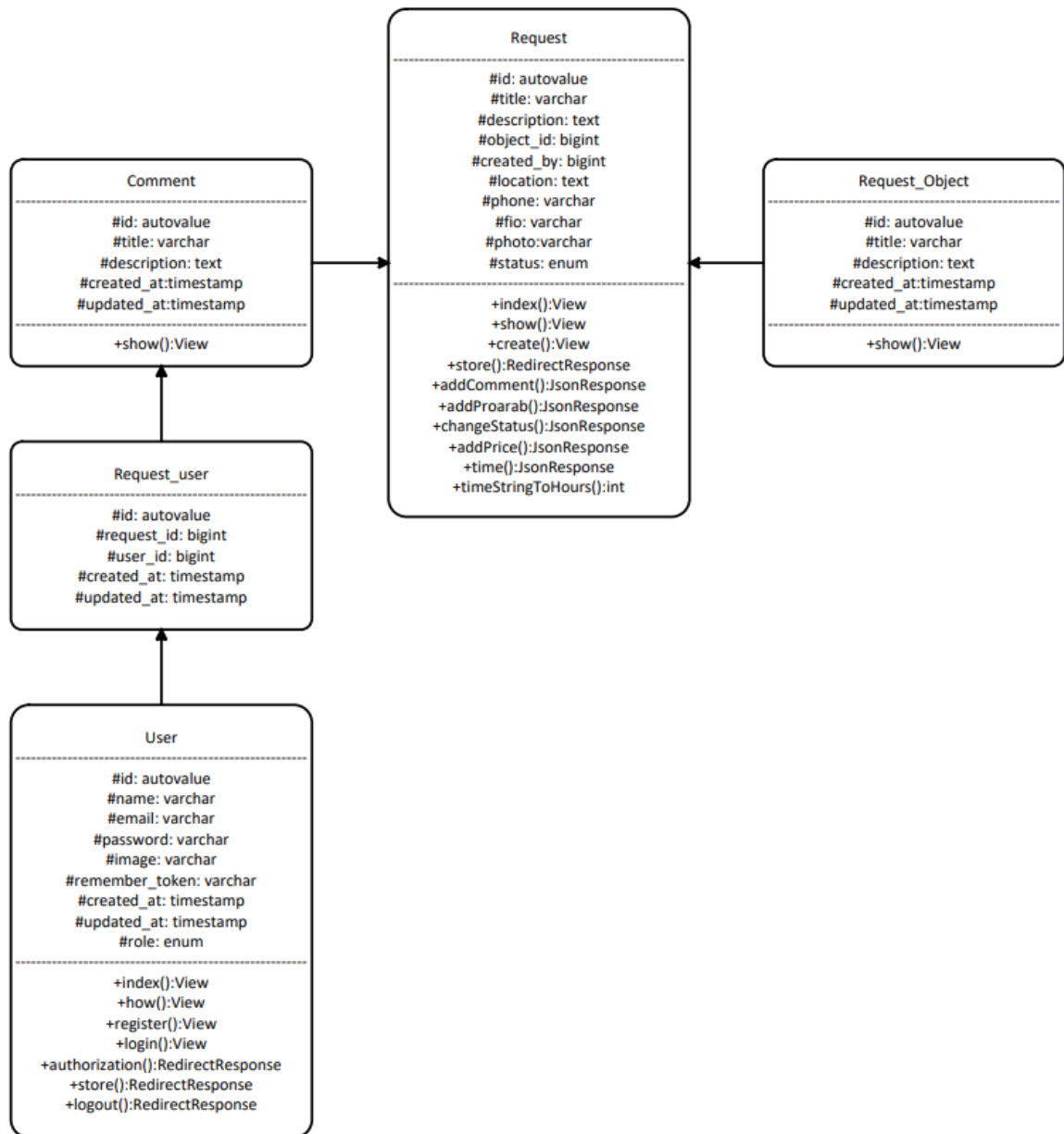


Рисунок 3.2 – Діаграма класів

Нижче приведені списки полів у бази даних та за що вони відповідають.

Таблиця users містить поля:

- Id – унікальний ідентифікатор користувача;
- Name – Ім'я користувача;
- Email – адреса електронної пошти, повинна бути унікальною;
- Password – пароль для входу;
- remember_token – токен для запам'ятовування даних користувача,

дозволяє користувачам залишатися у системі без повторного входу;

- image – шлях до зображення;
- created_at – час створення запису користувача;
- updated_at – час останнього оновлення запису користувача;
- role – роль користувача в системі.

Таблиця requests містить поля:

- id – унікальний ідентифікатор заявки;
- title – назва заявки;
- description – опис заявки;
- object_id – ідентифікатор об'єкта до якого відноситься заявка;
- created_by – ідентифікатор користувача, який створив заявку;
- location – опис місцезнаходження;
- phone – контактний телефон;
- fio – ПІБ особи, яка подала заявку;
- photo – шлях до фотографії;
- status – статус заявки;
- created_at – час створення заявки;
- updated_at – час останнього оновлення заявки.

Таблиця request_user містить поля:

- id – унікальний ідентифікатор зв'язку;
- request_id – ідентифікатор заявки;
- user_id – ідентифікатор користувача;
- created_at – час створення запису;
- updated_at – час останнього оновлення запису.

Таблиця request_object містить поля:

- Id – унікальний ідентифікатор об'єкта;
- Title – назва об'єкта;
- Description – опис об'єкта;
- created_at – час створення запису;
- updated_at – час останнього оновлення запису.

Таблиця comments містить поля:

- id – унікальний ідентифікатор коментаря
- comment – текст коментаря;
- user_id – ідентифікатор користувача, який залишив коментар;
- request_id – ідентифікатор заявки, до якої відноситься коментар;
- created_at – час створення коментаря;
- updated_at – час останнього оновлення коментаря.

4 ТЕСТУВАННЯ ТА АНАЛІЗ РОБОТИ

4.1 Характеристики програми

У веб-додатку було створено усі необхідні компоненти для нормального функціонування. А саме:

- авторизація користувачів;
- можливість створення робочих заявок;
- моніторинг статусу заявок;
- додавання коментарів;
- перегляд отриманих та відправлених заявок.

4.2 Запуск застосунку

Запуск застосунку можливий на будь-якій операційній системі. Перед запуском застосунку обов'язково необхідно запустити міграції, за допомогою команди 'php artisan migrate', що вставить усі необхідні бібліотеки для роботи. У файлі package.json описані усі назви та версії необхідних бібліотек. Для запуску програми необхідно запустити проект за допомогою коректно налаштованого Open

Server.

Після успішного запуску застосунку необхідно переконатися, що додаток працює коректно. Важливим етапом є перевірка роботи авторизації, створення робочих заявок, управління заявками, та інші основні функції системи.

Запуск застосунку можливий на будь-якій операційній системі, що підтримує PHP та MySQL. Перед запуском обов'язково необхідно виконати всі налаштування та підготувати середовище, щоб забезпечити коректну роботу всіх компонентів веб-застосунку.

4.3 Загальні відомості про програму

Програма призначена для спрощення робочих процесів у сфері управління інфраструктурою.

Додаток було створено за допомогою середовища розробки PhpStorm, Open Server.

