Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій

КОМПЛЕКСНА МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«АЛГОРИТМИ АДАПТИВНОГО ПЛАНУВАННЯ ДЛЯ ОПТИМІЗАЦІЇ ЛОГІСТИЧНИХ ПРОЦЕСІВ. ЧАСТИНА 2. ПРОЄКТУВАННЯ ТА ВПРОВАДЖЕННЯ ВЕБЗАСТОСУНКУ ДЛЯ АВТОМАТИЗОВАНОГО ОБЛІКУ РОБОЧОГО ЧАСУ »

Виконав: студент 2 курсу, групи 1АКІТР-24м
спеціальності 174 – Автоматизація,
комп'ютерно-інтегровані технології та
робототехніка

_____ Владислав ЧИЧИРКО
(підпис)

Керівник: к.т.н., доцент каф. АІІТ

_____ Ярослав КУЛИК
(підпис)

«4» 12 2025 р.

Опонент: д.т.н., професор каф. КСУ

_____ Марія ЮХИМЧУК
(підпис)

«5» 12 2025 р.

Допущено до захисту

Зав. кафедри АІІТ

_____ Олег БІСІКАЛО
(підпис)

«6» 12 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій
Рівень вищої освіти другий (магістерський)
Галузь знань – 17 – Електроніка, автоматизація та електронні комунікації
Спеціальність – 174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка
Освітньо-професійна програма – Інтелектуальні комп'ютерні системи

ЗАТВЕРДЖУЮ
Зав. кафедри АІІТ
Олег БІСІКАЛО
"26" вересня 2025 року

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Чичирко Владиславу Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи Алгоритми адаптивного планування для оптимізації логістичних процесів. Частина 2. Проєктування та впровадження вебзастосунку для автоматизованого обліку робочого часу.
керівник роботи Кулик Ярослав Анатолійович, к.т.н., доц. каф. АІІТ
затверджені наказом **ВНТУ від 14.10.2025 №346 .**
2. Термін подання студентом роботи "10" грудня 2025 року
3. Вихідні дані до роботи: модель логістичної компанії з 25 працівниками (логісти, водії, адміністратори) та 60–80 завдань на добу тривалістю 30–180 хв. Робочі зміни тривають 8–12 годин, з максимальною активністю 10 год/добу на співробітника. Використовується 10–15 транспортних одиниць, враховуються часові вікна персоналу та динамічне коригування до 20 % завдань.
4. Зміст текстової частини: вступ, теоретичні основи та аналіз предметної області, проєктування клієнт-серверного застосунку, реалізація програмного забезпечення, тестування та оцінка ефективності, економічна частина, література.
5. Перелік ілюстративного матеріалу: модель даних системи LogiTime з основними сутностями та їх зв'язками, послідовність обробки запиту до серверної частини, архітектура серверної частини, UML-діаграма варіантів використання системи LogiTime, UML-діаграма послідовності сценарію авторизації користувача, Модель розмежування доступу користувачів до ресурсів системи на основі ролей (RBAC).

1. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
1-4	Кулик Я.А. к.т.н. доцент кафедри АІІТ		
5	Козловський В. О. – професор кафедри ЕПВМ		

2. Дата видачі завдання "19" жовтня 2025 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Вибір та узгодження теми	14.10.2025		
2	Аналіз предметної області та ринку	20.10.2025	26.10.2025	
3	Розробка алгоритмів адаптивного планування	27.10.2025	01.11.2025	
4	Формування вимог до системи та обґрунтування архітектури	02.11.2025	08.11.2025	
5	Реалізація програмного забезпечення вебсистеми	09.11.2025	17.11.2025	
6	Розробка UML-діаграм	18.11.2025	20.11.2025	
7	Створення моделі бази даних системи	21.11.2025	24.11.2025	
8	Реалізація алгоритмів адаптивного планування та тестування	25.11.2025	27.11.2025	
9	Виконання економічної частини	28.11.2025	30.11.2025	
10	Оформлення МКР та збір підписів супроводжувальних документів	01.12.2025	05.12.2025	
11	Захист МКР	15.12.2025	16.12.2025	

Студент _____ Владислав ЧИЧИРКО
(Підпис) *(Ім'я ПРІЗВИЩЕ)*

Керівник _____ Ярослав КУЛИК
(Підпис) *(Ім'я ПРІЗВИЩЕ)*

АНОТАЦІЯ

УДК 621.374.415

Чичирко В. О. Алгоритми адаптивного планування для оптимізації логістичних процесів. Частина 2. Проєктування та впровадження вебзастосунку для автоматизованого обліку робочого часу.

Магістерська кваліфікаційна робота зі спеціальності 174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка, освітня програма – Інтелектуальні комп'ютерні системи. Вінниця: ВНТУ, 2025. 133 с.

Українською мовою, 5 розділів, 50 джерел, рис.:12, табл.: 6

Метою роботи є розробка клієнт-серверного вебзастосунку для автоматизованого обліку робочого часу персоналу логістичних компаній із використанням алгоритмів адаптивного планування та сучасних технологій програмування (Laravel, MySQL, React, React Native).

У теоретичній частині розглянуто актуальність оптимізації логістичних процесів, проведено аналіз існуючих інформаційних систем управління персоналом і засобів обліку робочого часу, виявлено їхні недоліки та обґрунтовано доцільність створення нової системи.

У практичній частині виконано проєктування інформаційної структури та бази даних, описано архітектуру клієнт-серверного застосунку, реалізовано серверну частину на базі Laravel і MySQL, вебінтерфейс засобами React та мобільний застосунок у середовищі React Native. Розроблено модуль адаптивного планування з використанням методів оптимізації та евристичних алгоритмів. Проведено тестування програмного забезпечення, результати якого підтвердили ефективність запропонованого рішення.

Ілюстративна частина складається з 6 рисунків, які включають UML-діаграми та результати роботи системи.

Ключові слова: адаптивне планування, оптимізація логістики, автоматизований облік часу, вебзастосунок, Laravel, React, MySQL, підкріплювальне навчання.

ABSTRACT

Chychyrko V. O. Adaptive Planning Algorithms for Optimizing Logistics Processes. Part 2. Design and Implementation of a Web Application for Automated Employee Time Tracking.

Master's Thesis in the specialty 174 – Automation, Computer-Integrated Technologies and Robotics, Educational Program – Intelligent Computer Systems. Vinnytsia: VNTU, 2025. 133 pages.

In Ukrainian, 5 chapters, 50 references, figures: 12, tables: 6.

The purpose of the work is the development of a client-server web application for automated time tracking of logistics company personnel using adaptive planning algorithms and modern programming technologies (Laravel, MySQL, React, React Native).

The theoretical part examines the relevance of optimizing logistics processes, analyzes existing personnel management information systems and time-tracking tools, identifies their shortcomings, and justifies the feasibility of creating a new system.

In the practical part, the information structure and database were designed, the architecture of the client-server application was described, the server-side was implemented using Laravel and MySQL, the web interface was developed using React, and the mobile application was created in the React Native environment. A module for adaptive planning was also developed using optimization methods, heuristic algorithms, and reinforcement learning. Software testing was carried out, and the results confirmed the effectiveness of the proposed solution.

The illustrative part consists of 6 figures, including UML diagrams and system operation results.

Keywords: adaptive planning, logistics optimization, automated time tracking, web application, Laravel, React, MySQL, reinforcement learning.

ЗМІСТ

ВСТУП

Актуальність. У сучасних умовах логістика є однією з ключових галузей економіки, що забезпечує ефективне функціонування ланцюгів постачання. Зростання обсягів перевезень, підвищення вимог до швидкості та надійності доставки, а також ускладнення логістичних процесів зумовлюють необхідність удосконалення методів управління трудовими ресурсами. Традиційні підходи до планування робочого часу персоналу є недостатньо гнучкими та не забезпечують оперативного реагування на динамічні зміни умов роботи.

Особливої актуальності проблема набуває в умовах воєнних дій на території України, коли логістичні компанії змушені працювати в умовах нестабільності, зміни маршрутів і підвищених ризиків. У таких умовах використання алгоритмів адаптивного планування та автоматизованих інформаційних систем дозволяє оптимізувати розподіл робочого часу, підвищити ефективність логістичних процесів і забезпечити гнучке управління персоналом. Тому дослідження та розробка інформаційної системи для автоматизованого обліку й адаптивного планування робочого часу є актуальним науково-практичним завданням.

Мета роботи полягає у підвищенні ефективності управління робочим часом у логістичних компаніях шляхом комплексного підходу, що включає аналіз вимог, розробку інформаційної структури та створення вебзастосунку для автоматизованого планування й обліку робочого часу. В основі запропонованих рішень лежать алгоритми адаптивного планування, що дозволяють динамічно коригувати розподіл завдань та оптимізувати використання ресурсів у реальному часі, підвищуючи продуктивність праці та ефективність логістичних процесів загалом.

Завданням магістерської кваліфікаційної роботи є:

– проаналізувати проблеми управління робочим часом у логістичних компаніях;

– дослідити сучасні підходи та інформаційні системи планування ресурсів;

– сформувати вимоги до автоматизованої системи обліку та адаптивного планування робочого часу з урахуванням специфіки логістичних процесів;

– розробити інформаційну та архітектурну моделі системи;

– розробити алгоритми адаптивного планування для оптимізації розподілу робочого часу персоналу;

– реалізувати клієнт-серверний вебзастосунок;

– провести експериментальну перевірку та оцінити ефективність запропонованих рішень.

Об’єктом дослідження є процеси управління робочим часом у логістичних компаніях та їх оптимізація з використанням інформаційних технологій.

Предметом дослідження є алгоритми адаптивного планування та програмні засоби, що забезпечують автоматизацію контролю та розподілу робочих завдань. До предмету також входить розробка інформаційної структури, яка дозволяє інтегрувати дані про співробітників, завдання та ресурси компанії для створення ефективної системи управління робочим часом.

Методи дослідження. У процесі виконання магістерської роботи застосовано методи системного аналізу для дослідження логістичних процесів та взаємозв’язків між їх складовими, методи математичного моделювання та оптимізації для формування алгоритмів адаптивного планування, а також евристичні методи й елементи підкріплювального навчання для автоматичного коригування робочих графіків. Для реалізації програмного забезпечення використано методи програмної інженерії, об’єктно-орієнтованого проєктування, REST-орієнтовану архітектуру та експериментальні методи тестування для оцінки ефективності розробленої системи.

Науково-технічний результат роботи полягає у розробці комплексної моделі автоматизованого управління робочим часом персоналу на основі алгоритмів адаптивного планування, яка враховує специфіку логістичних процесів. На відміну від існуючих підходів, запропоноване рішення поєднує методи математичного моделювання, системного аналізу та програмної інженерії, що забезпечує можливість гнучкої адаптації системи до змін

зовнішнього середовища. Крім того, новизна полягає у поєднанні інформаційної структури для обліку з алгоритмічними методами оптимізації, що створює підґрунтя для ефективного управління трудовими ресурсами у динамічних умовах.

Практична цінність дослідження полягає у можливості впровадження розробленої системи в діяльність логістичних компаній. Запропонований вебзастосунок може використовуватися для автоматизації планування робочого часу, розподілу завдань між співробітниками, контролю виконання та оперативного реагування на зміни. Використання системи сприяє підвищенню продуктивності персоналу за рахунок раціонального розподілу навантаження, зменшенню фінансових і часових витрат, а також забезпеченню прозорості та контрольованості процесів управління.

Структура магістерської роботи визначається логікою дослідження і складається зі вступу, двох частин, висновків, списку використаних джерел та додатків. У першій частині розглядаються теоретичні засади адаптивного планування, аналізуються вимоги до системи управління робочим часом і формується інформаційна структура. У другій частині здійснюється проєктування та впровадження вебзастосунку з використанням алгоритмів адаптивного планування та подаються результати експериментальної перевірки його ефективності.

Апробація: Результати роботи було представлено на Всеукраїнській науково-практичній Інтернет-конференції студентів, аспірантів та молодих науковців «МОЛОДЬ В НАУЦІ: ДОСЛІДЖЕННЯ, ПРОБЛЕМИ, ПЕРСПЕКТИВИ», 2025 [1].

1 ТЕОРЕТИЧНІ ОСНОВИ ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Розділ присвячено вивченню теоретичних засад та аналізу предметної області, що є основою для подальшої розробки інформаційних систем та оптимізації процесів у конкретній сфері діяльності. Розглянуто ключові поняття, методи та підходи, які дозволяють систематизувати знання, оцінити сучасний стан галузі та визначити потреби та вимоги до розроблюваних рішень. Аналіз предметної області дозволяє виявити основні проблеми, існуючі рішення та їх недоліки, що створює підґрунтя для формування ефективних і сучасних методів та інструментів у досліджуваній сфері.

1.1 Актуальність автоматизації обліку робочого часу в логістиці

Автоматизація обліку робочого часу на сучасному етапі розвитку суспільства набуває особливого значення та стає важливою складовою управлінських процесів на підприємствах різних галузей. Зростання обсягів інформації, ускладнення бізнес-процесів і необхідність підвищення продуктивності зумовлюють потребу у впровадженні сучасних інформаційних технологій, здатних забезпечити точний і своєчасний контроль використання робочого часу.

По-перше, в умовах цифрової трансформації підприємства прагнуть досягти максимальної ефективності за мінімальних витрат. Ручні методи обліку не відповідають сучасним вимогам через високий рівень помилок, значні часові витрати та складність формування аналітичної звітності [2]. Автоматизовані системи забезпечують отримання достовірних даних у режимі реального часу, що сприяє обґрунтованому прийняттю управлінських рішень і підвищенню продуктивності персоналу.

По-друге, соціально-економічна ситуація в Україні, ускладнена воєнними діями та пов’язаними з ними ризиками, робить питання оптимізації ресурсів

особливо актуальним. Підприємства стикаються з кадровим дефіцитом, нерівномірним завантаженням працівників та потребою швидкого реагування на зміни ринку. У таких умовах автоматизація процесів обліку робочого часу дозволяє знизити навантаження на адміністративний персонал, оперативно відслідковувати ефективність використання робочої сили та забезпечувати адаптивність організації до зовнішніх викликів.

По-третє, чинне трудове законодавство України та міжнародні стандарти регламентують численні аспекти використання робочого часу: від дотримання норм робочого дня до обліку понаднормових годин, відпусток та лікарняних. Виконання цих вимог у ручному режимі створює значні труднощі та підвищує ризики виникнення юридичних колізій [3]. Автоматизовані системи обліку забезпечують точність документування, прозорість кадрових процедур і зниження кількості порушень, що є критично важливим для будь-якого сучасного підприємства.

Особливу увагу слід приділити оптимізації управлінських і виробничих процесів. Автоматизація дозволяє усунути дублювання даних, скоротити кількість рутинних операцій, зменшити залежність від людського фактору, а також забезпечити інтеграцію з іншими інформаційними системами підприємства. Це створює умови для комплексного управління персоналом, де дані про відпрацьований час можуть використовуватися не лише для обліку заробітної плати, але й для планування змін, управління транспортними та логістичними процесами, прогнозування завантаженості ресурсів.

Важливим є й аспект гнучкості. Сучасні компанії потребують систем, які дозволяють швидко адаптуватися до нових умов: змінювати графіки роботи, впроваджувати віддалений режим, обліковувати понаднормову працю та відрядження. Автоматизовані рішення здатні забезпечити не лише точний облік, а й інструменти аналітики, які допомагають прогнозувати потреби у персоналі, зменшувати витрати на оплату простоїв і підвищувати загальну конкурентоспроможність компанії.

Таким чином, актуальність автоматизації обліку робочого часу визначається низкою чинників: необхідністю підвищення продуктивності праці, зниженням адміністративних витрат, забезпеченням прозорості кадрових процесів, дотриманням законодавчих норм та адаптацією до умов економічної та соціальної нестабільності. Автоматизація стає ключовим інструментом оптимізації бізнес-процесів, що робить вирішення цього завдання пріоритетним для сучасних підприємств, особливо у сфері логістики, де ефективне використання часу персоналу безпосередньо впливає на якість послуг і рівень прибутковості.

1.2 Аналіз існуючих програмних рішень та інформаційних систем управління персоналом

На сучасному ринку українських логістичних компаній для обліку робочого часу та управління персоналом найбільш поширеними є локальні системи та застаріле програмне забезпечення, серед яких варто виділити Soft4Trans, BAS ERP, а також більш сучасні та адаптовані під українські реалії рішення, такі як LogiTime та 4Logist [4]. Кожна із цих платформ має власні функціональні особливості, переваги та обмеження, що визначає їх ефективність та доцільність використання у логістичних підприємствах.

Система Soft4Trans є українським програмним продуктом, що поєднує функціонал TMS, ERP та CRM для комплексної автоматизації транспортно-логістичних процесів. Вона забезпечує управління перевезеннями, замовленнями, маршрутами, облік витрат і контроль виконання рейсів у режимі реального часу. Soft4Trans адаптована під українське законодавство, підтримує інтеграцію з бухгалтерськими системами та GPS-сервісами, що робить її універсальним рішенням для підприємств логістичної галузі. Система сприяє підвищенню прозорості бізнес-процесів, оптимізації маршрутів і зниженню операційних витрат.

Разом із тим, Soft4Trans (рисунок 1.1) має низку обмежень, що ускладнюють її використання на малих і середніх підприємствах. Впровадження системи потребує залучення технічних спеціалістів, що підвищує витрати на налаштування та інтеграцію. Крім того, вартість ліцензій та технічної підтримки залишається досить високою, що знижує економічну доцільність використання продукту для невеликих компаній. Оновлення системи й розширення функціоналу часто супроводжуються потребою в ручному доопрацюванні модулів, що може призвести до несумісності з попередніми версіями програмного забезпечення [5].