Базу даних було створено за допомогою веб-додатку phpMyAdmin. Підключення до баз даних проходить у папці проекту у файлі .env.

4.4 Структура програми

Проект зроблений на базі PHP та фреймворку Laravel. Laravel використовує потужний та гнучкий механізм шаблонізації під назвою Blade. За допомогою цього шаблону було створено динамічні веб-сторінки.

Основні компоненти програми включають декілька ключових директорій та файлів. Директорія `app/` містить основні бізнес-логіку додатка, включаючи контролери, посередники, запити та моделі бази даних. Директорія `bootstrap/`

містить файли для початкового завантаження додатка, а `config/` - файли конфігурації, де зберігаються налаштування для бази даних, сесій, кешування та інших сервісів.

4.5 Тестування роботи

Тестування є основним компонентом будь-якого процесу розробки програмного забезпечення, спрямованого на створення високоякісного додатку. Тестування спрямоване на пошук помилок у тестованому об'єкті та надання впевненості в його правильній поведінці шляхом виконання тестованого об'єкта з вхідними значеннями. Веб-додатки є найбільш швидкозростаючим класом програмних систем на сьогоднішній день. Веб-додатки використовуються для підтримки широкого спектру важливих видів діяльності: бізнес-транзакції, наукова діяльність, та медичні системи, наприклад, діагностика на основі експертних систем. Розгортання веб-додатків відбувалося швидкими темпами, що сприяло їх швидкому впровадженню, але водночас знижувало якість програмного забезпечення. Тому всі компоненти веб-додатків повинні бути протестовані. Для того, щоб забезпечити широке і успішне впровадження веб-додатків, методології тестування повинні бути гнучкими, автоматичними і здатними впоратися з їх динамічною природою.

Тестування веб-додатків має багато спільного з тестуванням настільних систем, наприклад, тестування функціональності, конфігурації та сумісності. Тестування веб-додатків складається з аналізу веб-помилки у порівнянні з типовими помилками програмного забезпечення. Існують різні проблеми нееквівалентності між традиційним тестуванням програмного забезпечення та тестуванням веб-додатків. Нижче наведені деякі проблеми:

- веб-додатки, як правило, потребують обслуговування швидше, ніж інші програмні системи;

- веб-додатки мають величезну кількість користувачів, а отже висувають високі вимоги до продуктивності сервера і здатності працювати з паралельними транзакціями. Більше того, коли велика кількість користувачів отримує доступ до ресурсу в один і той самий час, це може призвести до того, що одночасно, доведеться шукати можливості для рендерингу веб-контенту;
- зміна стану, наприклад, за допомогою кнопки «Назад» у браузері або прямого введення URL-адреси в браузері. Неправильна URL-адреса у динамічно сконструйованій веб-сторінці є специфічною помилкою, тоді як індекс за межами масиву є загальною помилкою, яка може виникнути в будь-якій програмі, незалежно від використовуваної веб-технології;
- веб-додатки слід тестувати, щоб перевірити їхню роботу на різних типах веб браузерів і під управлінням різних операційних систем. різних операційних системах;
- однією з основних відмінностей між традиційним і веб-тестуванням є архітектура. специфічним тестуванням є архітектура. У процесі тестування веб-додатків часто буває важко визначити, де саме виникає помилка і в якому помилка виникає і на якому рівні через багаторівневу архітектуру веб-додатків. Веб-додатки надають різноманітні послуги, але зазвичай будуються як трирівнева архітектура. Веб-додаток використовує багаторівневу архітектуру для проектування, розробки та розгортання компонентних додатків для підприємств;
- веб-додатки здатні динамічно відтворювати програмні компоненти під час виконання відповідно до вхідних даних, наданих користувачем, а також на основі відповіді сервера.

Найбільшою проблемою веб-додатків є збереження стану та збереження стану та пристосування звичної послідовності елементів керування історією (кнопки «Назад» і «Вперед»). Технологія веб-додатків AJAX дозволяє документу набувати станів, але коли користувач інстинктивно переходить до елементів керування історією в браузері, в AJAX часто виникає помилка, пов'язана зі зламаною кнопкою «Назад» у браузері. Динамічно змінений стан не реєструється в механізмі історії браузера. Поведінка в стані є складним завданням, оскільки сам

стан містить багато інформації, а поведінка стану змінюється через вміст всередині нього. Потрібен метод для тестування вмісту станів і валідації змін на основі вмісту конкретного стану.

У літературі представлено кілька методів та інструментів для підтримки тестування веб-додатків, але більшість з них зосереджені на відповідності протоколу, навантажувальному тестуванні, виявленні непрацюючих посилань, перевірці HTML та статичному аналізу, які не стосуються функціональної перевірки. Ці дефекти можна легко виявити без ручних зусиль. Інструменти, які автоматично звертаються до функціональних компонентів, є активною сферою досліджень. Методи структурного тестування вимагають побудови моделі, яка зазвичай виконується вручну. Інструменти тестування веб-додатків здатні автоматизувати генерацію тестових кейсів, виконання тестових кейсів та оцінку результатів тестування. За останні роки було розроблено велику кількість різноманітних інструментів для веб-тестування. Список з більш ніж 500 інструментів, перерахованих в 13 категоріях. Існуючі інструменти веб-тестування можуть бути використані для підтримки нефункціонального тестування, навігації веб-сайту. Деякі інструменти можуть бути використані для тестування функціональності веб-додатків [23].

Тестування веб-додатку є важливим етапом розробки, завдяки якому забезпечується висока якість та стабільність кінцевого продукту. Під час розробки системи управління інфраструктурою підприємства, тестування допомагає переконатися, що всі функціональні можливості працюють належним чином, а користувачі можуть користуватися всіма функціями, що передбачені у технічному завданні.

Перед тестуванням було створено тестове середовище проекту, яке повторяє середовище додатку. Це включало налаштування серверу, баз даних та всіх залежностей, які визначені у файлі 'package.json'.

Результати тестування показали, що система працює справно та виконує всі необхідні функції.

5 КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення програми

Розроблена програма призначена для автоматизації управління інфраструктурою. Основними цілями програми є підвищення ефективності роботи підприємства та покращення взаємодії між підрозділами та зовнішніми організаціями.

В даній системі користувачі можуть створювати та керувати робочими процесами, відстежувати статус заявок у реальному часі, додавати коментарі та взаємодіяти з іншими користувачами.

Кінцевою точкою у робочій заявці є підтвердження виконання роботи та ознайомлення з ціною яку склала система.

5.2 Умови виконання програми

В той час як збір вимог в цілому може бути важким, а в деяких випадках навіть неможливим, для додатків на PHP/MySQL він є досить простим. Це значною мірою пов'язано з тим, що додатки, які мають величезні вимоги до продуктивності або настільки складні, що потребують десятків або навіть сотень розробників, дуже рідко, якщо взагалі коли-небудь, програмуються на PHP або використовують MySQL замість баз даних, таких як Oracle, SQL Server або DB2 [20].

Для працездатності веб-застосунку, сервер, на якій розгорнута програма, має підтримувати PHP версії 8.1 або вище. База даних MySQL має бути версії 5.7 або вище для зберігання даних користувачів, заявок та інших об'єктів.