Також серед недоліків варто відзначити обмежену масштабованість і не завжди інтуїтивно зрозумілий інтерфейс, який потребує часу для адаптації користувачів. Система орієнтована переважно на середні та великі логістичні підприємства, що мають достатні фінансові та кадрові ресурси для впровадження й підтримки. Таким чином, попри свою функціональність, Soft4Trans поступається за гнучкістю і швидкістю інтеграції більш легким і практичним рішенням, таким як LogiTime, які краще підходять для оперативного управління робочим часом і процесами в малих логістичних компаніях.

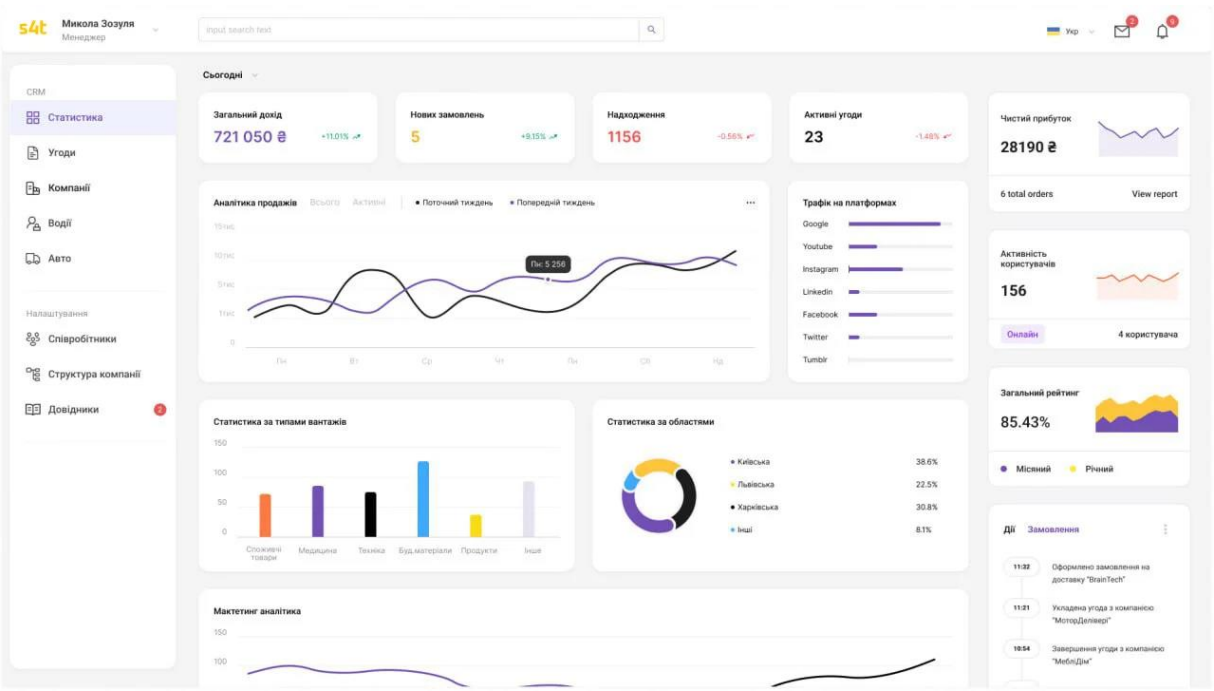


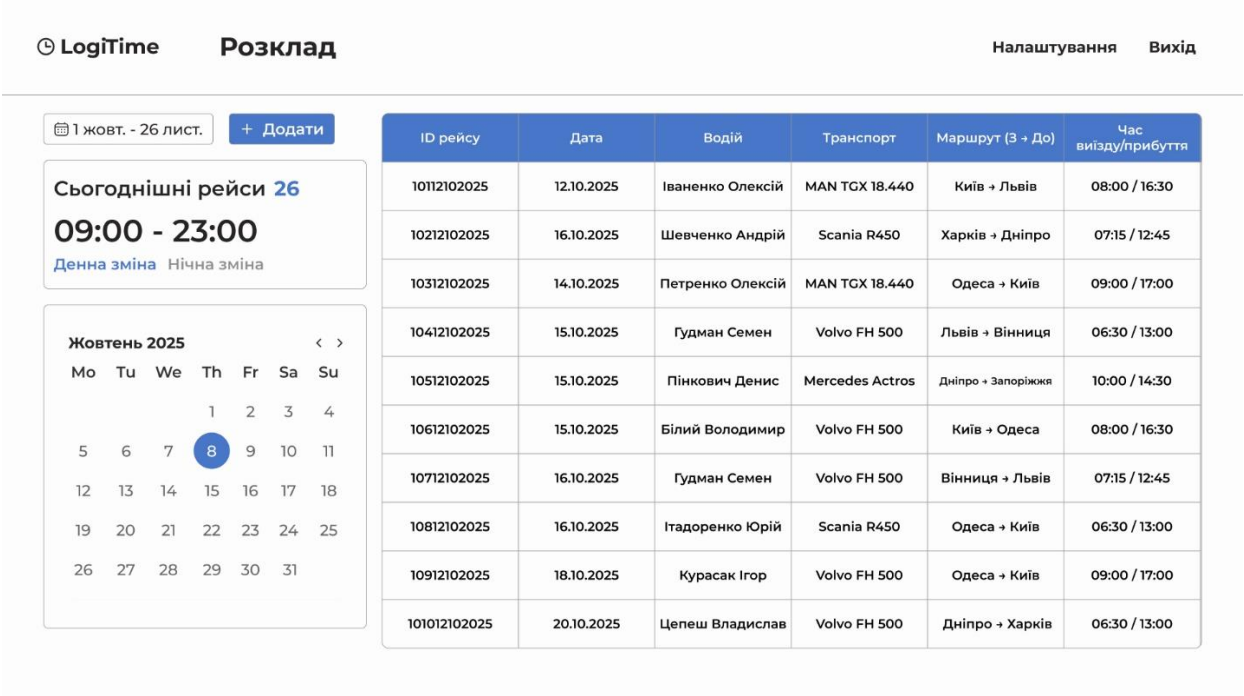
Рисунок 1.1 – Демонстрація Soft4trans

Система BAS ERP (Business Automation Software) призначена для автоматизації комплексних бізнес-процесів підприємства, зокрема обліку робочого часу, контролю кадрових процесів, управління фінансами та ресурсами компанії. Особливістю BAS є гнучка структура, яка дозволяє налаштовувати систему під потреби конкретного підприємства, автоматизуючи логістичні процеси, планування маршрутів, управління транспортом та контроль виконання завдань персоналом. До переваг системи належить можливість комплексного ведення бізнес-процесів у межах однієї платформи та інтеграція з існуючими локальними системами. Водночас BAS потребує залучення кваліфікованих спеціалістів для впровадження нових модулів, складна у інтеграції з іншими програмними продуктами та обмежує доступ до аналітичних даних без додаткових налаштувань. Тому BAS доцільна для середніх та великих компаній із базовими або стандартними логістичними процесами, проте не завжди оптимальна для малих підприємств, особливо за відсутності підтримки систем [6].

Система Soft4Trans є комплексним українським рішенням, що поєднує функціонал TMS/ERP/CRM і орієнтована на управління транспортними перевезеннями, маршрутами, замовленнями та контролем завантаженості транспорту. Вона дозволяє автоматизувати ключові логістичні процеси, забезпечує прозорість витрат та контроль виконання рейсів у реальному часі. Основними перевагами є адаптація під українські реалії, інтеграція з бухгалтерськими та GPS-системами і можливість комплексного управління операціями [7]. Проте Soft4Trans має низку недоліків: складне впровадження, високі витрати на ліцензії та супровід, обмежена масштабованість і складність освоєння інтерфейсу для нових користувачів. Таким чином, система більше підходить для середніх і великих логістичних компаній, тоді як для малих підприємств вона може бути громіздкою та дорогою.

Сучасні українські локальні рішення, такі як 4Logist і LogiTime, орієнтовані на середні та малі логістичні компанії та позиціонуються як гнучкі доповнення до вже наявних систем, включно з BAS ERP і Soft4Trans. 4Logist

дозволяє ефективно управляти замовленнями, маршрутами та транспортом, забезпечує планування перевезень і контроль виконання завдань у режимі реального часу, проте має обмежений функціонал у порівнянні з ERP-системами. LogiTime (рисунок 1.2) вирізняється простотою впровадження, швидкістю інтеграції з локальними платформами, гнучкістю налаштувань та масштабованістю, що робить її оптимальним рішенням для компаній із обмеженими технічними ресурсами. Вона враховує особливості українського законодавства та стандарти документообігу, забезпечуючи точний і ефективний контроль робочого часу та автоматизацію рутинних кадрових процесів.



ID рейсу	Дата	Водій	Транспорт	Маршрут (З → До)	Час виїзду/прибуття
10112102025	12.10.2025	Іваненко Олексій	MAN TGX 18.440	Київ → Львів	08:00 / 16:30
10212102025	16.10.2025	Шевченко Андрій	Scania R450	Харків → Дніпро	07:15 / 12:45
10312102025	14.10.2025	Петренко Олексій	MAN TGX 18.440	Одеса → Київ	09:00 / 17:00
10412102025	15.10.2025	Гудман Семен	Volvo FH 500	Львів → Вінниця	06:30 / 13:00
10512102025	15.10.2025	Пінкович Денис	Mercedes Actros	Дніпро → Запоріжжя	10:00 / 14:30
10612102025	15.10.2025	Білий Володимир	Volvo FH 500	Київ → Одеса	08:00 / 16:30
10712102025	16.10.2025	Гудман Семен	Volvo FH 500	Вінниця → Львів	07:15 / 12:45
10812102025	16.10.2025	Ітадоренко Юрій	Scania R450	Одеса → Київ	06:30 / 13:00
10912102025	18.10.2025	Курасак Ігор	Volvo FH 500	Одеса → Київ	09:00 / 17:00
101012102025	20.10.2025	Цепеш Владислав	Volvo FH 500	Дніпро → Харків	06:30 / 13:00

Рисунок 1.2 – Демонстрація LogiTime

Ще одним прикладом локальної системи, спеціально розробленої для управління логістичними процесами в Україні, є 4Logist. Вона орієнтована на планування перевезень, управління замовленнями та маршрутами, а також моніторинг транспортних операцій у режимі реального часу [8]. Програмне забезпечення надає базові інструменти для аналізу ефективності логістичних процесів та оптимізації маршрутів доставки.

Водночас система має низку обмежень. Функціонал 4Logist (рисунок 1.3) є відносно вузьким порівняно з комплексними ERP-рішеннями, що ускладнює інтеграцію з іншими корпоративними платформами та обмежує автоматизацію складніших бізнес-процесів. Крім того, система не завжди дозволяє гнучко адаптувати процеси під специфіку підприємства без додаткових налаштувань або доопрацювань [9]. Аналітичні можливості також обмежені: для отримання детальних показників ефективності, витрат або робочого часу часто потрібні сторонні інструменти.

Через такі фактори масштабування системи для середніх і великих компаній може вимагати значних ресурсів, хоча для малих підприємств 4Logist залишається прийнятним рішенням завдяки простоті впровадження та базовій функціональності.

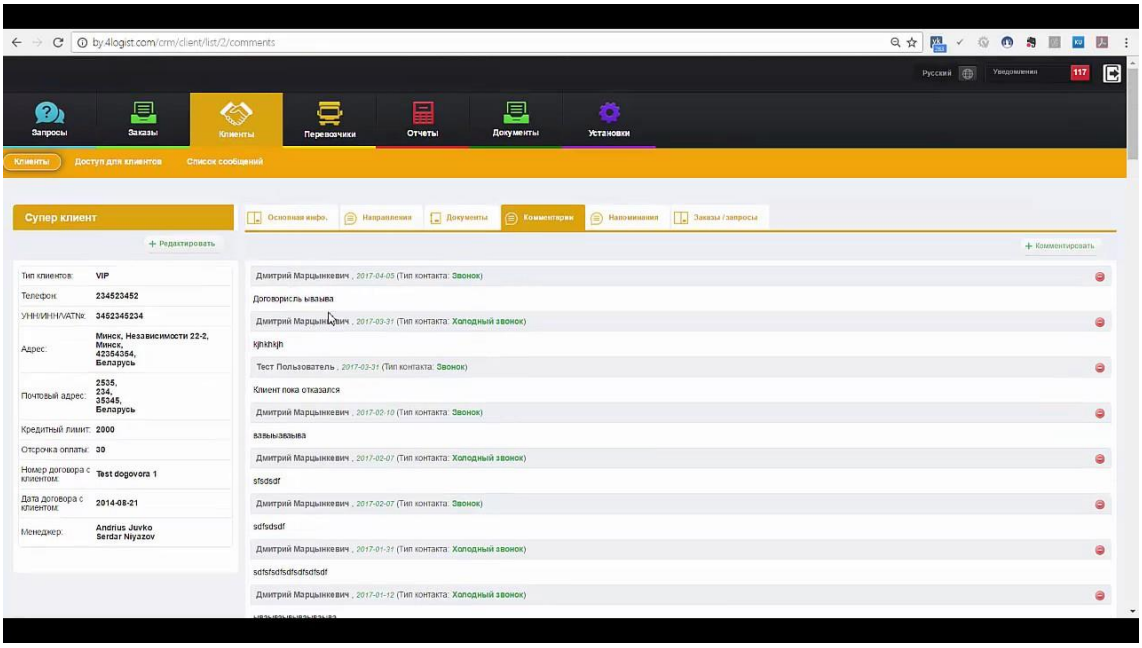


Рисунок 1.3 – Демонстрація 4Logist

Аналіз локальних та сучасних систем обліку робочого часу та управління логістикою показує, що традиційні платформи мають складність впровадження, потребують кваліфікованих фахівців і не завжди підходять малим та середнім підприємствам через високу вартість і обмежену гнучкість [10-12]. Сучасні локальні рішення, зокрема LogiTime, забезпечують просте впровадження,

швидку інтеграцію та оптимізацію управління персоналом і логістичними процесами. Порівняно з 4Logist, LogiTime вирізняється гнучкістю налаштувань, розширеними аналітичними можливостями та кращою масштабованістю, що дозволяє ефективніше адаптувати систему під потреби підприємства.

Таблиця 1.1 - Порівняння локальних систем обліку робочого часу

Система	Основні функції	Переваги	Недоліки	Актуальність для логістичних компаній
Soft4Trans	Управління замовленнями, маршрутами в режимі реального часу	Адаптована під українські реалії, інтеграція з бухгалтерією	Складне впровадження, потреба у технічних спеціалістах, висока вартість ліцензій	Висока для середніх і великих компаній, менш підходить для малих
BAS ERP	Комплексна автоматизація бізнес-процесів, включно з логістикою	Гнучкі можливості, знайомий персонал	Потребує спеціалістів, складність швидкого впровадження	Середня, підходить для компаній з базовими процесами
4Logist	Управління логістичними процесами, замовленнями та маршрутами	Адаптована під українські реалії, інтегрується з іншими системами	Обмежений функціонал, складність масштабування та аналітики	Середня, доцільна для малих логістичних компаній
LogiTime	Облік часу, інтеграція з наявним ПЗ, простота впровадження	Швидка інтеграція, економія на розробці, адаптовано під українські реалії	Обмежений функціонал у порівнянні з ERP	Висока актуальність для середніх і малих логістичних компаній

У Таблиці 1.1 наведено порівняння основних локальних систем обліку робочого часу та управління логістичними процесами, що використовуються на

українському ринку. Для кожної системи зазначено ключові функції, переваги, недоліки та актуальність застосування у логістичних компаніях. Як видно з аналізу, системи BAS ERP і Soft4Trans забезпечують комплексну автоматизацію бізнес-процесів та управління перевезеннями, однак їх впровадження є складним і потребує залучення спеціалістів [13]. 4Logist дозволяє ефективно контролювати замовлення та маршрути, адаптована під українські реалії, проте має обмежений функціонал у порівнянні з ERP-системами. Водночас LogiTime вирізняється простотою інтеграції, швидкістю впровадження та гнучкістю налаштувань, що робить її оптимальним рішенням для середніх і малих логістичних компаній.

1.3 Вимоги до сучасних клієнт-серверних застосунків у сфері планування робочого часу

На сучасному етапі розвитку інформаційних технологій клієнт-серверні застосунки виступають ключовим інструментом для організації та контролю робочого часу в логістичних компаніях. Вони забезпечують централізоване управління даними, оперативний обмін інформацією між учасниками бізнес-процесів та створюють прозору систему моніторингу діяльності персоналу.

Актуальність таких рішень визначається кількома чинниками. По-перше, логістичні компанії працюють в умовах високої динаміки та постійної зміни зовнішніх факторів (трафік, маршрути, затримки). Це вимагає гнучких інструментів, здатних швидко обробляти дані та забезпечувати їх доступність у режимі реального часу [14]. По-друге, ефективність роботи водіїв, диспетчерів та логістів напряму залежить від чіткого планування та точного логування робочого часу. По-третє, зростаюча конкуренція на ринку послуг стимулює підприємства впроваджувати сучасні клієнт-серверні рішення, що мінімізують людський фактор і підвищують якість управлінських рішень.

Таким чином, вимоги до клієнт-серверних застосунків у сфері планування робочого часу формуються на основі потреб бізнесу в надійності, масштабованості, гнучкості налаштувань та інтегрованості з ключовими

процесами компанії. У подальших підпунктах розглянуто основні технічні та функціональні характеристики, яким повинні відповідати сучасні програмні рішення цього класу.

1.3.1 Архітектурна надійність і масштабованість

Клієнт-серверний застосунок має базуватися на багаторівневій архітектурі, яка забезпечує стабільність роботи навіть при високих навантаженнях. Система повинна підтримувати горизонтальне та вертикальне масштабування, що дозволяє збільшувати кількість користувачів і обсяг даних без зниження продуктивності. Важливою вимогою є використання сучасних технологій серверної частини (наприклад, Microsoft SQL Server, PostgreSQL або Oracle) з можливістю балансування навантаження.

1.3.2 Висока продуктивність та швидкість обробки даних

Операції з планування робочого часу мають виконуватися в режимі реального часу. Це означає миттєве відображення змін у графіках, автоматичне перерахування навантажень на працівників та швидке реагування на відхилення. Від системи очікується оптимізація запитів до бази даних та ефективне використання оперативної пам’яті для обробки великих обсягів інформації.

1.3.3 Інтеграція між логістом та водієм

Система автоматизованого планування повинна забезпечувати безперервну інтеграцію між логістом і водієм у режимі реального часу. Це означає, що логіст має можливість відслідковувати маршрут руху транспортного засобу, контролювати фактичний час відправлення та прибуття, а також оперативно реагувати на зміни дорожньої ситуації чи відхилення від графіка. Водій, у свою чергу, отримує актуальні завдання, оновлення маршруту та

повідомлення безпосередньо в застосунок [15]. Такий підхід забезпечує максимально точне логування робочого часу, зменшує ризики втрат і створює прозору систему контролю для всіх учасників логістичного процесу.

1.3.4 Модульність та гнучкість налаштувань

Програмне забезпечення повинно мати модульну структуру, що дозволяє розширювати його функціональність залежно від потреб підприємства. Наприклад, до базової системи можна підключати модуль детального аналізу часу виконання рейсів або модуль оцінки ефективності водіїв. Гнучкі налаштування мають забезпечувати адаптацію до специфіки бізнес-процесів: від контролю виконання маршруту до автоматичного розрахунку відхилень у робочому часі. Це дозволяє уникнути жорсткої прив’язки до універсальних, але часто малоефективних рішень, і зробити систему максимально наближеною до реальних умов логістичної діяльності.

1.3.5 Автоматизація планування

Застосунок має підтримувати алгоритми автоматичного розподілу завдань і формування графіків роботи. Це передбачає використання адаптивних методів планування, які враховують навантаження на персонал, сезонні коливання, обмеження трудового законодавства та непередбачені зміни, наприклад, відсутність працівника через хворобу або відпустку. Система повинна не лише мінімізувати ручні операції та скорочувати час на створення розкладів, а й забезпечувати гнучкість у внесенні змін у реальному часі, автоматично коригуючи графіки відповідно до нових умов [16].

Крім того, застосунок може передбачати пріоритетизацію завдань, оптимізацію робочого навантаження між співробітниками та попередження конфліктів у розкладі. Інтеграція з календарними сервісами та мобільними додатками дозволяє працівникам оперативно отримувати оновлену інформацію

про свої зміни, а керівникам — контролювати виконання завдань та ефективність роботи персоналу. Такий підхід значно підвищує продуктивність, знижує ризик помилок при плануванні та забезпечує більш ефективне використання ресурсів компанії.

1.3.6 Контроль і прозорість обліку

Функціонал має забезпечувати точне фіксування робочого часу з можливістю відстеження фактичної присутності працівників. Це включає інтеграцію з біометричними системами контролю доступу, мобільними застосунками чи електронними картками. Дані повинні зберігатися у захищеній базі та бути доступними для аналітики й перевірок.

1.3.7 Аналітика та звітність

Застосунок повинен формувати багаторівневу звітність: від індивідуальних показників співробітників до узагальненої статистики для керівництва. Аналітичні модулі мають включати візуалізацію даних у вигляді графіків і діаграм, що дозволяє швидко оцінювати стан використання трудових ресурсів та приймати управлінські рішення.

1.3.8 Інформаційна безпека

Оскільки система працює з персональними даними співробітників, вона повинна відповідати вимогам законодавства України та міжнародним стандартам захисту даних (GDPR, ISO/IEC 27001). Передбачаються механізми багаторівневої автентифікації, контроль доступу, аудит дій користувачів та регулярне резервне копіювання. Додатково система має забезпечувати шифрування даних під час передачі та зберігання, використання захищених каналів зв’язку, а також наявність політик управління інцидентами

інформаційної безпеки. Важливим аспектом є й обмеження доступу відповідно до ролей, що мінімізує ризики несанкціонованої обробки даних та підвищує загальний рівень захищеності інформаційного середовища.

1.4 Порівняння методів і технологій реалізації клієнт-серверних систем

У сучасних умовах розвитку інформаційних технологій клієнт-серверна архітектура є базовим підходом до побудови вебзастосунків, які забезпечують багатокористувацький доступ, централізоване управління даними та гнучке масштабування. Вона лежить в основі більшості сучасних корпоративних систем, включаючи ERP-, CRM- та SCM-рішення, а також систем управління логістичними процесами. Її ключова перевага полягає в тому, що основні обчислювальні ресурси, механізми зберігання інформації та бізнес-логіка зосереджені на сервері, тоді як клієнтська частина відповідає лише за відображення інформації та взаємодію з користувачем [17].

У системах, що реалізують алгоритми адаптивного планування у логістичних процесах, правильний вибір архітектури та технологічного стеку є визначальним чинником ефективності функціонування. Такі системи повинні обробляти великі обсяги динамічних даних у реальному часі: оновлення графіків, зміни маршрутів, реєстрацію робочого часу співробітників, аналітику навантаження тощо. З огляду на це, архітектура має бути не лише продуктивною, але й адаптивною, тобто здатною до оперативного реагування на зміни умов роботи — наприклад, перерозподіл ресурсів при зміні кількості активних співробітників або транспортних одиниць.

Клієнт-серверна архітектура дозволяє реалізувати розподіл навантаження між різними рівнями системи, що особливо важливо у сфері логістики, де кількість користувачів може варіюватися залежно від часу доби або сезону. Серверна частина може виконувати складні алгоритми планування, оптимізації маршрутів чи прогнозування навантаження, тоді як клієнтська частина — лише отримувати результати та відображати їх у зручній формі. Це не тільки знижує

навантаження на клієнтські пристрої, а й спрощує підтримку системи, оскільки всі оновлення і вдосконалення виконуються централізовано на сервері.

Крім того, клієнт-серверна модель є фундаментом для побудови розподілених інформаційних систем, у яких клієнти можуть підключатися до одного чи кількох серверів з будь-якої точки світу. Це відкриває можливість інтеграції між різними підрозділами компанії, філіями чи транспортними вузлами, що робить її особливо актуальною для логістичних організацій із географічно розподіленою структурою.

Сучасний розвиток вебтехнологій призвів до появи нових варіацій клієнт-серверної архітектури, таких як багаторівневі та мікросервісні системи, які забезпечують ще більшу гнучкість і масштабованість. Використання REST API, GraphQL або WebSocket-технологій дозволяє реалізувати асинхронну взаємодію між клієнтом і сервером, завдяки чому дані оновлюються у реальному часі без перезавантаження сторінки. Це надзвичайно важливо для адаптивних систем, де планування та облік робочого часу повинні змінюватися миттєво при виникненні непередбачуваних подій (наприклад, затримки транспорту або хвороби співробітника) [18].

Ще однією перевагою клієнт-серверної архітектури є можливість реалізації централізованої безпеки та контролю доступу. Сервер може здійснювати автентифікацію користувачів, шифрування даних і моніторинг активності, що суттєво підвищує рівень захищеності системи. У логістичних компаніях це критично важливо, адже обробляються дані персоналу, маршрути транспорту, фінансова інформація та аналітика.

Вибір конкретної архітектури та технологічного підходу визначається рядом факторів: кількістю користувачів, обсягом даних, частотою оновлення інформації, потребою в інтеграції з іншими системами (наприклад, бухгалтерськими або ERP-рішеннями), а також вимогами до безпеки й швидкодії. Саме тому етап аналізу технологій і архітектурних рішень є невід’ємною складовою проєктування будь-якої сучасної інформаційної системи.

Таким чином, клієнт-серверна архітектура є найбільш універсальним і перевіреним підходом для створення адаптивних логістичних систем, оскільки забезпечує баланс між стабільністю, продуктивністю, безпекою та можливістю масштабування. Вона дає змогу побудувати гнучку інфраструктуру, здатну обслуговувати велику кількість користувачів і швидко реагувати на зміни у бізнес-процесах.

1.4.1 Архітектурні підходи до побудови клієнт-серверних систем

У сучасних умовах розвитку інформаційних технологій клієнт-серверна архітектура виступає основною парадигмою побудови вебзастосунків, що забезпечують багатокористувацький доступ до даних, централізоване управління інформаційними потоками та високу масштабованість. Вона є базовим елементом для створення інтегрованих інформаційних систем у логістиці, де критично важливою є синхронна робота різних користувачів — логістів, водіїв, диспетчерів, менеджерів і аналітиків.

Клієнт-серверна модель визначає логіку взаємодії між двома основними компонентами: клієнтом, який ініціює запити до системи (через вебінтерфейс або мобільний застосунок), і сервером, який обробляє запити, виконує бізнес-логіку та повертає результати у вигляді структурованих даних. Така модель дозволяє розподілити навантаження між клієнтською та серверною частинами, що є важливим для систем з інтенсивним обміном даними — наприклад, у логістиці, де відбувається постійне оновлення інформації про маршрути, завдання, робочий час персоналу та транспортні ресурси.

Залежно від складності проєкту, кількості користувачів та вимог до інтеграції, виділяють кілька основних архітектурних підходів до побудови клієнт-серверних систем:

Двохрівнева архітектура (thin client – thick server)

У цій моделі клієнтська частина відповідає лише за відображення інтерфейсу та взаємодію з користувачем, тоді як більшість бізнес-логіки і

обробка даних зосереджені на сервері. Такий підхід забезпечує централізований контроль і спрощує адміністрування, але має обмеження у масштабованості, адже зростання кількості користувачів збільшує навантаження на сервер. Ця архітектура часто використовується у невеликих локальних системах управління або у внутрішніх корпоративних рішеннях, де кількість користувачів незначна, а вимоги до швидкодії помірні [19].

Трирівнева архітектура (three-tier architecture)

Цей підхід передбачає поділ системи на три логічні рівні, кожен з яких виконує власну функцію у процесі обробки даних. Першим є презентаційний рівень, який відповідає за взаємодію з користувачем і реалізується у вигляді веб- або мобільного клієнта. Другим виступає рівень бізнес-логіки, що виконує основну обробку запитів, реалізує алгоритми адаптивного планування та координує роботу між користувацьким інтерфейсом і сховищем даних. Третім є рівень даних, який забезпечує збереження та структуровану організацію інформації про користувачів, робочий час, маршрути, зміни та інші параметри, необхідні для функціонування системи. Така структура дозволяє забезпечити чітке розмежування обов’язків між компонентами, підвищити стабільність системи, спростити її підтримку та масштабування.

Така структура забезпечує незалежність між компонентами, що дозволяє оновлювати або масштабувати кожен рівень без впливу на інші. Наприклад, зміна інтерфейсу користувача не вимагає втручання у бізнес-логіку чи структуру бази даних. Це особливо важливо для вебзастосунків із великою кількістю користувачів та мобільними клієнтами, які працюють у режимі реального часу.

Трирівнева архітектура також спрощує інтеграцію із зовнішніми системами, такими як ERP, CRM або HRM. Наприклад, дані про відпрацьований час співробітників можуть автоматично передаватися до бухгалтерських модулів або систем планування маршрутів. Це дозволяє створити єдиний інформаційний простір логістичної компанії.

Багаторівнева або мікросервісна архітектура (multi-tier / microservices)

Такий підхід передбачає розподіл системи на низку незалежних модулів (сервісів), кожен з яких виконує окрему функцію — наприклад, модуль обліку часу, модуль аналітики, модуль планування чи авторизації. Кожен мікросервіс має власну базу даних або сховище, а взаємодія між ними здійснюється через API або черги повідомлень.

Основна перевага цього підходу — висока гнучкість і масштабованість, адже можна оновлювати окремі частини системи без простоїв, а також розподіляти навантаження між кількома серверами [20]. Однак розробка і підтримка мікросервісної архітектури вимагає значних ресурсів і складнішої інфраструктури (наприклад, Docker, Kubernetes), що не завжди доцільно для середніх проєктів.

Для системи автоматизованого обліку робочого часу у логістичних компаніях найбільш доцільною є саме трирівнева архітектура, оскільки вона поєднує простоту реалізації, масштабованість та гнучкість. На першому рівні працює вебінтерфейс на React і мобільний застосунок на React Native, які забезпечують користувацький доступ і взаємодію в реальному часі. На другому рівні — серверна частина на Laravel, що реалізує бізнес-логіку адаптивного планування, включно з алгоритмами оптимізації завантаження персоналу. На третьому рівні — база даних MySQL, де зберігається інформація про співробітників, графіки, зміни, маршрути та аналітичні показники.

Таке розділення дозволяє: забезпечити високу продуктивність при великій кількості користувачів;

спростити обслуговування та оновлення окремих компонентів системи, реалізувати адаптивне масштабування, коли обробка запитів користувачів та робота алгоритмів планування розподіляються між кількома серверами, підвищити рівень безпеки, адже база даних фізично ізольована від інтерфейсної частини, а доступ до неї здійснюється лише через серверний API.

Крім того, трирівнева модель дає змогу легко впровадити адаптивні алгоритми планування, оскільки серверний рівень може динамічно змінювати логіку розподілу завдань без потреби змінювати клієнтську частину [21].

Наприклад, якщо система виявляє перевантаження окремого співробітника, вона автоматично коригує графік, а клієнтські застосунки просто отримують оновлені дані через REST API.

Отже, обраний архітектурний підхід є оптимальним компромісом між гнучкістю, стабільністю та масштабованістю, що забезпечує ефективну роботу системи адаптивного планування у сфері логістики та дозволяє надалі розширювати її функціональність без суттєвих витрат на рефакторинг чи інтеграцію.

1.4.2 Порівняння технологій серверної частини

Серверна частина клієнт-серверної системи є ядром усього застосунку, оскільки саме тут реалізуються бізнес-правила, алгоритми планування, обробка запитів від клієнтів і взаємодія з базою даних. Вибір технології для серверної логіки безпосередньо впливає на швидкодію, масштабованість, стабільність і безпеку всієї системи.

Таблиця

Сучасні серверні технології розвиваються у напрямку підвищення гнучкості, модульності та продуктивності, що дозволяє ефективно реалізовувати адаптивні алгоритми в інформаційних системах [22]. Для систем, які займаються плануванням, аналітикою й управлінням логістичними процесами, серверна частина повинна забезпечувати комплекс вимог, які визначають її стабільність та ефективність.

По-перше, вона має бути здатною обробляти велику кількість одночасних запитів від користувачів та клієнтів, включно з вебінтерфейсами й мобільними застосунками. Це особливо важливо в умовах інтенсивного використання системи кількома категоріями працівників — логістами, диспетчерами, водіями чи адміністраторами, які взаємодіють із нею одночасно.

По-друге, важливою є здатність серверної частини гарантувати збереження цілісності даних під час постійного оновлення інформації, що

стосується робочих графіків, маршрутів, відпрацьованого часу чи зміни статусу завдань. У логістичних системах дані змінюються динамічно, тому навіть незначні затримки або колізії при записі можуть призвести до порушення узгодженості інформації між модулями системи.

По-третє, серверна архітектура має бути достатньо гнучкою для подальшого розширення функціональності без необхідності кардинальної переробки всього програмного коду. Це означає, що нові модулі — наприклад, блок аналітики, прогнозування навантаження чи автоматичної оптимізації маршрутів — повинні легко інтегруватися у вже створену структуру.

По-четверте, важливим аспектом є підтримка інтеграції з іншими корпоративними сервісами через програмні інтерфейси (API) або системи обміну повідомленнями. Такий підхід дозволяє налагодити взаємодію із зовнішніми ERP-, CRM- чи бухгалтерськими системами, забезпечуючи обмін інформацією про персонал, замовлення, транспортні ресурси або фінансові показники.

Таким чином, ефективна серверна частина сучасної інформаційної системи повинна не лише виконувати бізнес-логіку, а й виступати зв’язковою ланкою між усіма компонентами корпоративного середовища, підтримуючи безперервність процесів, високу продуктивність та надійність функціонування.

Вибір технології для реалізації серверної частини

У межах даної магістерської роботи для реалізації серверної частини обрано Laravel (PHP), що є одним із найпопулярніших і найзручніших фреймворків для створення вебсистем середнього рівня складності. Його використання зумовлене низкою технічних і практичних переваг:

Висока швидкість розробки. Laravel має зрозумілу архітектуру MVC (Model–View–Controller), завдяки якій програмна логіка, представлення й дані чітко розділені. Це дає змогу легко підтримувати й масштабувати систему.

Зручна ORM Eloquent. Механізм Object-Relational Mapping спрощує взаємодію з базою даних, дозволяючи працювати з таблицями як з об’єктами. Це особливо корисно при розробці модулів адаптивного планування, де потрібно швидко змінювати дані про співробітників, зміни, маршрути тощо.

Підтримка REST API. Laravel має вбудовані інструменти для створення RESTful сервісів, що дозволяє організувати ефективну взаємодію між сервером і клієнтом (React, React Native). Завдяки цьому система може передавати оновлення даних у реальному часі, що важливо для динамічних логістичних процесів.

Високий рівень безпеки. Фреймворк забезпечує захист від основних вебзагроз (SQL Injection, CSRF, XSS), підтримує шифрування паролів і токенів, а також має модульну систему контролю доступу, що дозволяє розмежовувати права користувачів (адміністратор, логіст, водій тощо).

Модульність і розширюваність. Laravel дозволяє легко додавати нові функціональні модулі (наприклад, аналітику продуктивності, автоматичні сповіщення або інтеграцію з GPS-трекерами) без суттєвого впливу на основну логіку системи.