Клієнт повинен мати сучасний веб-браузер (Google Chrome, Mozilla Firefox, Microsoft Edge тощо) для доступу до веб-інтерфейсу програми. Інтернет-з'єднання повинне бути стабільним для забезпечення доступу до серверу та обміну даними в

режимі реального часу.

Система виносить наступні вимоги до апаратного та програмного забезпечення сервера для коректної роботи баз даних MySQL.

Мінімальні вимоги до сервера: 1 комп'ютер з процесором Pentium II 233 МГц і вище, 128 Мб оперативної пам'яті. Рекомендовані вимоги до сервера: 2 комп'ютери з процесором Pentium II 233 МГц і вище, 256Мб оперативної пам'яті. У цьому випадку передбачається, що один комп'ютер буде виконувати функції SQL сервера, а інший - сервера додатків CimWebCenter .

Для роботи системи необхідно наступні, або схоже встановлене програмне забезпечення: Apache Web Server, PHP 4.3.x, MySql, на платформі FreeBSD або Linux. Існує також можливість інсталяції системи на платформі Windows. При цьому вимоги до попередньо програмного забезпечення залишаються колишніми.

Для роботи з клієнтською частиною системи потрібен комп'ютер, підключений по протоколу TCP / IP до мережі, в якій знаходиться сервер із встановленою CimWebCenter.

Мінімальні вимоги до комп'ютера клієнта: Pentium 200 МГц і вище, 32 Мб оперативної пам'яті. Робота з системою здійснюється за допомогою браузера Internet Explorer версії 6.0.

Для роботи в режимі WYSIWIG редактора, також потрібна наявність на клієнтському комп'ютері встановленого пакета MsOffice. При першому запуску системи може знадобитися дистрибутив пакета MsOffice [13].

5.3 Експлуатація програми

Експлуатація веб-застосунку повинна включати у себе всі процеси, які пов'язані з його використанням після розробки та впровадження.

Для того, щоб користуватися системою користувачам необхідно пройти авторизацію по даним які були їм надані (рис 5.1).

Email address

Password

☒ Remember me

[Forgot password?](#)

[Log in](#)

Not a member? [Register](#)

or sign up with:

Рисунок 5.1 – Авторизація у систему

UI веб-застосунку примітивно простий та працювати користувачу у додатку легко. Головна сторінка у всіх типів користувачів однакова, на ній можна переглянути заявки адресовані користувачем та адресовані до користувача (рис. 5.2).

Success User logged in successfully!

user

Admin Dashboard

Created by me

#	Name	Object	Location	FIO
3	Quia nostrum ipsum.	101 Кімната зустрічей	out	FIO enim
7	Atque suscipit.	112 Кімната охорони	labore	voluptatibus
8	Repellat repellat ipsa.	Територія 1	a	nam
10	Corrupti impedit veniam.	101 Кімната зустрічей	cupidditate	molestiae

Created for me

#	Name	Object	Location	Created by	FIO
---	------	--------	----------	------------	-----

Активация Windows
Чтобы активировать Windows, перейдите в раздел "Параметры".

Logout

Рисунок 5.2 – Вигляд головної сторінки

Кожен користувач може натиснути на кнопку create job для відкриття шаблону створення заявки (рис. 5.3). Шаблон створення заявки містить в собі:

- тему;
- опис проблеми ;
- вибір приміщення;
- детальний опис знаходження;
- особистий номер телефону користувача;
- ПІБ;
- фото проблеми.

Create request

Subject
Проблема з дверною ручкою

Description
В кімнаті охорони зламався замок на ручці, двері не зачиняються

Example select
112 Кімната охорони

Description location
Головний вхід

Phone
+ 38 (063) 888-88-88

First & second name
Андрій Дудник

Photo
Выберите файл | sdefault.jpg

Create

Активация Windows
Чтобы активировать Windows, перейдите в раздел "Параметры".

Logout

Рисунок 5.3 – Вигляд шаблону заявки

Після того як заявка створена, вона відправляється до адміністратора системи (рис 5.4). Адміністратор системи має вибрати підрядну організацію, яка буде виконувати заявку (рис. 5.5).