Активна спільнота та надійна документація. Завдяки великій кількості бібліотек, плагінів та готових рішень Laravel знижує витрати часу на розробку й тестування.

Для реалізації адаптивного планування у логістичних процесах Laravel також надає можливість використовувати черги (queues) та завдання у фоновому режимі (jobs), що дозволяє обробляти великі обсяги даних без затримки основного інтерфейсу. Наприклад, при оновленні змін або розподілі маршрутів система може виконувати обчислення у фоновому режимі, а користувач миттєво бачить лише оновлені результати [23].

Ще однією перевагою є підтримка API-автентифікації через Laravel Sanctum або Passport, що забезпечує безпечну взаємодію з мобільним застосунком (React Native). Таким чином, користувачі можуть авторизуватись у системі з різних пристроїв, зберігаючи високий рівень безпеки даних.

У поєднанні з MySQL як основною системою керування базами даних Laravel демонструє стабільну роботу навіть при великій кількості одночасних підключень, що підтверджує його придатність для систем класу HRM/ERP, де необхідно керувати персоналом, змінами та аналітикою у режимі реального часу.

Отже, вибір Laravel для реалізації серверної частини системи є обґрунтованим з точки зору технічної ефективності, безпеки та гнучкості. Фреймворк забезпечує швидку розробку, можливість інтеграції з сучасними фронтенд-технологіями, такими як React, а також має усі необхідні інструменти для побудови адаптивної архітектури, здатної динамічно реагувати на зміни у логістичних процесах.

1.5 Висновки до розділу

У першому розділі було здійснено комплексний теоретичний аналіз проблематики розробки клієнт-серверних систем, що реалізують алгоритми адаптивного планування в логістичних процесах, а також визначено основні вимоги до архітектури, функціональної структури та вибору технологій для створення вебзастосунку автоматизованого обліку робочого часу.

Проведене дослідження дало змогу встановити, що сучасні інформаційні системи у сфері логістики потребують високого рівня гнучкості, інтегрованості та продуктивності. В умовах динамічного середовища, коли логістичні процеси постійно змінюються, а кількість об’єктів управління (персонал, транспорт, склади, маршрути) зростає, класичні централізовані рішення без можливості масштабування вже не задовольняють вимоги до швидкості реакції, стабільності роботи та точності планування. Саме тому клієнт-серверна архітектура стає найбільш доцільною моделлю побудови таких систем, адже вона дозволяє розподілити функціональні обов’язки між рівнями, забезпечити централізований контроль за даними та створити передумови для адаптивного управління ресурсами.

У ході аналізу було розглянуто основні архітектурні підходи до побудови клієнт-серверних систем — двохрівневий, трирівневий і багаторівневий (мікросервісний). Встановлено, що для реалізації системи автоматизованого обліку робочого часу в логістичних компаніях найоптимальнішим є трирівневий підхід, який забезпечує чіткий розподіл функцій між рівнями представлення,

бізнес-логіки та даних. Така архітектура дозволяє досягти високої стабільності, спростити супровід, забезпечити можливість паралельної розробки та оновлення окремих модулів без зупинки всієї системи. Крім того, трирівнева модель відкриває широкі можливості для інтеграції із зовнішніми корпоративними сервісами через REST API, що є важливим для логістичних організацій із розгалуженою структурою.

Окрему увагу приділено порівнянню сучасних технологій реалізації серверної частини. Було досліджено переваги та недоліки найбільш поширених платформ: Node.js, Django, Spring Boot, ASP.NET Core та Laravel. Аналіз показав, що, незважаючи на певні відмінності у продуктивності та вимогах до ресурсів, кожна з технологій має свою нішу застосування. Для розробки систем середнього рівня складності з орієнтацією на гнучкість, безпеку й швидку інтеграцію найбільш доцільним виявився вибір Laravel (PHP).

Вибір Laravel як серверного фреймворку зумовлений його високим рівнем абстракції, зручною архітектурою MVC, вбудованими засобами для створення REST API, потужною ORM (Eloquent) та розвиненими механізмами безпеки. Завдяки цьому фреймворк дозволяє ефективно реалізовувати алгоритми адаптивного планування, виконувати складні транзакції, забезпечувати асинхронну обробку завдань та взаємодію з клієнтськими додатками. Також важливою перевагою є наявність розвиненої екосистеми, що включає інструменти для роботи з чергами, подіями, системами кешування й контролем доступу.

Було також відзначено, що ефективна серверна частина логістичної системи повинна відповідати низці ключових вимог: підтримувати обробку великої кількості одночасних запитів від користувачів, забезпечувати збереження цілісності даних у процесі постійних оновлень, бути гнучкою для розширення функціональності без необхідності повного рефакторингу, а також мати можливість інтеграції з іншими корпоративними рішеннями через API. Ці характеристики визначають основу для побудови системи, здатної до масштабування та адаптації під реальні потреби підприємства.

У результаті проведеного аналізу встановлено, що поєднання технологій Laravel (для серверної частини), MySQL (для управління даними) та React/React Native (для клієнтської частини) забезпечує оптимальний баланс між швидкодією, гнучкістю та стабільністю. Таке рішення дозволяє створити єдину інтегровану платформу, що поєднує веб- та мобільний інтерфейси, підтримує роботу в реальному часі та забезпечує безпечний обмін даними через REST API.

Загалом, результати теоретичного аналізу підтверджують доцільність використання клієнт-серверної архітектури для побудови вебзастосунку автоматизованого обліку робочого часу у сфері логістики. Такий підхід не лише забезпечує технічну надійність і гнучкість системи, а й створює основу для впровадження адаптивних алгоритмів планування, які здатні підвищити ефективність використання людських і транспортних ресурсів, зменшити кількість помилок при формуванні розкладів і підвищити загальний рівень продуктивності підприємства.

Таким чином, у першому розділі сформовано теоретичне підґрунтя для подальшого етапу роботи — проєктування клієнт-серверного вебзастосунку, що буде реалізовано в другому розділі. На основі проведеного аналізу визначено архітектурну модель системи, обґрунтовано вибір технологій, сформульовано ключові вимоги до функціональності, інформаційної структури та взаємодії компонентів. Отримані результати є базою для розробки ефективного, масштабованого та безпечного рішення, здатного забезпечити автоматизацію процесів планування й обліку робочого часу у логістичних компаніях.

2 ПРОЄКТУВАННЯ КЛІЄНТ-СЕРВЕРНОГО ЗАСТОСУНКУ

2.1 Обґрунтування вибору архітектури та технологій реалізації

При проєктуванні інформаційної системи автоматизованого обліку робочого часу з модулем адаптивного планування для логістичних процесів основною вимогою є поєднання продуктивності, надійності, безпеки та можливості подальшого розвитку без кардинальної переробки фундаментальних компонентів. Аналіз предметної області показує, що система повинна опрацьовувати динамічні дані в реальному часі (зміни графіків, статуси рейсів, події від мобільних клієнтів), забезпечувати узгодженість транзакцій при одночасних оновленнях та інтегруватися з внутрішніми корпоративними сервісами. У цьому контексті обґрунтування вибору архітектури базується на чотирьох ключових аспектах: модульність і розмежування відповідальностей, продуктивність при одночасних навантаженнях, гарантування цілісності даних і безпека, а також операційна керованість і можливість поступової еволюції системи.

Першим аргументом на користь трирівневої клієнт-серверної архітектури є чітке розділення функцій між презентаційною частиною, серверною логікою та рівнем збереження даних. Це розмежування полегшує паралельну розробку, спрощує тестування і дозволяє впроваджувати оновлення локально на одному рівні без порушення роботи інших [25]. У практичних умовах логістичного підприємства це означає, що інтерфейс водія або диспетчера може оновлюватись без необхідності змінювати механізми адаптивного планування або структуру бази даних, що знижує ризики простоїв при введенні нових features.

Другим обґрунтуванням є потреба в ефективній обробці одночасних запитів та фонового виконання обчислюваних задач адаптації планів. Тому серверний шар має бути спроєктований так, щоб підтримувати асинхронні черги завдань і механізми фонового виконання (jobs/queues), кешування гарячих даних

та горизонтальне масштабування за потреби. Практична реалізація цього підходу в межах обраного стека (Laravel + Redis/черги) [26] дозволяє виокремити «важкі» обчислення (оптимізація розкладів, навчання/виконання моделей підкріплювального навчання, генерація звітів) у фон і повертати результати користувачам асинхронно, не блокуючи інтерфейси. Використання кешу (Redis) для часто запитуваних агрегатів (поточний стан змін, список доступних співробітників) зменшує затримки при відображенні інформації у веб- і мобільному клієнтах.

Третім критичним аспектом є забезпечення цілісності даних і коректної обробки транзакцій у багатокористувацькому середовищі. Для цього архітектура має опиратися на реляційне сховище з підтримкою ACID-транзакцій (MySQL або PostgreSQL), чітко продуманою схемою даних, індексами для продуктивних запитів та механізмами контролю конкурентного доступу (оптимістична/песимістична блокування там, де це необхідно). У межах адаптивного планування важливо проектувати структуру так, щоб операції з масової генерації розкладів виконувалися в ізольованих транзакціях або у фоні з відкладеним застосуванням змін, при цьому зберігаючи можливість відкату в разі конфліктів або невдалих оптимізацій. Архітектурне рішення повинно також передбачати механізми версіонування розкладів та збереження історії для аудиту та аналізу [27-30].

Четвертим аспектом є безпека та відповідність регуляторним вимогам. Оскільки система оперує персональними даними та кадровою інформацією, архітектура повинна забезпечувати централізовану автентифікацію та авторизацію, шифрування даних при передачі і збереженні, журналювання доступу й аудиторні логи. У локальному середовищі це означає розгортання TLS-з’єднань на граничних вузлах (Nginx), зберігання ключів і сертифікатів у захищеному сховищі, внутрішню мережеву сегментацію (відокремлення БД-сегменту від зовнішніх інтерфейсів) та регулярне резервне копіювання на виділені носії з позамежним копіюванням (off-site) для відновлення після аварій. Архітектура має також підтримувати багаторівневу модель доступів (RBAC) з

розширюваним набором ролей (адміністратор, логіст, диспетчер, водій, аналітик) та політиками доступу до чутливих операцій.

Ще одне практичне обґрунтування — вимога до інтеграції з існуючими корпоративними системами (ERP, 1C, BAS тощо). Архітектура повинна забезпечувати стандартизований інтерфейс обміну (REST API з чіткою документацією, можливість Webhook-подій, черги повідомлень для асинхронної інтеграції). У локальній інфраструктурі це дає змогу налаштувати внутрішні шлюзи та ETL-процеси без виходу даних у публічні хмари, що важливо з погляду політики безпеки організації.

З огляду на вищевикладене, обґрунтованим вибором є трирівнева архітектура з монолітною (або модульною) серверною частиною на базі Laravel, реляційною базою даних (MySQL) і клієнтськими застосунками на React (веб) і React Native (мобільні). Така конфігурація поєднує переваги швидкої розробки, зрозумілої структури коду, вбудованих механізмів безпеки і широкої екосистеми для інтеграції та операційної підтримки. Монолітний підхід на старті дозволяє швидко вивести систему в експлуатацію і за потреби, коли зросте навантаження або з’являться вимоги до окремого масштабування компонентів, поступово виділяти ресурсоємні підсистеми (модуль адаптивного планування, аналітичний рушій) у вигляді окремих сервісів або контейнеризованих мікросервісів, що комунікують через API або брокер повідомлень (AMQP).

Нарешті, архітектурне обґрунтування включає операційні вимоги: використання контейнеризації (Docker Compose для локального розгортання та тестування), централізованих механізмів логування й моніторингу (локальні стек-рішення для збирання метрик і алертів), політик оновлення (CI/CD-потоки у внутрішньому GitLab/Gitea) та планів резервного копіювання і відновлення. У локальному середовищі ці підходи дозволяють зберегти контроль над інфраструктурою й водночас підвищити надійність і відтворюваність розгортання.

Отже, вибір трирівневої клієнт-серверної архітектури з Laravel у ролі серверного шару та реляційною БД у локальному середовищі є оптимальним

компромісом між швидкістю реалізації, технічною надійністю, безпекою та можливістю еволюційного масштабування системи для забезпечення адаптивного планування в логістиці [31]. Це рішення відповідає як функціональним вимогам (реальне часове оновлення, обробка транзакцій, інтеграція), так і нефункціональним (продуктивність, безпека, керованість, підтримка резервного копіювання і аудиту), і створює надійну основу для подальшої реалізації алгоритмів оптимізації та аналітики.

2.2 Аналіз вимог користувачів і визначення функціональних можливостей

Метою цього підрозділу є формалізація вимог кінцевих користувачів і зацікавлених сторін, ідентифікація бізнес-процесів, які має автоматизувати система, та визначення повного переліку функціональних і нефункціональних можливостей, необхідних для реалізації клієнт-серверного вебзастосунку автоматизованого обліку робочого часу з модулем адаптивного планування. Аналіз базується на типових методах збору вимог: опитування представників замовника, опис бізнес-процесів, моделювання сценаріїв використання (use-cases) та аналіз юридичних і технічних обмежень, що діють у предметній галузі.

2.2.1 Ідентифікація зацікавлених сторін і ролей користувачів

До основних зацікавлених сторін належать: адміністратор системи, HR/кадровик, логіст/планувальник, диспетчер, водій/польовий працівник, бухгалтер, аналітик і служба інформаційної безпеки організації. Адміністратор відповідає за налаштування системи, розподіл ролей і політик доступу; HR працює з персональними картками співробітників, графіками і відпустками; логіст формує зміни і завдання, контролює їх виконання; диспетчер моніторить поточні рейси і коригує маршрути в реальному часі; водій використовує мобільний додаток для фіксації часу, отримання завдань і повідомлень; бухгалтер отримує дані для нарахування заробітної плати та звітності; аналітик

формує звіти і KPI; служба безпеки — забезпечує моніторинг інцидентів, аудит доступів і дотримання політик. Для кожної ролі визначаються права доступу, робочі сценарії і вимоги до інтерфейсу (наприклад, мобільний UI для водіїв має бути мінімалістичним і працездатним в умовах обмеженого з'єднання).

2.2.2 Функціональні вимоги (детальний перелік)

Система повинна забезпечувати повний набір функцій, що відповідає ролям і бізнес-процесам. Нижче подано детальний опис основних функціональних областей з мінімальними (must), бажаними (should) та додатковими (could) вимогами.

Аутентифікація та авторизація. Система повинна підтримувати реєстрацію та автентифікацію користувачів, багатофакторну автентифікацію для облікових записів з підвищеними правами, механізм відновлення пароля. Необхідна реалізація ролей і політик доступу (RBAC) з можливістю тонкого налаштування прав на операції читання/запису для кожного модуля. (Має: токен-базована авторизація для мобільних клієнтів; Бажано: MFA; Додатково: SSO/LDAP інтеграція.)

Реєстрація робочого часу. Модуль збору даних про фактичний початок і закінчення робочого часу з можливістю фіксації через вебінтерфейс, мобільний додаток, інтеграцію з біометричними терміналами або картами доступу. Система має підтримувати маркування подій (початок зміни, кінець зміни, перерви, відрядження, хвороба), можливість вручну коригувати записи з обов'язковим журналом змін (audit trail) та механізм погодження корекцій керівництвом.

Планування змін і адаптивне планування. Інструмент для створення шаблонів графіків, формування розкладів на підставі навантаження і наявності співробітників; модуль адаптивного планування має вміти коригувати розклади в реальному часі з урахуванням обмежень (трудове законодавство, кваліфікація персоналу, часові вікна доставки), пріоритетів замовлень та непередбачуваних подій. Система повинна дозволяти налаштування правил оптимізації (мінімізація

перевитрат, вирівнювання навантаження, мінімізація простоїв) та зберігати історію версій розкладів.

Сповіщення та повідомлення. Реалізація push-сповіщень у мобільному додатку, SMS або email-сповіщень про зміни у графіку, екстрені повідомлення/оновлення маршрутів, нагадування про початок зміни. Повинна бути можливість налаштовувати канали і пріоритети повідомлень за ролями.

Моніторинг у реальному часі. Дашборди для логістів і диспетчерів із відображенням поточного статусу змін, завантаженості персоналу, геолокації транспортних засобів (за умови інтеграції з GPS), та індикаторами відхилень (пропущені відмітки, затримки). Візуалізація аналітики в режимі near-real-time.

Аналітика та звітність. Набір стандартних звітів: відпрацьований час за період, понаднормові години, запізнення, KPI по співробітниках/підрозділах, витрати часу на рейс, ефективність планування. Має бути можливість формувати кастомні звіти, експортувати їх у CSV/PDF і автоматично планувати періодичні звітні завдання.

Інтеграція з корпоративними системами. REST API для двосторонньої інтеграції з ERP/1C/BAS/бухгалтерією, можливість налаштування webhook-ів для подій, підтримка черг повідомлень для асинхронного обміну (наприклад, через RabbitMQ). Механізми трансформації даних і ETL-процеси для періодичної синхронізації.

Адміністрування і налаштування. Консоль адміністрування для налаштування ролей, політик, шаблонів розкладів, робочих правил і меж доступу; логування і перегляд аудиторних записів; механізми резервного копіювання і відновлення; конфігурація черг і планувальників фонового виконання.

Мобільний клієнт. Підтримка офлайн-режиму з синхронізацією при відновленні зв’язку; зручний інтерфейс для фіксації часу, отримання завдань і повідомлень; мінімальні витрати батареї й трафіку; безпечне зберігання токенів. Керування помилками та відкатом змін. Механізми попереднього перегляду запропонованих змін, тестового застосування розкладів у песимістичному

середовищі, журналювання результатів оптимізації і можливість відкотити застосовані корекції до попередньої версії.

Документування та допомога. Вбудована система підказок, документація API, ручні інструкції для HR/адміністраторів і модуль навчання для користувачів.

2.2.3 Нефункціональні вимоги

Система повинна відповідати прийнятним рівням якості обслуговування: час відгуку для операцій читання не більше 200–500 мс при типовому навантаженні; операції запису (створення/корекція відмітки) — відповіді в межах 500–1000 мс при нормальному навантаженні; масштабованість — можливість горизонтального масштабування серверного шару при зростанні кількості користувачів; відмовостійкість — забезпечення RTO і RPO, погоджених з політикою організації; доступність сервісу — цільовий SLA для критичних модулів (наприклад, 99,5% або інший показник, погоджений із замовником). Важлива вимога — локальне розгортання й повний контроль над інфраструктурою (відсутність хмарних постачальників), що накладає обмеження на архітектуру (напр., необхідність резервних носіїв у внутрішній мережі, організація off-site копій).

Крім того, система повинна бути зручною у використанні (UX), локалізованою українською мовою з можливістю додавання інших мов, доступною на типовому обладнанні співробітника (веб-браузер, смартфон андроїд/iOS), тестованою на безпеку (SAST/DAST) і відповідати вимогам продуктивності при пікових навантаженнях (план на тести навантаження).

Додатково система має підтримувати централізоване журналювання та моніторинг ключових показників продуктивності, що забезпечить оперативне виявлення відхилень у роботі. Передбачене також використання модульної архітектури, яка спростить інтеграцію з іншими.

2.2.4 Модель даних (схематично)

Основні сутності системи: «Користувач», «Роль», «Посада/Кваліфікація», «Зміна/Шаблон зміни», «Призначення (assignment)», «Запис часу (time_entry)», «Подія/Ісключення», «Маршрут/Рейс», «Транспортний засіб», «Пристрій фіксації (термінал/GPS)», «Журнал аудиту», «Звіт». Для кожної сутності необхідно визначити обов’язкові атрибути, ключові індекси для продуктивних запитів і обмеження цілісності (FK, унікальні індекси), а також політики збереження (retention) і архівації даних (рисунок 2.1).

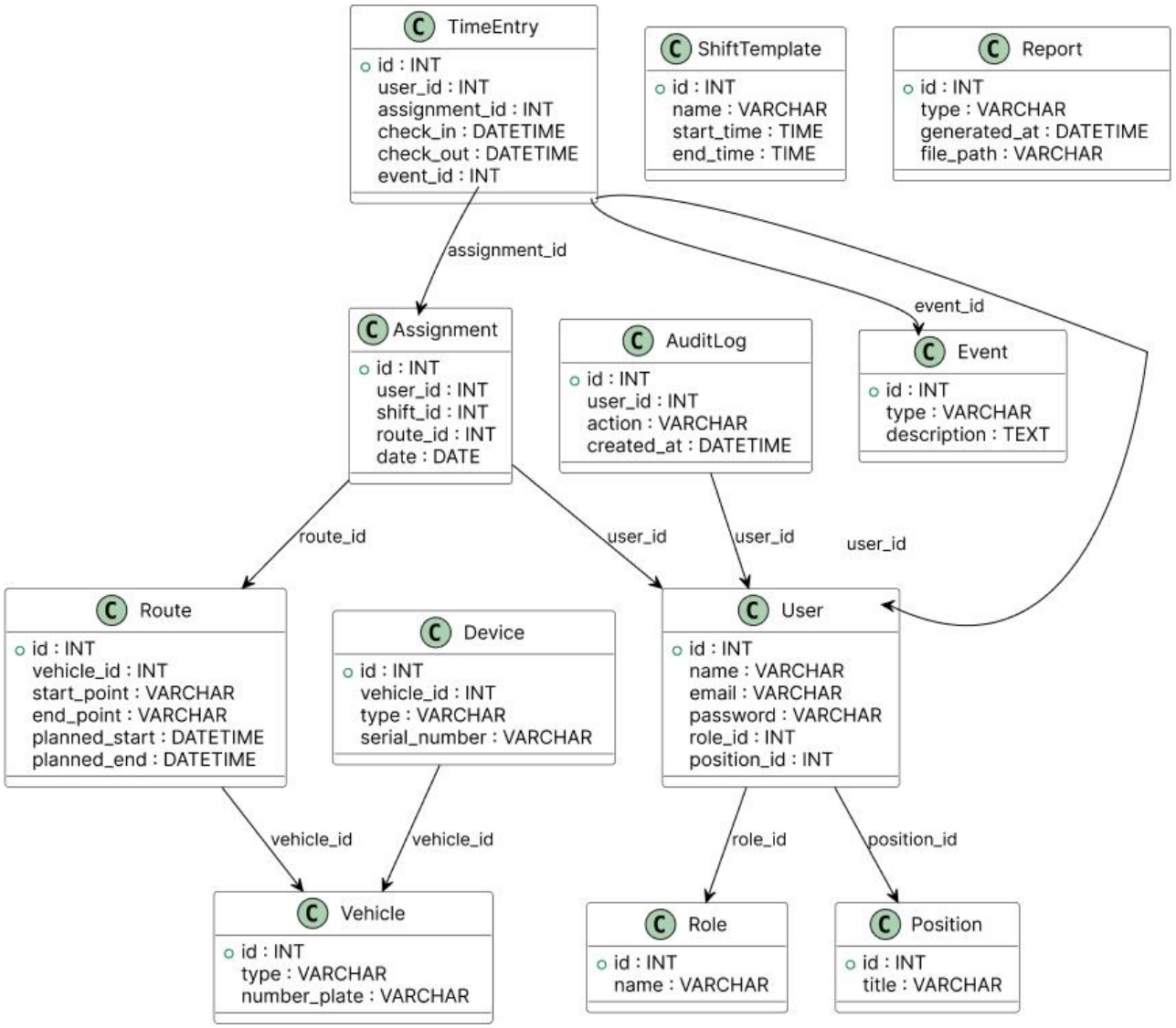


Рисунок 2.1 – Модель даних системи LogiTime з основними сутностями та їх зв’язками.

2.2.5 API та інтеграційні вимоги

Комунікація між клієнтом і сервером має здійснюватися через RESTful API з використанням JSON як формату обміну. Для подій реального часу рекомендується реалізувати WebSocket-канали або використовувати push-повідомлення для мобільних клієнтів. API має підтримувати версіонування, обмеження частоти запитів (rate limiting), механізми пагінації й фільтрації, детальну специфікацію (OpenAPI/Swagger) та тестові стенди (sandbox). Для інтеграцій з ERP/1C потрібно передбачити ETL-шари із можливістю мапінгу полів та періодичного імпорту/експорту.

2.2.6 Безпека та захист інформації — вимоги відповідно до чинного законодавства та стандартів ISO

Система оброблятиме персональні дані співробітників, зокрема прізвище, ім’я, по батькові, унікальні ідентифікатори, робочі графіки, інформацію про відрядження та інші службові відомості. Тому її архітектура, логіка обробки даних і експлуатаційне середовище повинні відповідати вимогам чинного законодавства України [32] у сфері захисту персональних даних, а також міжнародним стандартам інформаційної безпеки. Згідно із Законом України «Про захист персональних даних», обробка таких відомостей є регульованою діяльністю, що потребує визначення правових підстав, дотримання принципів мінімізації даних, прозорості, точності та обмеження термінів зберігання.

Відповідно до цього, система повинна гарантувати цілісність, конфіденційність і доступність персональних даних у процесі збирання, зберігання, обробки, передачі та видалення інформації.

Додатково система автоматизованого обліку робочого часу має враховувати вимоги трудового законодавства, зокрема положення Кодексу законів про працю України, що регламентують порядок ведення табелів обліку робочого часу, зберігання відомостей про відпрацьовані години та формування звітності. Це

означає, що система повинна забезпечувати не лише технічний, але й правовий рівень достовірності даних, можливість їх відтворення, перевірки та підтвердження у разі внутрішнього чи зовнішнього аудиту [33].

З точки зору міжнародних норм, доцільно будувати систему інформаційної безпеки відповідно до принципів і вимог стандартів сімейства ISO/IEC 27000. Найбільш значущим є стандарт ISO/IEC 27001, який визначає вимоги до створення, впровадження, підтримки та вдосконалення системи менеджменту інформаційної безпеки (ISMS). Реалізація принципів цього стандарту дозволяє системно підходити до управління ризиками, формувати політики безпеки, визначати відповідальних осіб, а також впроваджувати процедури моніторингу та реагування на інциденти. Для систем, що обробляють персональні дані співробітників, це особливо важливо, оскільки дає змогу гарантувати відповідність як внутрішнім вимогам безпеки організації, так і зовнішнім нормативним актам [34].

Побудова безпечного середовища передбачає реалізацію низки технічних і організаційних заходів. Передусім необхідно визначити правову основу обробки персональних даних — виконання трудового договору, законний інтерес роботодавця або письмову згоду працівника в окремих випадках. Політики обробки мають бути належним чином задокументовані, а операції з персональними даними — відображені у внутрішньому реєстрі. Важливим аспектом є забезпечення інформування суб’єктів даних про мету обробки, строк зберігання та їхні права на доступ, виправлення чи видалення інформації.

З технічної точки зору система повинна реалізовувати чіткий контроль доступу до даних на основі ролей (RBAC) з принципом найменших привілеїв.

Адміністративні облікові записи мають бути захищені багатофакторною автентифікацією, а паролі користувачів — відповідати політикам складності та періодичної зміни. Для конфіденційних полів бази даних, таких як ідентифікаційні номери чи фінансові реквізити, рекомендується застосовувати логічне шифрування на рівні застосунку.

Передавання інформації між клієнтською та серверною частиною має здійснюватися виключно через зашифровані канали із застосуванням протоколу TLS версії не нижче 1.2, а збережені дані — шифруватися на дисковому рівні або у прикладному шарі із використанням стійких алгоритмів, наприклад AES-256.

Ключі шифрування повинні зберігатися у захищених сховищах із регламентованою політикою ротації.

Особливу роль у системі відіграє журналювання подій. Усі операції, пов’язані зі створенням, зміною чи видаленням записів, мають фіксуватися у незмінному журналі аудиту із зазначенням користувача, часу, IP-адреси та дії. Це забезпечує прозорість процесів, контроль доступу і можливість подальшого розслідування інцидентів безпеки.

Інфраструктура системи повинна бути ізольованою у межах локального середовища підприємства. Сервери бази даних, додатків і адміністративні консолі мають розміщуватись у різних сегментах мережі (VLAN), захищених міжмережевими екранами. Доступ до адміністративних ресурсів дозволяється виключно через захищені канали VPN із обов’язковим журналюванням усіх сесій. Для додаткового рівня безпеки доцільно використовувати системи виявлення та запобігання вторгненням (IDS/IPS), а також реверс-проксі для захисту зовнішніх API.

Забезпечення безперервності роботи передбачає реалізацію політики резервного копіювання — регулярне створення інкрементних та повних копій даних, зберігання їх у захищених місцях поза межами основного сайту, тестування відновлення та розробку плану аварійного відновлення (DRP). Важливо визначити цільові показники часу відновлення (RTO) і допустимої втрати даних (RPO), які відповідають бізнес-вимогам організації.

З точки зору продуктивності, система має бути побудована з урахуванням масштабованості — застосування кешування для швидкодії, черг повідомлень для фонової обробки даних, а також можливості горизонтального розширення серверного шару.

Розробка програмного забезпечення повинна здійснюватися за принципами безпечного життєвого циклу (Secure SDLC), що включає статичне та динамічне тестування безпеки (SAST і DAST), регулярне оновлення залежностей, усунення відомих вразливостей та використання контейнеризації для ізоляції середовищ.

У межах управління інцидентами інформаційної безпеки мають бути розроблені процедури виявлення, реєстрації та реагування на інциденти, включаючи повідомлення відповідальних органів і суб'єктів даних у випадку порушення конфіденційності. Для забезпечення приватності при аналітичних обчисленнях необхідно застосовувати анонімізацію або псевдонімізацію даних, що унеможливлює ідентифікацію конкретних осіб.

При створенні модулів, які здійснюють автоматизовану обробку великих обсягів персональних даних, рекомендовано проводити оцінку впливу на захист даних (DPIA) для виявлення ризиків і визначення додаткових заходів контролю. Постачальники зовнішніх сервісів, які мають доступ до даних системи (наприклад, SMS-шлюзи або GPS-трекінг), повинні укладати договори з чітко визначеними умовами обробки та відповідальності за дотримання безпеки.

В Україні триває процес гармонізації законодавства у сфері захисту персональних даних із загальноєвропейським регламентом GDPR. Тому при проектуванні системи доцільно орієнтуватися на принципи законності, справедливості, прозорості, обмеження мети, мінімізації обсягу обробки, точності, обмеження зберігання, цілісності та конфіденційності. Дотримання цих принципів забезпечить як відповідність чинним нормам, так і готовність до майбутніх змін законодавства.

Інтегрування положень стандарту ISO/IEC 27001 у процес розробки та експлуатації системи дозволяє створити формалізовану модель управління ризиками інформаційної безпеки. Це включає оцінку ризиків, визначення політик і процедур безпеки, контроль доступу, моніторинг, реагування на інциденти та постійне вдосконалення заходів безпеки. Такий підхід демонструє відповідність міжнародним практикам управління інформаційною безпекою, що

є особливо важливим при взаємодії з державними або корпоративними партнерами. Впровадження системи менеджменту інформаційної безпеки (ISMS) також спрощує проходження аудиту та сертифікації, підвищуючи довіру до інформаційної системи в цілому.

2.3 Проєктування інформаційної структури системи

Проєктування інформаційної структури є одним із найважливіших етапів створення клієнт-серверної системи, оскільки саме на цьому етапі формується логіка зберігання, обробки та обміну даними між усіма компонентами програмного комплексу. Для системи адаптивного планування, орієнтованої на оптимізацію логістичних процесів, інформаційна структура повинна забезпечувати цілісність, узгодженість, своєчасність та безпеку даних у реальному часі. З огляду на це, основним завданням даного етапу є створення логічно обґрунтованої моделі даних, яка б відповідала функціональним вимогам користувачів, технологічним можливостям системи та стандартам інформаційної безпеки.

Побудова інформаційної структури базується на принципах системного підходу та моделювання предметної області. Для цього спочатку проводиться аналіз бізнес-процесів, які система має автоматизувати, визначаються основні інформаційні потоки, що циркулюють між користувачами, підрозділами та технічними модулями. На основі отриманих даних формується узагальнена концептуальна модель, яка описує сутності, їх властивості та взаємозв’язки. Далі відбувається логічне проєктування, у межах якого визначаються структури таблиць, ключі, обмеження цілісності, а також принципи нормалізації даних, що дозволяють уникнути надлишковості та дублювання інформації.

У контексті розробки системи адаптивного планування у сфері логістики інформаційна структура ґрунтується на реляційній моделі даних, що забезпечує високу стабільність, формалізованість і зручність у реалізації механізмів контролю цілісності. Основними сутностями системи є користувачі, робочі

зміни, записи робочого часу, маршрути, завдання, транспортні засоби, журнали аудиту та звіти. Усі ці елементи формують єдиний логічний простір даних, який підтримує гнучку взаємодію між модулями, що відповідають за планування, моніторинг і аналітику.

Інформаційна структура системи будується відповідно до трирівневої клієнт-серверної архітектури, де виділяють рівень даних, рівень бізнес-логіки та рівень представлення. На рівні даних функціонує система керування базами даних MySQL, яка забезпечує високий рівень продуктивності, підтримку транзакцій, реплікацію та надійні засоби резервного копіювання. Рівень бізнес-логіки, реалізований у межах фреймворку Laravel, відповідає за обробку запитів, виконання алгоритмів адаптивного планування, перевірку прав доступу та узгодження взаємодії між компонентами [35]. На рівні представлення користувачі отримують доступ до даних через веб- або мобільний інтерфейс, побудований за допомогою сучасних технологій (наприклад, React), що забезпечує зручність використання, динамічне оновлення інформації та адаптацію під різні пристрої.

Логічна модель даних у системі передбачає чітку структуризацію інформаційних об’єктів. Для кожної сутності визначено набір атрибутів, унікальний ідентифікатор та зовнішні зв’язки, що дозволяють підтримувати узгодженість інформації. Наприклад, кожен користувач системи має певний набір характеристик — персональні дані, посаду, роль, відділ і статус активності.

Кожна робоча зміна асоціюється з конкретними співробітниками, має часові межі, тип зміни, прив’язку до маршруту або об’єкта логістики. Записи робочого часу фіксують події початку, завершення зміни, відхилення від графіка чи запізнення, а також зберігають інформацію про пристрій, через який було здійснено реєстрацію. Така деталізація дозволяє формувати достовірну картину робочого процесу та забезпечує аналітичну основу для модулів адаптивного планування.

Між сутностями системи встановлюються логічні зв’язки, які відображають реальні відносини у предметній області. Залежно від характеру

взаємодії, можуть застосовуватися зв’язки типу «один до багатьох», «багато до багатьох» або «один до одного». Наприклад, один користувач може мати декілька записів про робочий час, але кожен запис належить лише одному користувачу [36]. Аналогічно, одна зміна може бути пов’язана з кількома завданнями, що реалізуються одночасно у різних підрозділах. Для підтримки таких взаємозв’язків використовуються зовнішні ключі та проміжні таблиці, що забезпечують референційну цілісність бази даних.

Обмін інформацією між клієнтською та серверною частинами системи реалізується за допомогою RESTful API, який використовує формат обміну даними JSON. Це забезпечує незалежність між компонентами, спрощує інтеграцію з іншими системами та дозволяє реалізувати асинхронну взаємодію. Для оптимізації продуктивності застосовується кешування результатів запитів, а транзакційна модель гарантує узгодженість змін при виконанні операцій запису. У випадках, коли потрібна обробка великої кількості запитів, використовується черга повідомлень, що дає змогу розподіляти навантаження між обчислювальними вузлами.