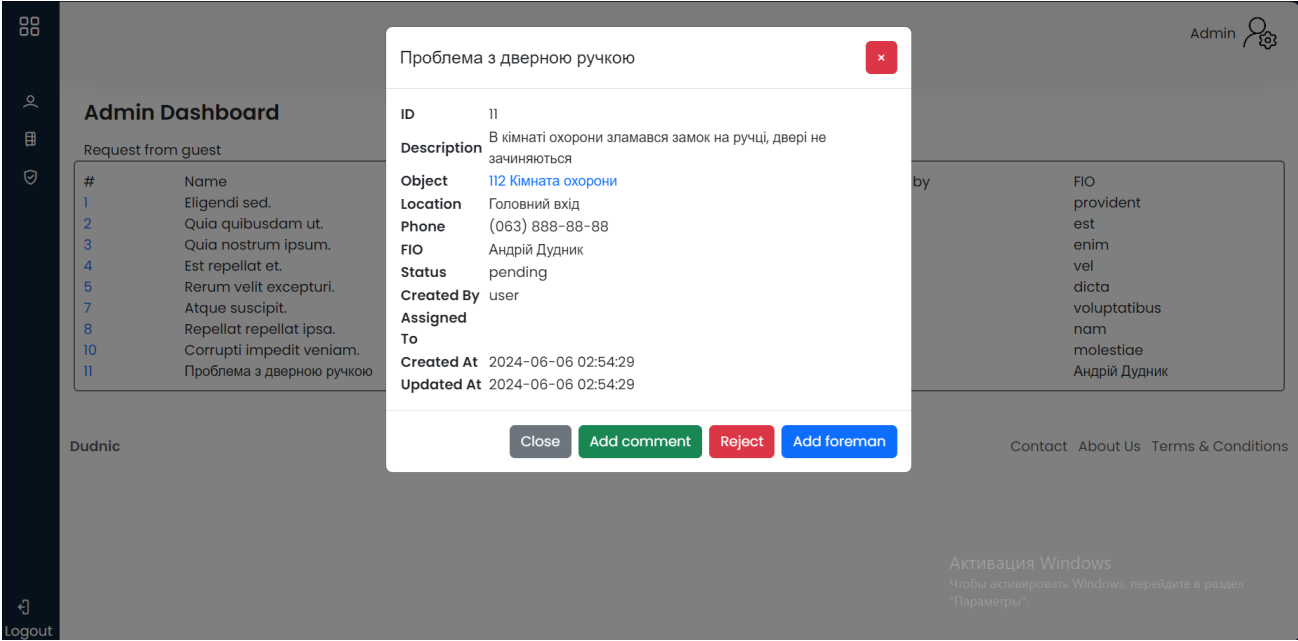


Рисунок 5.4 – Вигляд заявки у адміністратора

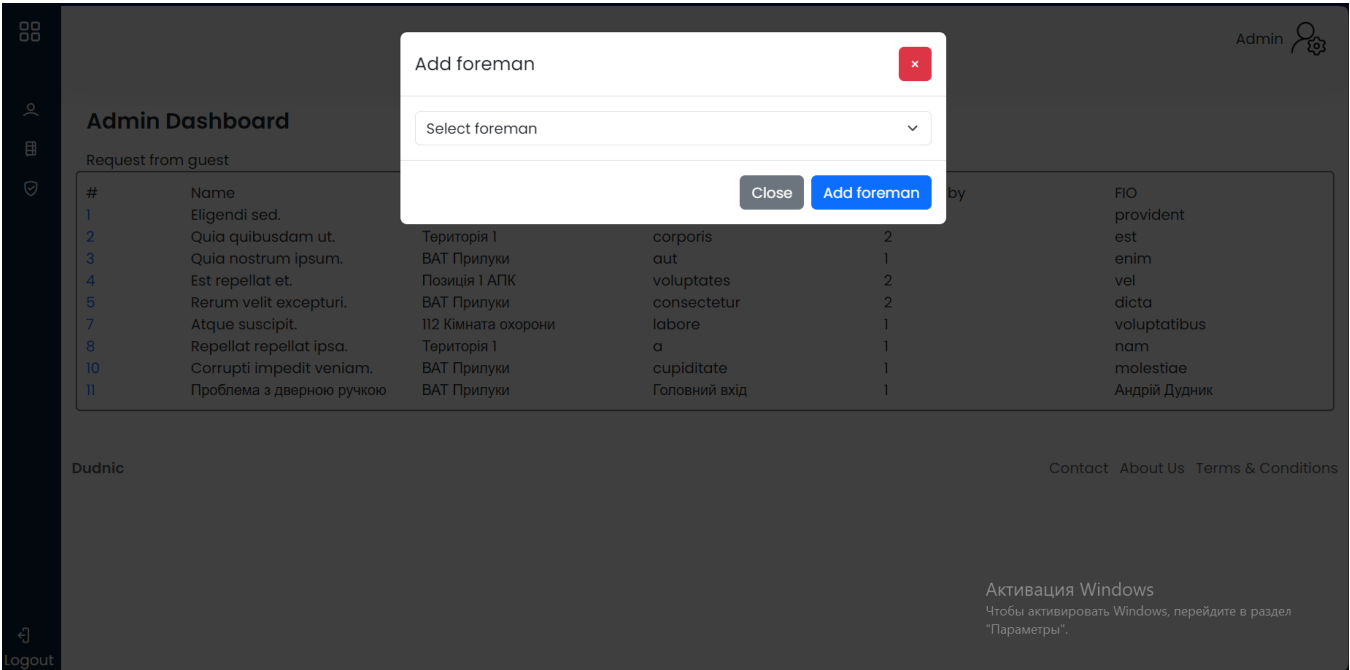


Рисунок 5.5 – Вибір підрядної організації

Коли заявка приходить до керівника підрядної організації, він вказує погодинну оплату за роботу та вибирає працівника який буде виконувати цю задачу (рис. 5.6, 5.7).

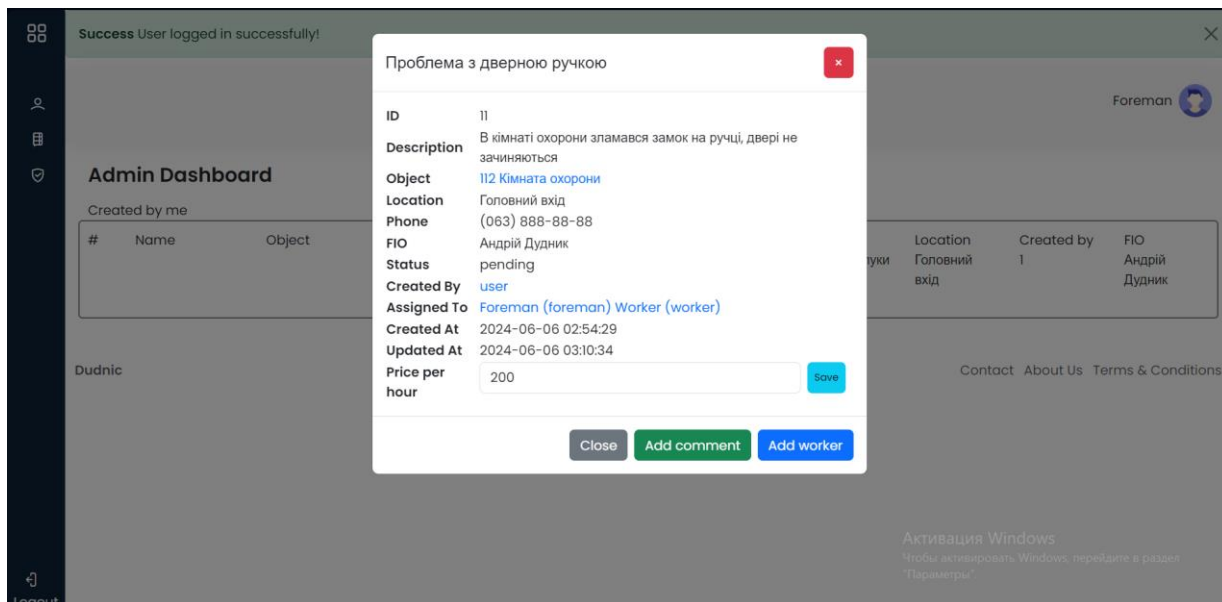


Рисунок 5.6 – Вигляд заявки у керівника підрядної організації

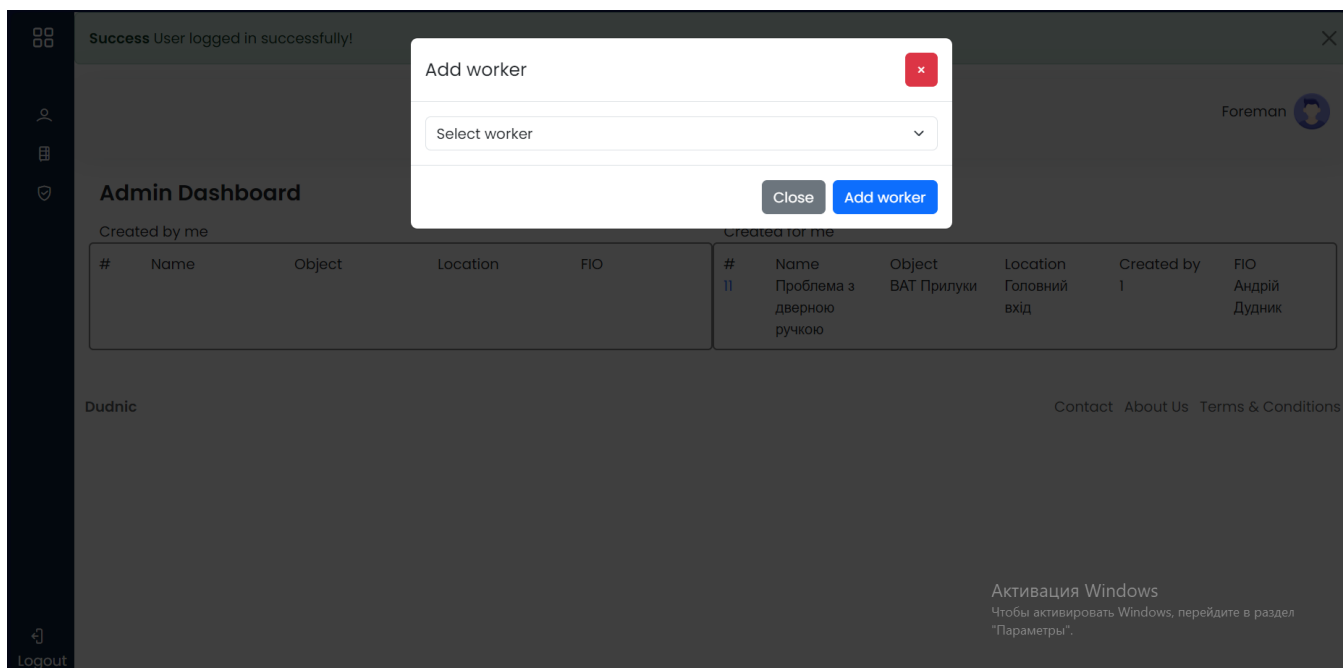


Рисунок 5.7 – Вибір працівника

Працівник, коли отримав заявку має підійти на місце виникнення проблеми та перед початком своєї роботи натиснути на кнопку “start work” для початку підрахунку часу роботи (рис. 5.8, 5.9).