Проєктування інформаційної структури також включає заходи з забезпечення безпеки даних. Кожен елемент бази даних повинен бути захищений відповідно до принципів конфіденційності, цілісності та доступності. Це передбачає розмежування прав доступу до таблиць і полів залежно від ролі користувача, шифрування критичних атрибутів, регулярне журналювання змін і проведення контролю аудиту [37]. Усі транзакції з персональними даними мають проходити через шифровані канали зв’язку, що відповідають стандартам безпеки TLS 1.3, а резервне копіювання даних виконується автоматично з перевіркою цілісності копій.

Розроблена інформаційна структура забезпечує високу узгодженість між логічною та фізичною моделями даних. Вона є гнучкою до модифікацій, що дозволяє масштабувати систему у разі зростання кількості користувачів, розширення функціональних можливостей або підключення додаткових аналітичних модулів. Водночас дотримання принципів нормалізації,

стандартизації атрибутів і чіткої типізації полів гарантує ефективне використання ресурсів бази даних і зниження ризику втрати або дублювання інформації.

Таким чином, інформаційна структура системи є основою її ефективного функціонування. Вона не лише визначає логіку взаємодії між об’єктами, але й забезпечує надійну підтримку алгоритмів адаптивного планування, що дозволяють автоматизувати управління робочим часом, маршрутизацію та аналіз ефективності логістичних процесів. Реалізація такої структури створює передумови для стабільної роботи системи, її подальшого розвитку та інтеграції в єдину інформаційну екосистему підприємства.

2.4 Модель бази даних і взаємозв’язки між таблицями

Модель бази даних є структурною основою будь-якої інформаційної системи, що визначає логіку зберігання, обробки та взаємодії даних. Для системи адаптивного планування, спрямованої на оптимізацію логістичних процесів і облік робочого часу, модель бази даних повинна забезпечувати ефективність, масштабованість, узгодженість і безпеку обробки великої кількості транзакцій у реальному часі.

Під час проєктування бази даних було застосовано реляційну модель, оскільки вона надає формальний апарат для опису зв’язків між сутностями та гарантує підтримку цілісності даних. Основним інструментом реалізації обрано систему керування базами даних MySQL, яка добре інтегрується з фреймворком Laravel, забезпечує транзакційність, підтримує ACID-властивості, індексування та реплікацію, що є важливими для стабільної роботи корпоративних систем.

2.4.1 Концептуальна модель бази даних

Концептуальна модель відображає основні об’єкти предметної області та їх зв’язки. Центральним елементом системи є сутність «Користувач», яка описує

персонал, що бере участь у виробничих і логістичних процесах. Для кожного користувача фіксуються ідентифікаційні дані, роль у системі, посада, контактна інформація, статус активності та історія авторизацій. З користувачем пов’язані сутності «Робоча зміна», «Запис робочого часу», «Завдання» та «Журнал аудиту».

Сутність «Робоча зміна» містить інформацію про графік роботи, тривалість зміни, її тип (денна, нічна, позмінна), а також прив’язку до відділу чи маршруту. Таблиця «Записи робочого часу» забезпечує фіксацію фактичного початку, закінчення та тривалості роботи, враховуючи відхилення від планового графіка, запізнення, перерви та відрядження. Кожен запис має посилання на конкретного користувача та зміну, що забезпечує можливість повного відтворення історії діяльності працівника.

Важливе місце у структурі займає сутність «Завдання» (Task), яка визначає конкретні дії або маршрути, що повинні бути виконані в межах зміни. Завдання можуть бути пов’язані з транспортними засобами, маршрутами доставки чи іншими елементами логістичної діяльності. Це дозволяє реалізувати адаптивне планування, коли система автоматично коригує завдання залежно від зміни умов — затримок, відсутності персоналу або збільшення навантаження [38].

Таблиця «Транспортний засіб» зберігає технічні параметри транспорту, ідентифікаційні дані, статус обслуговування та поточну геолокацію (якщо використовується GPS-моніторинг). У разі потреби ця інформація використовується модулем адаптивного планування для оптимізації розподілу ресурсів між маршрутами.

Сутність «Журнал аудиту» (AuditLog) виконує функцію забезпечення прозорості та відстеження дій користувачів. У ній фіксуються всі зміни даних — створення, оновлення, видалення записів, вхід у систему, зміна налаштувань безпеки. Це дозволяє не лише гарантувати відповідність вимогам інформаційної безпеки, а й підвищує довіру до системи з боку адміністрації [39].

Додатково реалізована таблиця «Звіти» (Reports), що містить агреговані дані про виконання планів, ефективність працівників, навантаження за змінами,

час простоїв тощо. Дані цієї таблиці формуються автоматично за допомогою вбудованих алгоритмів аналізу.

2.4.2 Логічна модель бази даних

У межах логічної моделі всі сутності зв’язані між собою через первинні та зовнішні ключі. Основний зв’язок між таблицями «Користувач» і «Робоча зміна» має тип «один до багатьох» — один користувач може брати участь у кількох змінах, але кожна зміна належить одному користувачу. Між таблицями «Користувач» і «Роль» реалізується зв’язок «багато до багатьох», що дозволяє гнучко керувати правами доступу через проміжну таблицю «user_roles». Аналогічний тип зв’язку використовується між «Змінами» та «Завданнями», оскільки одна зміна може містити кілька завдань, а одне завдання може бути виконане в межах різних змін.

Зв’язок між таблицями «Зміна» і «Запис робочого часу» також має тип «один до багатьох», що забезпечує збереження всіх фіксацій подій у межах зміни. Сутність «Журнал аудиту» має зв’язки з усіма основними таблицями, оскільки фіксує операції користувачів у межах усієї системи. Таким чином, модель бази даних формує взаємопов’язану структуру, де всі зміни можуть бути простежені від дії конкретного користувача до кінцевого результату у звіті.

Для забезпечення продуктивності та узгодженості даних застосовуються індекси по найчастіше використовуваних полях — ідентифікаторах користувачів, датах змін, статусах завдань. Це дозволяє зменшити час виконання запитів і підвищує ефективність обробки великих обсягів інформації.

Забезпечення цілісності та узгодженості даних

Особливу увагу приділено механізмам підтримки цілісності даних. Використання обмежень зовнішніх ключів (foreign keys) запобігає появі «сирітських» записів у таблицях, що підвищує достовірність інформації.

Транзакційна модель бази даних гарантує атомарність операцій, тобто у

разі помилки вся серія змін скасовується. Це особливо важливо для систем, які працюють у багатокористувацькому режимі.

В рамках моделі реалізовано також механізм логічного видалення записів (soft delete), що дозволяє зберігати історію змін без фактичного видалення даних із таблиць. Завдяки цьому система зберігає можливість відновлення інформації, що є критично важливим для внутрішнього аудиту та юридичного підтвердження дій користувачів.

2.4.3 Відображення інформаційних потоків

У системі забезпечено двосторонній обмін даними між базою даних і серверним додатком через ORM (Object-Relational Mapping), що спрощує роботу розробників і зменшує ймовірність помилок при виконанні запитів. Передбачено регулярне оновлення інформації у реальному часі, використання черг обробки подій для ресурсомістких операцій та механізмів кешування часто використовуваних даних.

2.5 Розробка API та механізмів взаємодії між клієнтом і сервером

Розробка прикладного програмного інтерфейсу (API) є одним із ключових аспектів побудови клієнт-серверних систем, оскільки саме цей рівень забезпечує взаємодію між компонентами системи, узгодженість передачі даних, безпеку комунікацій і ефективну інтеграцію з зовнішніми сервісами. У системі адаптивного планування для оптимізації логістичних процесів API виступає посередником між клієнтським застосунком (веб- або мобільним інтерфейсом) та серверною частиною, яка реалізує основну бізнес-логіку, алгоритми планування та управління даними.

Архітектурні принципи побудови API

Основою взаємодії у системі є RESTful API (Representational State Transfer), який базується на принципах стандартизації запитів і використанні

протоколу HTTP/HTTPS. Такий підхід забезпечує незалежність клієнтської та серверної частин, спрощує масштабування, підтримує кросплатформеність і гарантує ефективний обмін структурованими даними у форматі JSON. REST-архітектура передбачає використання стандартних методів запитів — GET, POST, PUT, PATCH та DELETE, що відповідають операціям читання, створення, оновлення й видалення ресурсів.

Всі ресурси API системи представлені у вигляді URL-ендпойнтів, кожен з яких відповідає певній сутності бази даних — користувачам, змінам, записам робочого часу, завданням або транспортним засобам. Така структура сприяє логічній прозорості системи та полегшує інтеграцію сторонніх сервісів, наприклад бухгалтерських чи аналітичних модулів.

2.5.1 Механізми автентифікації, авторизації та захисту даних

Безпека взаємодії між клієнтом і сервером є критично важливою складовою системи. Для забезпечення контролю доступу реалізовано токен-базовану авторизацію з використанням механізму JWT (JSON Web Token), який дозволяє користувачам проходити автентифікацію один раз і зберігати токен для подальших звернень до сервера. Це підвищує швидкодію системи та мінімізує навантаження на сервер при повторних запитах.

Кожен запит до API супроводжується передачею токена у заголовку авторизації, що дозволяє серверу перевіряти права користувача та виконувати лише ті операції, які дозволені його роллю. Такий підхід реалізує модель RBAC (Role-Based Access Control), яка чітко визначає права доступу до кожного ресурсу системи.

Усі з’єднання між клієнтом і сервером здійснюються виключно через зашифрований протокол HTTPS із підтримкою TLS 1.3, що гарантує конфіденційність і цілісність переданої інформації. Додатково реалізовано механізми rate limiting для запобігання надмірній кількості запитів і CSRF-захист

для відбиття атак, спрямованих на підробку запитів від імені автентифікованого користувача.

2.5.2 Обробка запитів і структура відповідей

Клієнтські застосунки надсилають запити до API у вигляді HTTP-звернень, після чого серверна частина обробляє їх у кілька етапів: валідація вхідних даних, перевірка прав доступу, виконання бізнес-логіки та формування відповіді. Для валідації застосовується система правил, що перевіряє коректність введених параметрів, а у випадку помилки сервер повертає стандартизовану відповідь з кодом помилки та поясненням.

Результати обробки повертаються клієнту у форматі JSON, що спрощує їх подальше відображення у вебінтерфейсі або мобільному додатку. Усі відповіді структуровані за єдиним шаблоном: об’єкт даних, код статусу (наприклад, 200 — успіх, 400 — помилка запиту, 401 — неавторизовано, 500 — внутрішня помилка сервера) та додаткова службова інформація (timestamp, версія API, повідомлення для користувача).

2.5.3 Реалізація асинхронної взаємодії

Для забезпечення актуальності даних у реальному часі в системі передбачено механізм асинхронної взаємодії між клієнтом і сервером. Для цього використовуються WebSocket-з’єднання, що дозволяють підтримувати двонапрямлену комунікацію без необхідності постійного оновлення сторінки. Такий підхід особливо ефективний для модулів моніторингу логістичних процесів, де важливо оперативно відображати зміни статусу маршрутів, завдань або робочих змін.

Для мобільних користувачів реалізовано систему push-сповіщень, яка інформує про зміни у графіках, нові завдання або критичні події. Сервер формує

повідомлення та надсилає їх через API до клієнтських додатків, забезпечуючи миттєве реагування працівників на зміни у виробничих процесах.

Інтеграція з зовнішніми системами

API системи розроблено з урахуванням можливості інтеграції з іншими корпоративними рішеннями, такими як ERP, CRM, бухгалтерські або транспортні системи. Для цього передбачено механізм webhook-повідомлень, який дозволяє автоматично передавати дані про зміни у системі на зовнішні сервіси. Крім того, реалізовано можливість обміну даними через REST API сторонніх систем, що забезпечує двосторонню інтеграцію.

Для великого обсягу даних використовується механізм ETL-процесів (Extract, Transform, Load), який дозволяє періодично синхронізувати інформацію між різними базами даних, перетворювати структуру полів і зберігати узгодженість інформаційних потоків.

2.5.4 Відповідність принципам безпечного програмування

Розробка API у системі здійснюється з дотриманням сучасних принципів безпечного програмування (secure coding) та рекомендацій OWASP API Security Top 10, що забезпечує комплексний захист від найпоширеніших типів вразливостей. Серед основних заходів безпеки реалізовано захист від SQL-ін’єкцій, XSS-атак, CSRF, підміни токенів та несанкціонованого доступу до ресурсів системи, що дозволяє мінімізувати ризики компрометації даних та забезпечити стабільність роботи додатка. Серверна частина регулярно проходить перевірку безпеки із застосуванням сучасних інструментів статичного та динамічного аналізу коду (SAST і DAST), а результати тестування враховуються при плануванні оновлень та впровадженні нових функціональних модулів, що підвищує загальний рівень надійності системи.

Для контролю доступу передбачено використання гнучких моделей RBAC (Role-Based Access Control) та ABAC (Attribute-Based Access Control), що дозволяють детально налаштовувати права користувачів відповідно до їх ролей,

атрибутів та контексту використання. Це забезпечує диференційований доступ до ресурсів та мінімізує ймовірність помилкового або зловмисного використання системи. Крім того, в системі реалізовано аудит всіх критичних операцій, включаючи зміну даних, доступ до конфіденційної інформації та взаємодію з ключовими сервісами. Аудит ведеться у вигляді журналів дій із часовими мітками та ідентифікацією користувачів, що дозволяє забезпечити прозорість і відстежуваність дій, а також полегшує аналіз інцидентів безпеки у разі потреби.

У поєднанні всі ці заходи формують комплексну стратегію кіберзахисту API, яка гарантує високий рівень безпеки обміну даними, збереження конфіденційної інформації та відповідність сучасним стандартам інформаційної безпеки організації. Такий підхід дозволяє системі адаптивного планування надійно працювати у корпоративному середовищі, де критично важливе збереження цілісності даних та контроль за доступом до інформації.

2.5.6 Тестування, версіонування та документація API

Для забезпечення стабільності роботи API реалізовано систему версіонування, що дозволяє підтримувати сумісність старих клієнтів при оновленні серверної частини. Кожна нова версія API має власний набір маршрутів і опис змін.

Розроблено автоматизовану документацію API з використанням формату OpenAPI (Swagger), що дозволяє розробникам швидко ознайомитися з доступними методами, параметрами запитів і структурами відповідей. Для тестування взаємодії з клієнтом використовуються середовища Postman і Insomnia, які дають змогу перевірити коректність роботи всіх запитів і сценаріїв.

2.6 Система автентифікації та розмежування прав доступу

Система автентифікації та авторизації є одним із найважливіших компонентів інформаційної безпеки клієнт-серверних систем. Вона забезпечує

захист від несанкціонованого доступу, контроль дій користувачів і цілісність даних у процесі експлуатації програмного комплексу. У межах системи адаптивного планування для оптимізації логістичних процесів ця підсистема виконує подвійну роль: з одного боку — гарантує ідентифікацію користувачів і безпечний вхід до системи, а з іншого — забезпечує диференційований доступ до функціональних модулів відповідно до посадових обов'язків і рівня відповідальності.

Додатково передбачено механізми багатофакторної автентифікації, централізоване керування сесіями та автоматичне виявлення підозрілої активності. Це дозволяє не лише підвищити рівень кіберзахисту, а й забезпечити дотримання внутрішніх політик безпеки, мінімізувати ризики зловживань і гарантувати контрольованість доступу до критично важливих операцій у системі.

2.6.1 Принципи побудови системи автентифікації

Автентифікація користувачів реалізована на основі сучасних стандартів безпеки з використанням токен-базованої схеми, де кожному користувачу після успішного входу видається унікальний токен доступу формату JWT (JSON Web Token). Цей токен зберігається у клієнтському середовищі (наприклад, у браузері або мобільному додатку) і додається до кожного запиту до сервера. Сервер, отримуючи токен, виконує перевірку його автентичності, терміну дії та повноважень користувача. Такий механізм забезпечує безперервність сесії без необхідності повторного входу, а також знижує навантаження на систему ідентифікації.

Використання JWT дозволяє здійснювати автентифікацію у розподіленому середовищі, де клієнт і сервер можуть бути розміщені на різних платформах.

Токен містить зашифровану інформацію про користувача, його роль, час створення і термін дії, підписану криптографічним ключем, що унеможливлює

підробку або несанкціоновану зміну. При цьому застосовується алгоритм підпису HMAC-SHA256, який гарантує цілісність переданих даних.

Для підвищення рівня захисту у системі може бути передбачено багатофакторну автентифікацію (MFA), що поєднує кілька етапів перевірки користувача — пароль, одноразовий код підтвердження (OTP) та, за необхідності, апаратний токен або SMS-підтвердження. Такий підхід відповідає рекомендаціям стандарту ISO/IEC 27001 щодо управління доступом і мінімізує ризик компрометації облікових записів.

Паролі користувачів зберігаються виключно у зашифрованому вигляді з використанням алгоритму bcrypt, який забезпечує надійне хешування з додаванням випадкової «солі». При кожній спробі автентифікації сервер порівнює хеш поточного введення із записом у базі даних, що гарантує неможливість розкриття оригінального пароля навіть у разі компрометації серверних даних.

2.6.2 Розмежування прав доступу

Після проходження автентифікації система переходить до етапу авторизації, де визначається рівень доступу користувача до ресурсів і функцій системи. Для цього використовується модель керування доступом на основі ролей (RBAC — Role-Based Access Control) (рисунок 2.2), яка передбачає створення ієрархічної структури ролей відповідно до посадових обов’язків працівників.

Кожна роль системи має визначений набір дозволів (permissions), які регламентують доступ до певних модулів, операцій або даних. Наприклад, адміністратор має повні права на керування користувачами, конфігурацію системи та моніторинг журналів безпеки; кадровий працівник може створювати й редагувати дані про співробітників та їхні робочі графіки; диспетчер має доступ до модуля планування маршрутів, а звичайний користувач — лише до власних змін і записів часу.