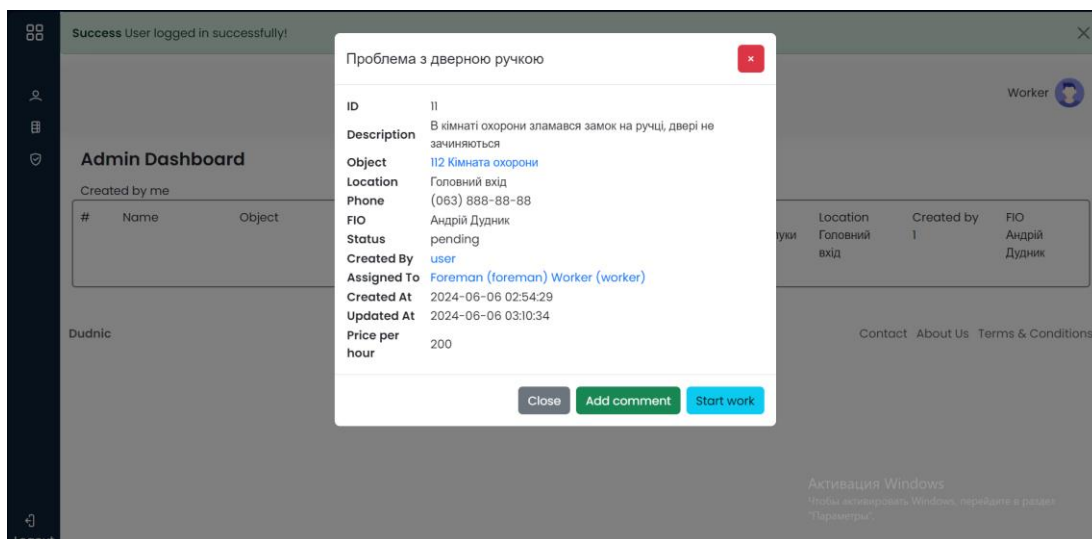


Рисунок 5.8 – Вигляд заявки у працівника

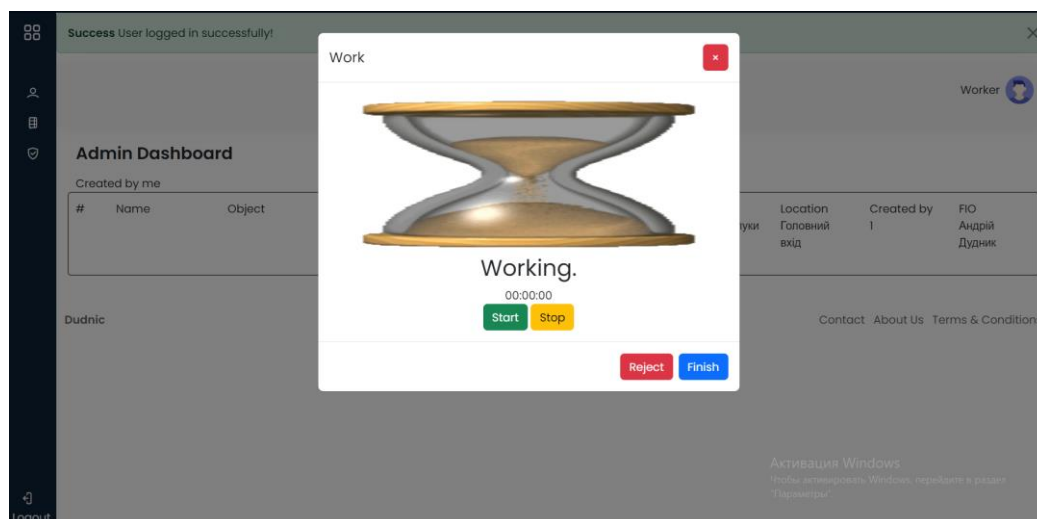


Рисунок 5.9 – Вікно початку роботи

Коли робота виконана, заявка відправляється в історію з прорахованою вартістю робіт (рис. 5.10).

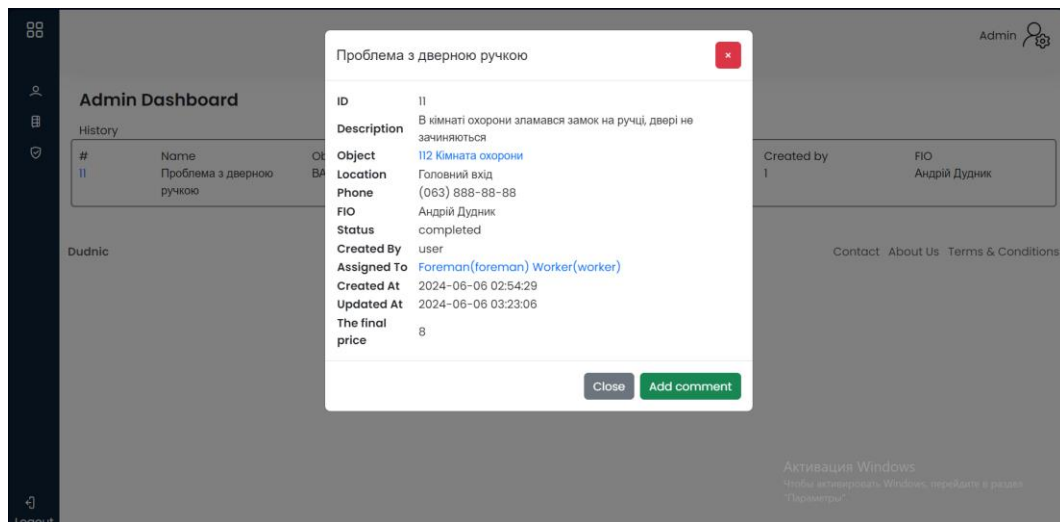


Рисунок 5.10 – Вигляд закінченої заявки

Під час експлуатації програми ми повністю перевірили її функціонал, щоб перевірити відповідність всім вимогам. У результаті тестування була зроблена заявка, котра успішно завершила робочий процес.

ВИСНОВКИ

В результаті виконання бакалаврської роботи всі завдання, які пов'язані з розробкою веб-додатка, додавання до нього основних функцій, створення декількох типів користувачів та формування правильної працездатності заявок були виконані.

Для розробки використані такі мови як: PHP, JavaScript. Проект був зроблений у середовищі розробки PhpStorm. Додаток був протестован завдяки локальному серверу OpenServer

Впровадження CAFM систем може стати гарним розвитком бізнесу у сферах обслуговування інфраструктури підприємств. Завдяки автоматизації та діджиталізації робочих процесів значно підвищується ефективність роботи. Підприємства отримують можливість знизити свої витрати на управління, підвищити оперативність, а також покращити взаємодію між своїми підрозділами та зовнішніми організаціями.

Новий погляд на управління інфраструктурою з використанням CAFM систем дозволяє інтегрувати різноманітні аспекти бізнесу в єдину надійну систему, за допомогою якої можна забезпечити кращу координацію та прийняття обґрунтованих рішень.

Розроблена система дозволяє керувати робочими процесами з управління інфраструктурою, а також зберігати усі виконані процеси в історії заявок, що може стати корисною альтернативою роботи з паперовими архівами.

Підводячи підсумки, можна сказати що розробка та впровадження систем такого типу є важливим етапом у розвитку підприємства, завдяки чому підвищується конкурентоспроможність бізнесу в умовах глобалізації та постійного посилення конкуренції.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Computer Aided Facilities Management (CAFM). URL: <https://www.wbdg.org/facilities-operations-maintenance/computer-aided-facilities-management-cafm>
2. CAFM: система автоматизації управління інфраструктурою нерухомості. URL: <http://surl.li/ueqjd>
3. Facility management software systems. Best CAFM Software Tools in 2024. URL: <https://revizto.com/en/facility-management-software/>
4. Побудова моделі системи. URL: <http://surl.li/uitpn>
5. CAFM DATA STRUCTURES: A REVIEW AND EXAMPLES. URL: https://ifpwww.ifp.uni-stuttgart.de/publications/2000/Schuerle_1586.pdf
6. Архітектура веб-додатків. URL: <http://surl.li/uiwmb>
7. Lerdorf R., Tatroe K. Programming Php. – " O'Reilly Media, Inc.", 2022.
8. Williams H. E., Lane D. Web Database Applications with PHP and MySQL: Building Effective Database-Driven Web Sites. – " O'Reilly Media, Inc.", 2022.
9. Stauffer M. Laravel: Up & Running. - "O'Reilly Media, Inc.", 2023.
10. He R. Y. Design and implementation of web based on Laravel framework //2014 International Conference on Computer Science and Electronic Technology (ICCSET 2014). – Atlantis Press, 2015. – С. 301-304.
11. Що таке Laravel? Огляд переваг для веброзробки. URL: <https://asabix.com.ua/what-is-laravel/>
12. Forta B. MySQL crash course. - Addison-Wesley Professional, 2023.
13. Бази даних. URL: <https://e5.ua/uk/junior-pm-and-ba/bazi-danih/>
14. Williams H. E., Lane D. Web Database Applications with PHP and MySQL: Building Effective Database-Driven Web Sites. – " O'Reilly Media, Inc.", 2004.
14. База даних MySQL. URL: <https://promoter.net.ua/articles/baza-danix-mysql.html>
15. Šušter I., Ranisavljević T. Optimization of MySQL database // Journal of

process management and new technologies. - 2023. - Т. 11. - №. 1-2. - С. 141-151.

16. Чому варто вивчати JavaScript? URL: <https://qagroup.com.ua/publications/why-learn-javascript/>

17. Shcherbakova M.E., Shcherbakov E.V. FUNCTIONAL FEATURES OF JAVASCRIPT-APPLICATIONS. URL: https://filelibsnu.at.ua/naukovi/vestnik_10-217-2014.pdf#page=142

18. З чого почати вивчення JavaScript. URL: <https://merge.academy/blog/z-chogho-pochati-vivchiennia-javascript>

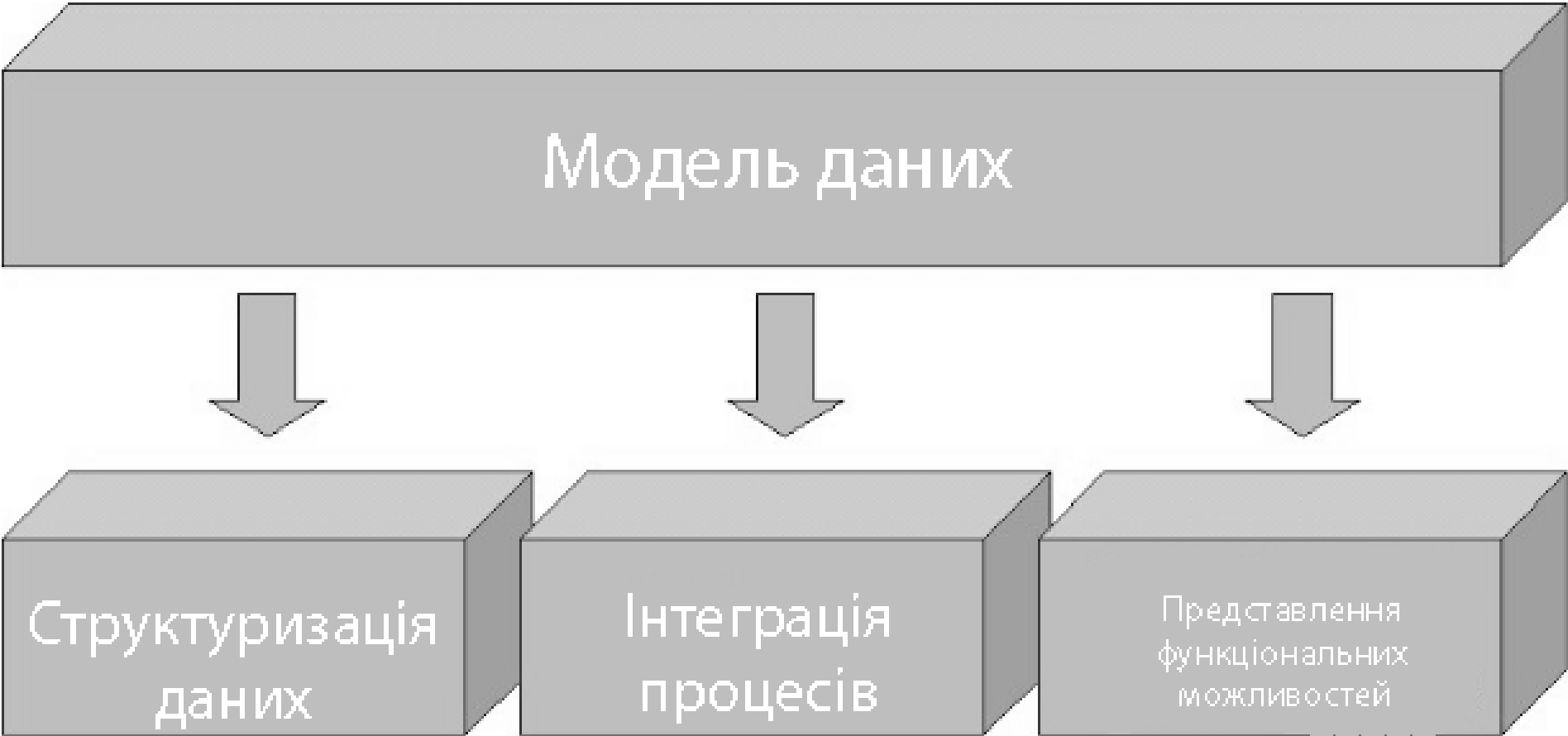
19. Abelson H., Sussman GJ Structure and Interpretation of Computer Programs: JavaScript Edition. - MIT Press, 2022.