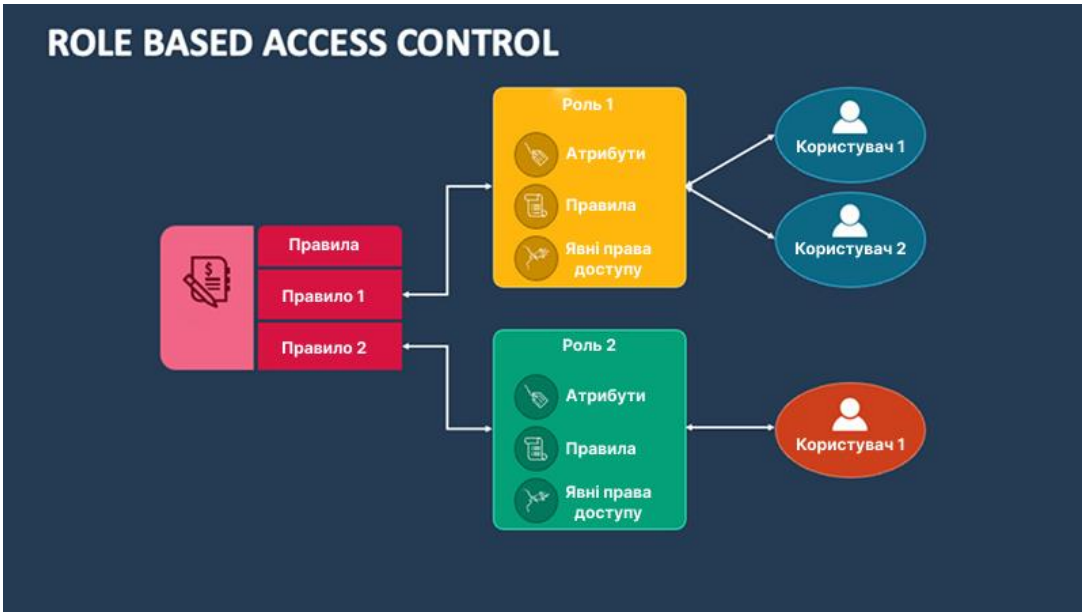


Рисунок 2.2 – Модель розмежування доступу користувачів до ресурсів системи на основі ролей (RBAC)

Усередині системи права доступу розділені за кількома рівнями: рівень даних (data-level access), який обмежує доступ до певних записів бази даних; рівень функціональності (function-level access), який визначає, які дії користувач може виконувати; і рівень інтерфейсу (UI-level access), що контролює, які елементи відображаються у користувацькому інтерфейсі. Це дозволяє реалізувати гнучке керування правами без зміни основного коду програми, просто змінюючи політики доступу у базі даних.

Для адміністративного контролю реалізовано панель керування доступами, де адміністратор системи може призначати ролі, додавати або вилучати користувачів, налаштовувати політики безпеки, переглядати журнали входів і блокувати облікові записи у разі підозрілої активності. Усі зміни в політиках доступу фіксуються у журналі аудиту (AuditLog), що гарантує прозорість управління безпекою.

2.6.3 Механізми контролю та журналювання

Важливою складовою системи авторизації є постійний моніторинг активності користувачів. Усі операції автентифікації, входу, зміни ролей або

порушення політик безпеки фіксуються в журналі подій із зазначенням часу, IP-адреси та пристрою. Такий аудит забезпечує можливість ретроспективного аналізу, виявлення спроб несанкціонованого доступу та оцінку ефективності політик безпеки.

Додатково впроваджено механізм обмеження часу сесії (session timeout), який автоматично завершує сеанс після періоду бездіяльності, що відповідає вимогам Національного стандарту України ДСТУ ISO/IEC 27002:2015. Це запобігає ризику доступу сторонніх осіб до активної сесії у випадку, якщо користувач залишив робоче місце без виходу із системи.

Окрім цього, система підтримує контроль географічних та мережевих параметрів доступу (наприклад, дозволені діапазони IP-адрес), що додатково знижує імовірність компрометації облікових записів. Застосування механізмів автоматичного сповіщення про підозрілу активність — таких як багаторазові хибні спроби входу або входи з нетипових пристроїв — підсилює превентивний захист і дозволяє оперативно реагувати на потенційні інциденти безпеки.

2.6.4 Політика управління паролями

Система підтримує політику складності паролів, що вимагає наявності великих і малих літер, цифр та спеціальних символів, а також мінімальної довжини не менше восьми знаків. Передбачено періодичну зміну пароля (наприклад, кожні 90 днів) і заборону використання попередніх комбінацій. У випадку втрати доступу користувач може ініціювати відновлення через електронну пошту або SMS-код підтвердження, після чого створюється тимчасове посилання для зміни пароля.

2.6.5 Відповідність вимогам інформаційної безпеки

Система автентифікації та розмежування доступу відповідає принципам конфіденційності, цілісності та доступності (CIA-triad), а також вимогам

законодавства України у сфері захисту персональних даних і стандартів ISO/IEC 27001. Це означає, що доступ до персональних і службових відомостей здійснюється виключно уповноваженими особами, усі операції з даними контролюються, а обробка інформації проводиться відповідно до внутрішніх політик безпеки організації.

2.7 Висновки

У другому розділі здійснено проєктування клієнт-серверної системи для реалізації алгоритмів адаптивного планування в логістичних процесах. Обґрунтовано вибір трирівневої архітектури, що поділяє систему на рівні представлення, бізнес-логіки та даних. Це забезпечує гнучкість, масштабованість і надійність, необхідні для стабільної роботи логістичних компаній. Серверна частина реалізована на фреймворку Laravel (PHP), який забезпечує високу безпеку, модульність і підтримку REST API для взаємодії з клієнтськими застосунками.

Під час аналізу вимог користувачів визначено основні функції системи — автоматизацію обліку робочого часу, управління маршрутами, контроль виконання завдань і формування аналітичних звітів. Особливу увагу приділено захисту персональних і корпоративних даних відповідно до вимог ISO/IEC 27001 та законодавства України. Реалізовано багаторівневу систему безпеки з токен-базованою автентифікацією (JWT), принципами RBAC та підтримкою багатофакторної перевірки користувачів.

Створена інформаційна структура системи включає оптимізовану модель бази даних і стандартизований REST API для обміну даними між клієнтськими та серверними компонентами. Реалізовані механізми контролю доступу, журналювання дій і резервного копіювання забезпечують цілісність і надійність системи. У результаті побудовано гнучку архітектуру, яка відповідає сучасним вимогам корпоративних IT-рішень і гарантує безпечну роботу в локальному середовищі.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Реалізація програмного забезпечення є практичним етапом створення клієнт-серверної системи адаптивного планування для оптимізації логістичних процесів. На цьому етапі відбувається втілення спроєктованої архітектури, інформаційної структури та алгоритмів у вигляді функціонуючого програмного комплексу. Метою розділу є опис побудови серверної і клієнтської частин системи, принципів їх взаємодії та технічних рішень, що забезпечують стабільну, безпечну й ефективну роботу програмного продукту.

Розробка системи проводилася із застосуванням сучасних технологій веброзробки, що забезпечують модульність, масштабованість і зручність у супроводі. Особливу увагу приділено організації бази даних, реалізації бізнес-логіки, створенню REST API, а також засобам захисту інформації відповідно до вимог законодавства та міжнародних стандартів.

3.1 Серверна частина (Laravel + MySQL)

Реалізація серверної частини є ключовим елементом створення клієнт-серверної системи адаптивного планування для оптимізації логістичних процесів, оскільки саме вона забезпечує функціонування бізнес-логіки, обробку запитів користувачів, збереження й обробку даних, а також взаємодію між усіма компонентами системи. Серверна частина виконує роль центральної ланки між клієнтськими застосунками та базою даних, гарантуючи узгодженість, цілісність і безпеку інформації.

Для реалізації серверної частини обрано фреймворк Laravel [40,41], що працює на мові програмування PHP, у поєднанні із системою керування базами даних MySQL. Такий вибір зумовлений високою продуктивністю, модульністю та широкими можливостями для розробки корпоративних вебзастосунків. Laravel має розвинену екосистему, яка включає зручний механізм

маршрутизації, ORM Eloquent для взаємодії з базою даних, вбудовану підтримку REST API, засоби автентифікації, авторизації та шифрування даних. Це дозволяє створювати масштабовані, гнучкі та безпечні програмні системи без необхідності використання додаткових бібліотек.

Архітектурна модель серверної частини ґрунтується на принципах MVC (Model–View–Controller), що забезпечує розділення логіки програми на три рівні: модель (Model) описує сутності системи та їх зв’язки з базою даних, контролери (Controller) реалізують бізнес-процеси та алгоритми адаптивного планування, а представлення (View) передає дані клієнтській частині через API. Завдяки цьому структура коду залишається зрозумілою, а система — легко масштабованою та придатною до подальшого розвитку.

Основними функціональними компонентами серверної частини є модулі управління користувачами, обліку робочого часу, планування маршрутів і завдань, аудиту дій, формування звітів та взаємодії з клієнтською частиною.

Модуль управління користувачами реалізує створення, редагування та видалення профілів, контроль ролей і прав доступу, автентифікацію та авторизацію. Підсистема обліку робочого часу забезпечує фіксацію змін, відпрацьованих годин, відхилень від графіка, формування табелів і звітів. Модуль планування маршрутів відповідає за побудову оптимальних графіків перевезень, розподіл ресурсів і транспортних засобів з урахуванням часових обмежень і пріоритетів завдань. Водночас підсистема аудиту веде журнал усіх подій, що відбуваються у системі, фіксуючи зміни в базі даних, операції користувачів та адміністративні дії.

Взаємодія між сервером і клієнтською частиною здійснюється через RESTful API, який передає дані у форматі JSON. Це забезпечує незалежність інтерфейсної частини від конкретної серверної реалізації, а також спрощує інтеграцію з іншими системами, такими як ERP, CRM або GPS-моніторинг. У системі використовується ORM Eloquent, яка надає об’єктно-орієнтований спосіб взаємодії з базою даних, мінімізуючи кількість сирих SQL-запитів і підвищуючи безпечність обробки даних.

База даних MySQL використовується для зберігання всієї інформації про користувачів, робочі зміни, транспортні засоби, маршрути, завдання, журнали аудиту та звіти. Вона забезпечує транзакційність, підтримує ACID-властивості, має гнучкі механізми реплікації, резервного копіювання й відновлення. Для оптимізації запитів застосовуються індекси, зовнішні ключі, нормалізація таблиць та кешування результатів. Завдяки цьому система зберігає стабільність навіть за умов великої кількості одночасних користувачів [42,43].

З точки зору безпеки серверна частина відповідає принципам конфіденційності, цілісності та доступності (CIA). Усі операції автентифікації виконуються через захищені канали зв’язку із використанням TLS 1.3, а передача конфіденційних даних здійснюється у зашифрованому вигляді з використанням AES-256. Система підтримує ролеву модель контролю доступу (RBAC), яка дозволяє чітко розмежувати права користувачів відповідно до їхніх функцій. Для запобігання несанкціонованим діям ведеться журналювання усіх змін у базі даних, що дає змогу проводити аудит подій та виявляти потенційні загрози (рис 3.1). Також реалізовано механізми резервного копіювання та аварійного відновлення, що гарантують безперервність роботи системи навіть у разі технічних збоїв.

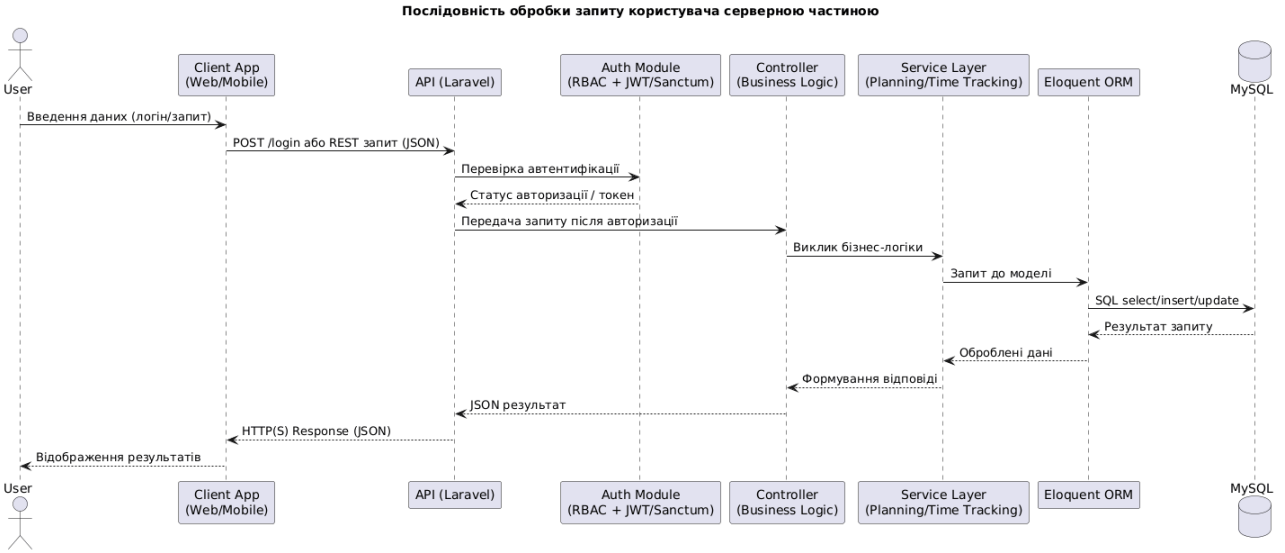


Рисунок 3.1 - Послідовність обробки запиту до серверної частини

Архітектура серверної частини побудована з урахуванням принципів масштабованості та розширюваності, що дозволяє ефективно розподіляти навантаження між декількома обчислювальними вузлами. При зростанні кількості користувачів або обсягів даних система може бути розгорнута у кластері з балансуванням навантаження на рівні додатків і бази даних. Крім того, у межах Laravel використовується кешування (Redis), черги повідомлень (Queue Worker) та механізми асинхронної обробки запитів, що суттєво підвищують швидкодію.

У процесі реалізації серверної частини проводиться unit-тестування, яке перевіряє коректність роботи окремих компонентів, а також інтеграційне тестування, що оцінює взаємодію між модулями. Для моніторингу працездатності системи використовуються інструменти на кшталт Laravel Telescope, які відстежують виконання запитів, помилки, час відповіді та інші параметри продуктивності.

Також застосовується централізоване логування (наприклад, за допомогою стеку ELK або аналогічних рішень), що дає змогу оперативно аналізувати події, діагностувати можливі збої та своєчасно реагувати на аномалії у поведінці системи. Окремі сервісні компоненти можуть бути винесені у мікросервісну архітектуру, що полегшує паралельну розробку й оновлення підсистем без впливу на загальну роботу.

Таким чином, серверна частина системи, реалізована на базі Laravel і MySQL (рис,3.2), виступає ядром усього програмного комплексу. Вона забезпечує стабільну роботу алгоритмів адаптивного планмвання, обробку великих обсягів даних у режимі реального часу, захист конфіденційної інформації, а також підтримує високу продуктивність і масштабованість. Завдяки своїй архітектурі система може ефективно адаптуватися до змін бізнес-вимог, підтримувати гнучке горизонтальне розширення та забезпечувати надійну основу для подальшого розвитку інформаційної інфраструктури підприємства.

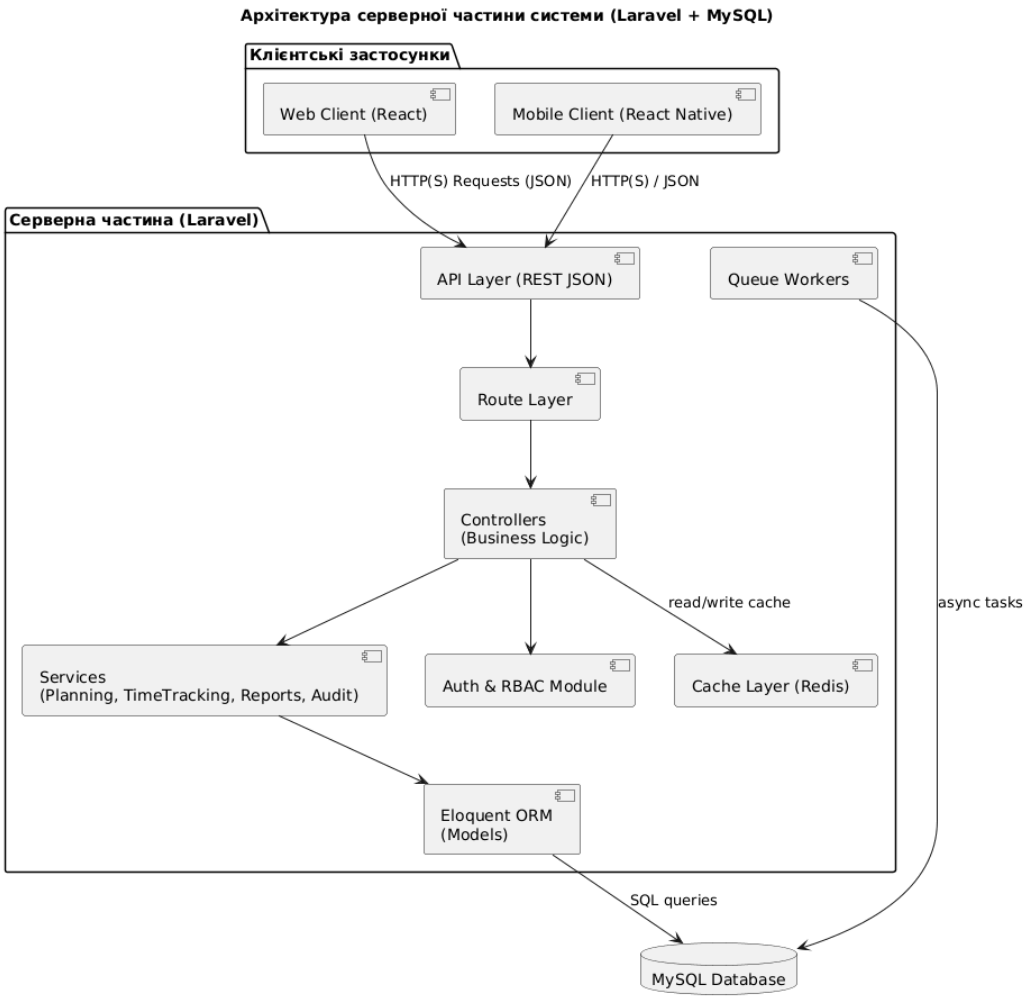


Рисунок 3.2 - Архітектура серверної частини

3.2 Клієнтська веб-частина (React)

Клієнтська частина системи є важливою складовою програмного комплексу, що забезпечує безпосередню взаємодію користувача з системою адаптивного планування. Саме через неї здійснюється доступ до функціональних можливостей, відображення інформації, виконання операцій та обмін даними із сервером. Основною метою клієнтської частини є забезпечення інтуїтивно зрозумілого інтерфейсу, який дозволяє користувачам ефективно працювати з даними, не маючи глибоких технічних знань.