20. Rochkind M. Expert PHP and MySQL: application design and development. – Apress, 2013.

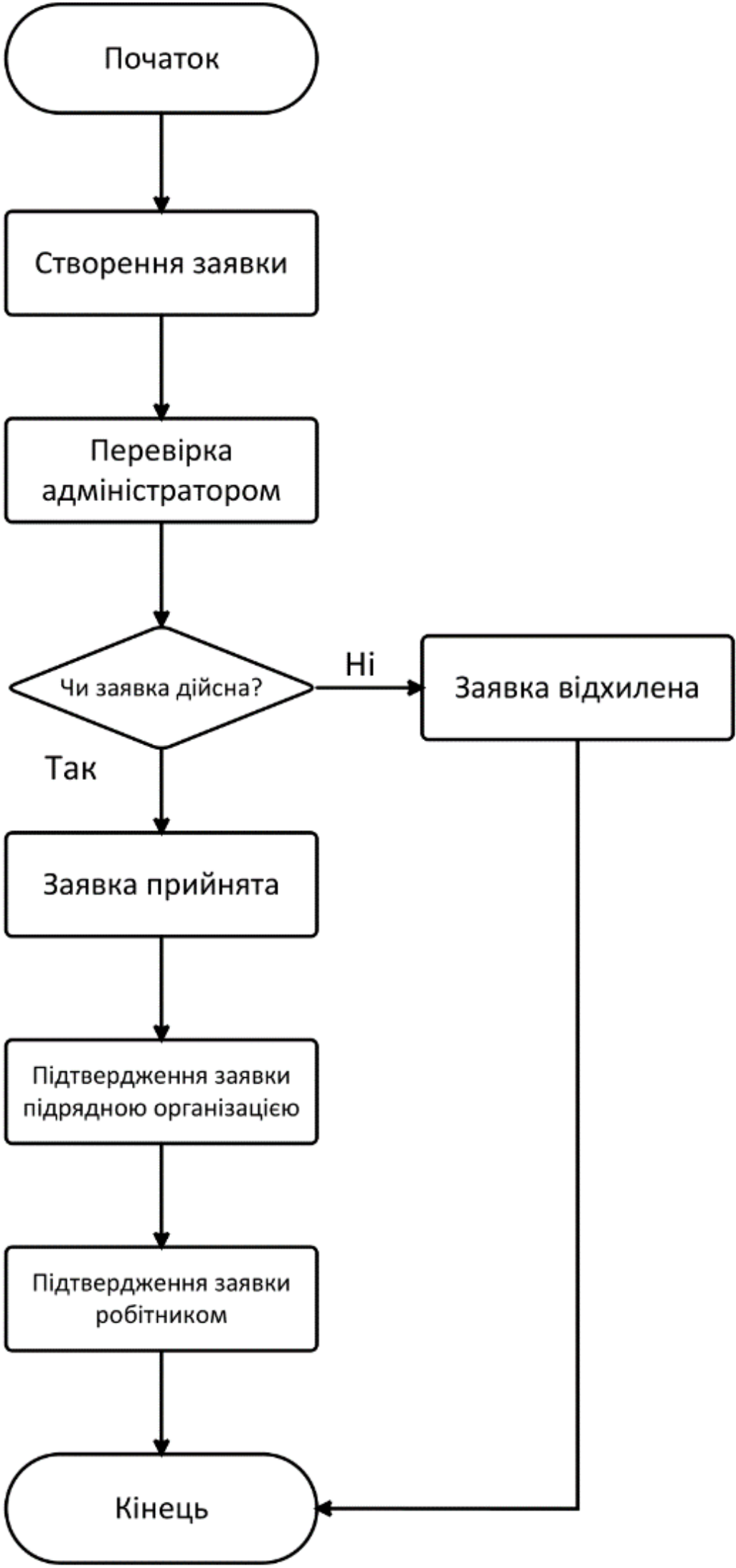
21. Database: Migrations. URL: <https://laravel.com/docs/10.x/migrations>

22. UML class diagram syntax: an empirical study of comprehension. URL: <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=fae00885887732fcf1769170d4b8514e3730e358>

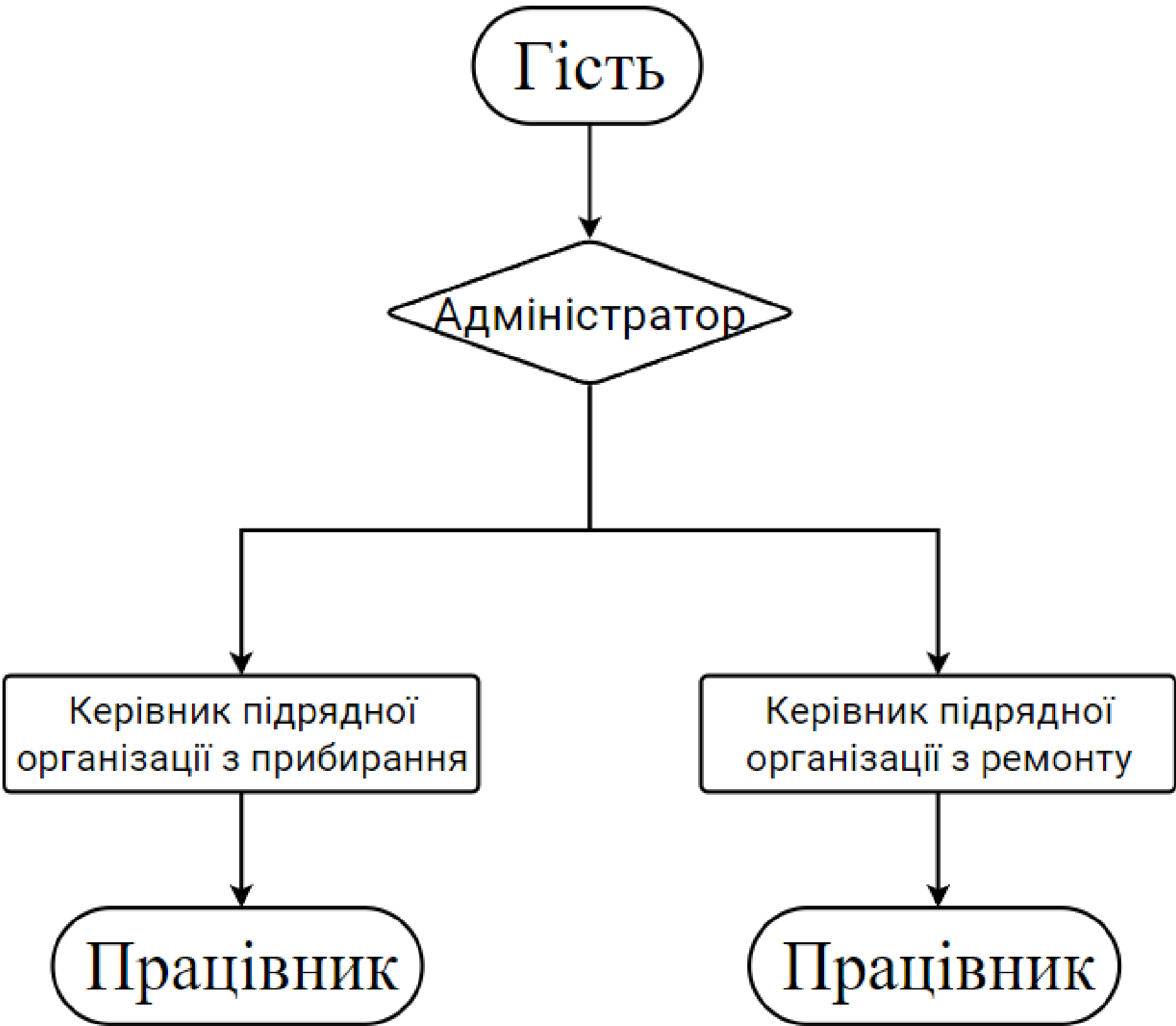
23. Web Application Testing: A Review on Techniques, Tools and State of Art. URL: <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=28115c63c146e4b77fa690e5a55175a36dacc634>



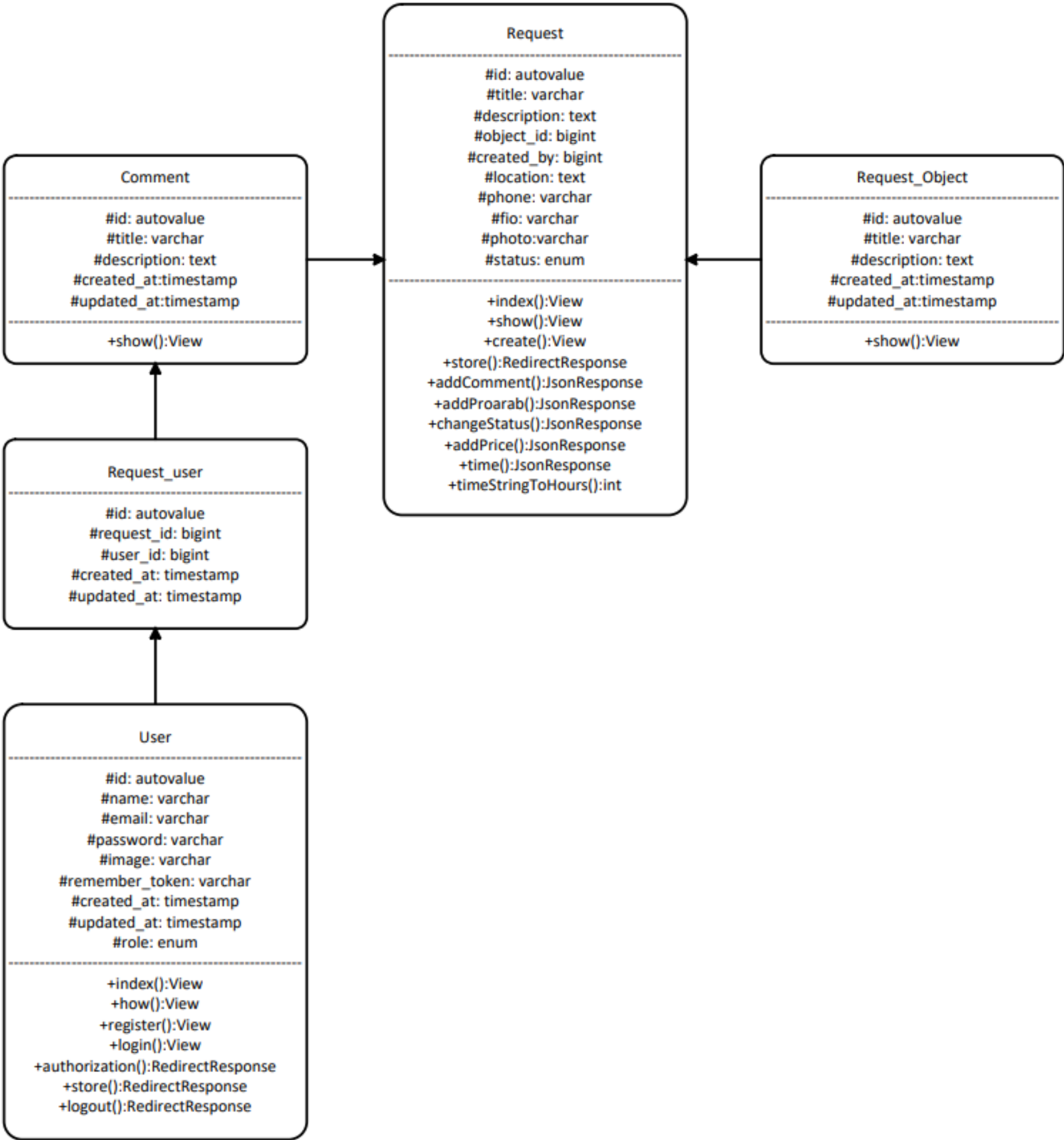
					13.02070849.00039 ПЛ1						
					Розробка САФМ системи для покращення обслуговування інфраструктури підприємства	Літ.			Маса	Масштаб	
Зам.	Лист.	№ докум.	Підп.	Дата							
Розроб.		Дудник А.В.									
Перевір.		Киричек Г.Г.									
Т. контр.											
					Основні вимоги до моделі даних	Лист. 1			Листів 1		
Н.контр.		Польська О.В.				НУ «Запорізька політехніка» КНТ-520					
Затвердж.		Кудерметов Р.К.									



					13.02070849.00039 ПЛ2				
					Розробка САФМ системи для покращення обслуговування інфраструктури підприємства Алгоритм станів заявки	Літ.		Маса	Масштаб
Зам.	Лист.	№ докум.	Підп.	Дата					
Розроб.		Дудник А.В.							
Перевір.		Киричек Г.Г.							
Т. контр.						Лист. 1		Листів 1	
						НУ «Запорізька політехніка» КНТ-520			
Н.контр.		Польська О.В.							
Затвердж.		Кудерметов Р.К.							



					13.02070849.00039 ПЛЗ					
					Розробка САФМ системи для покращення обслуговування інфраструктури підприємства Проходження заявки	Літ.		Маса	Масштаб	
Зам.	Лист.	№ докум.	Підп.	Дата						
Розроб.		Дудник А.В.								
Перевір.		Киричек Г.Г.								
Т. контр.										
						Лист. 1		Листів 1		
Н.контр.		Польська О.В.				НУ «Запорізька політехніка» КНТ-520				
Затвердж.		Кудерметов Р.К.								



					13.02070849.00039 ПЛ4					
					Розробка САФМ системи для покращення обслуговування інфраструктури підприємства Діаграма класів	Літ.		Маса	Масштаб	
Зам.	Лист.	№ докум.	Підп.	Дата						
Розроб.		Дудник А.В.								
Перевір.		Киричек Г.Г.								
Т. контр.										
						Лист. 1		Листів 1		
Н.контр.		Польська О.В.				НУ «Запорізька політехніка» КНТ-520				
Затвердж.		Кудерметов Р.К.								