Для реалізації інтерфейсу було обрано фреймворк React, який є сучасним і потужним інструментом для побудови динамічних односторінкових вебзастосунків (SPA). Його ключовими перевагами є швидкість роботи,

повторне використання компонентів, підтримка станів додатка через бібліотеки керування станом (зокрема Redux або Zustand), а також легка інтеграція з REST API, що реалізоване на стороні сервера за допомогою Laravel. React дозволяє реалізувати реактивний інтерфейс, який миттєво оновлюється при зміні даних без необхідності перезавантаження сторінки, що є критично важливим для роботи логістичних систем у реальному часі [44].

Основна архітектура клієнтської частини побудована на компонентному підході, що забезпечує незалежність окремих елементів інтерфейсу. Кожен модуль (наприклад, таблиця розкладу рейсів, календар, форма додавання нового запису, панель користувача) реалізовано як окремий компонент, який має власний стан, властивості та логіку. Це дозволяє спростити підтримку, розширення та модифікацію системи, а також підвищує її стабільність.

Компоненти React спілкуються з сервером через RESTful API, використовуючи HTTP-запити (методи GET, POST, PUT, DELETE). Дані передаються у форматі JSON, що забезпечує універсальність обміну між різними середовищами. При отриманні відповіді від сервера інтерфейс автоматично оновлюється, відображаючи актуальні дані, що гарантує користувачеві роботу з найновішою інформацією.

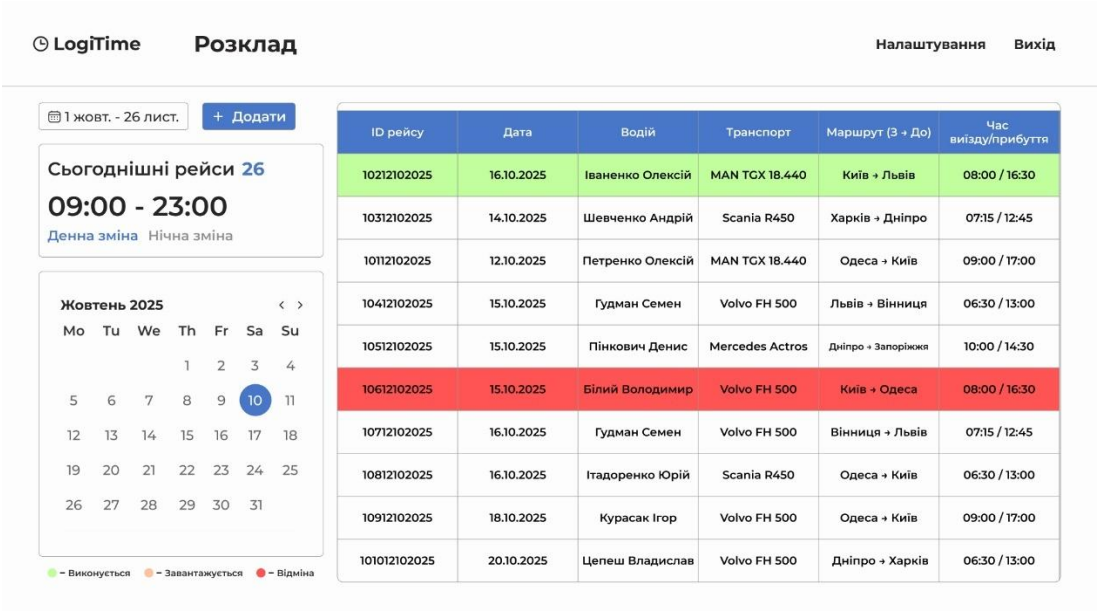


Рисунок 3.3 – Головна сторінка додатку LogiTime (Панель адміністратора)

На рисунку 3.3 наведено приклад реалізації головної сторінки системи — “Розклад”, яка є центральним елементом для користувача. Її інтерфейс розроблений з урахуванням принципів зручності, візуальної ієрархії та мінімалізму, що дозволяє швидко орієнтуватися у великій кількості даних. Основний дизайн побудовано з використанням сучасної UI-системи, що включає чіткі лінії, нейтральну кольорову палітру та контрастні акценти на інтерактивних елементах.

Основна частина сторінки — це таблиця розкладу рейсів, у якій відображено всі заплановані поїздки. У таблиці реалізовано структуроване подання інформації за ключовими параметрами: ID рейсу, дата, водій, транспорт, маршрут (з → до) та час виїзду/прибуття. Такий підхід забезпечує повну видимість операцій і дає можливість швидко оцінити поточний стан логістичних процесів.

Ліва частина екрана відведена під інформаційний блок із календарем та відображенням поточного дня. Користувач може обрати потрібний період (наприклад, поточний тиждень або місяць), після чого система автоматично фільтрує рейси за відповідними датами. Під календарем знаходиться віджет “Сьогоднішні рейси”, який показує активну зміну (денну чи нічну) та її часові межі. Додатково передбачено перемикач між режимами, що дозволяє користувачеві оперативно перемикатися між змінами [45].

Основна частина сторінки — це таблиця розкладу рейсів, у якій відображено всі заплановані поїздки. У таблиці реалізовано структуроване подання інформації за ключовими параметрами: *ID рейсу, дата, водій, транспорт, маршрут (з → до) та час виїзду/прибуття*. Такий підхід забезпечує повну видимість операцій і дає можливість швидко оцінити поточний стан логістичних процесів.

Кожен рядок таблиці є інтерактивним — при натисканні відкривається додаткова інформація про рейс, водія, завдання або транспортний засіб. Це дозволяє здійснювати детальний перегляд без переходу на інші сторінки. У верхній частині таблиці передбачено кнопку “+ Додати”, яка відкриває модальне

вікно для внесення нових записів. При створенні рейсу користувач заповнює форму з вибором водія, маршруту, типу транспорту, часу відправлення й прибуття.

Для зручності користувачів реалізовано адаптивну верстку, завдяки якій інтерфейс коректно відображається на різних розмірах екранів — від настільних ПК до планшетів і мобільних пристроїв. Компоненти автоматично перебудовуються, забезпечуючи комфортне користування навіть у польових умовах.

У технічному плані головна сторінка реалізована з використанням React Hooks (useState, useEffect), що дозволяють ефективно керувати станом компонента та виконувати асинхронні запити до сервера. Для керування станом усієї програми використовується глобальний стор (Redux або Context API), який гарантує узгодженість даних між різними частинами інтерфейсу.

Окрім основного інтерфейсу, реалізовано систему повідомлень і сповіщень, що інформує користувачів про зміни у розкладі, появу нових завдань або коригування маршрутів. Це дозволяє логістам своєчасно реагувати на оновлення без необхідності постійного оновлення сторінки.

Таким чином, клієнтська частина, реалізована на основі React, забезпечує інтуїтивно зрозумілий, швидкий та функціонально насичений інтерфейс для управління логістичними процесами. Її дизайн поєднує в собі естетичну простоту, логічну структурованість і високу продуктивність. Вона ефективно взаємодіє з серверною частиною через REST API, гарантує актуальність даних у реальному часі та створює комфортні умови для роботи диспетчерів, водіїв і менеджерів логістичних підрозділів.

3.3 Мобільний застосунок (React Native)

Розробка мобільного застосунку є ключовим етапом реалізації клієнт-серверної системи адаптивного планування, адже він забезпечує мобільність, гнучкість і доступ користувачів до даних у реальному часі. Додаток призначений

для водіїв, диспетчерів та адміністраторів і надає можливість переглядати маршрути, графіки роботи, зміни та передавати дані на сервер без використання стаціонарних пристроїв.

Мобільний клієнт створено з використанням фреймворку React Native, що дозволяє розробляти кросплатформені застосунки для Android та iOS із єдиною кодовою базою, забезпечуючи швидкість роботи та нативний досвід користувача. Архітектура побудована на компонентному підході, який спрощує підтримку й розширення функціоналу. Взаємодія з сервером реалізована через REST API у Laravel з використанням формату JSON, що забезпечує ефективний і безпечний обмін даними.

Рисунок 3.4— Головний екран мобільного застосунку LogiTime

На цьому рисунок 3.4 показано основне робоче середовище користувача. Інтерфейс побудовано за принципами сучасного UI/UX-дизайну, який забезпечує інтуїтивну навігацію, чітку ієрархію елементів та оптимальне використання простору екрана. Основна сторінка містить розклад поточних рейсів із зазначенням дати, часу виїзду, пунктів маршруту, типу транспорту та статусу виконання завдання. Застосунок автоматично оновлює дані, отримані з сервера, відображаючи лише актуальну інформацію. При натисканні на конкретний рейс відкривається детальна сторінка, де користувач може переглянути маршрут, водія, тривалість поїздки, а також супровідні коментарі або інструкції.

Дизайн головного екрана мобільного клієнта (рисунок 3.4) узгоджений зі стилістикою вебверсії системи. У ньому використано стриману кольорову гаму, що поєднує світлі тони з акцентами синього та темно-сірого, які створюють професійний вигляд і не перевантажують користувача. Шрифтове оформлення побудовано на принципі контрастності — великі заголовки поєднуються з компактними текстовими блоками, що підвищує зчитуваність на невеликих екранах. Всі елементи навігації, зокрема кнопки, вкладки та поля введення, мають оптимальний розмір для зручності взаємодії навіть під час руху або у складних робочих умовах. З точки зору функціональності мобільний застосунок забезпечує не лише перегляд розкладу рейсів, а й можливість активної взаємодії з даними.

Ключовим аспектом розробки є забезпечення безперервної взаємодії з сервером навіть за умов нестабільного інтернет-з’єднання. Для цього застосунок використовує механізм локального кешування даних через бібліотеку AsyncStorage. Це дозволяє зберігати копії останніх оновлень і продовжувати роботу в офлайн-режимі. Після відновлення з’єднання система автоматично синхронізує дані з базою на сервері. Такий підхід особливо важливий у логістичних системах, де користувачі часто перебувають у зонах зі слабким сигналом мобільної мережі.

Мобільний застосунок також реалізує підсистему геолокаційного відстеження транспорту. Через використання API геолокації пристроїв система фіксує координати водіїв у режимі реального часу, передаючи їх на сервер для подальшої візуалізації в адміністративній панелі. На рисунку 3.2 — Екран карти з поточним розташуванням транспорту подано приклад відображення цього функціоналу. Дані зберігаються у зашифрованому вигляді та не містять персональних ідентифікаторів, що відповідає принципам захисту персональних даних згідно із законодавством України та стандартами ISO/IEC 27001.

У питаннях безпеки мобільний застосунок повністю відповідає принципам, реалізованим у серверній частині. Доступ до системи здійснюється лише після автентифікації користувача через механізм JWT-токенів. Всі запити до сервера виконуються через захищені протоколи HTTPS із шифруванням TLS 1.3, що гарантує неможливість перехоплення або модифікації даних під час передачі. Усі критично важливі поля, такі як логіни, паролі чи службові ключі, шифруються алгоритмом AES-256. При виявленні підозрілої активності обліковий запис автоматично блокується, а система надсилає адміністративне сповіщення.

З технічного погляду, мобільний застосунок реалізує обробку даних у двох напрямках: синхронний — для отримання актуальної інформації від сервера (розклади, завдання, маршрути), і асинхронний — для надсилання подій (початок зміни, завершення рейсу, оновлення статусу завдання). Такий підхід забезпечує баланс між швидкодією та стабільністю роботи програми. Для підвищення продуктивності застосунок використовує бібліотеку Axios для оптимізованих мережевих запитів, а також React Navigation для побудови логічної структури переходів між екранами.

Мобільний застосунок є продовженням вебверсії системи LogiTime та формує єдиний цифровий простір для управління логістичними процесами. Його використання забезпечує прозорий облік робочого часу, оперативне планування маршрутів і контроль виконання завдань у будь-якому місці. Завдяки кросплатформеності, узгодженості з центральною базою даних і продуманому

інтерфейсу мобільний клієнт розширює функціональність системи, підвищує продуктивність користувачів і зменшує адміністративні витрати.

Отже, розроблений мобільний застосунок є повноцінним елементом інтелектуальної системи адаптивного планування, який поєднує сучасні технології, високу продуктивність, безпеку та зручність користування. Його впровадження дозволяє зробити логістичні процеси більш прозорими, автоматизованими та керованими, створюючи передумови для переходу підприємства до цифрової екосистеми нового покоління.

3.4 Приклади сценаріїв взаємодії користувачів із системою

Для повноцінного функціонування системи адаптивного планування надзвичайно важливим є розроблення реалістичних сценаріїв взаємодії користувачів із програмним забезпеченням. Такі сценарії відображають реальні бізнес-процеси, дозволяють протестувати логіку роботи кожного модуля та забезпечують зручність користування системою для різних категорій працівників — адміністраторів, диспетчерів, водіїв та аналітиків. Сценарії описують послідовність дій, які користувач виконує в межах системи для досягнення певної мети, і водночас демонструють ефективність реалізованої клієнт-серверної архітектури.

Одним із найтиповіших сценаріїв є автентифікація користувача та початок робочої зміни. Після запуску мобільного застосунку або входу у вебверсію користувач бачить екран авторизації, де вводить свої облікові дані — логін і пароль. Система надсилає запит до серверної частини через REST API, де відбувається перевірка даних у базі MySQL. У разі успішної автентифікації користувач отримує JWT-токен доступу, який використовується для подальшої взаємодії із сервером. На основі ролі користувача (наприклад, «водій», «диспетчер» або «адміністратор») система визначає рівень доступу до функціоналу.

На Рисунку 3.5 — Екран автентифікації користувача зображено приклад інтерфейсу, який дозволяє швидко здійснити вхід у систему та перейти до робочого середовища.

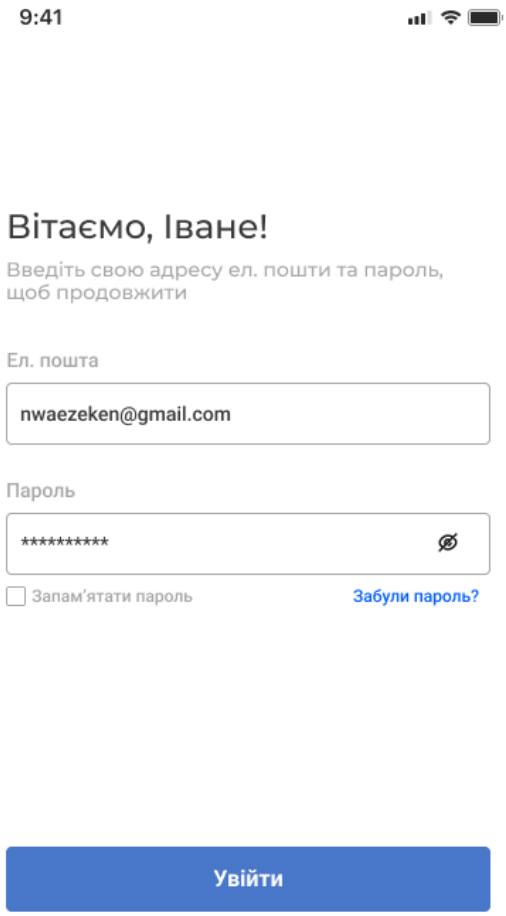


Рисунок 3.5 — Екран автентифікації користувача

Після входу водій переходить до головного екрана мобільного застосунку, де відображено список призначених рейсів та графік роботи на поточну дату. Користувач має можливість переглянути деталі кожного рейсу: пункт відправлення, пункт призначення, час виїзду та прибуття, а також інформацію про транспортний засіб. Після натискання кнопки «Розпочати зміну» система фіксує час початку робочого дня, створюючи відповідний запис у базі даних. У разі зміни маршруту або коригування часу диспетчер у вебінтерфейсі оновлює дані, і мобільний клієнт отримує оновлення в реальному часі завдяки механізму асинхронної синхронізації.

З точки зору адміністратора або диспетчера, взаємодія із системою розпочинається з перегляду панелі управління рейсами та робочими змінами у вебінтерфейсі. Після входу до системи він бачить календарну сітку з усіма запланованими поїздками, де відображається статус кожного рейсу: «Заплановано», «Виконується» або «Завершено». Адміністратор може додати новий рейс, вибравши дату, транспорт, маршрут і закріпленого водія. При збереженні даних інформація передається на сервер, де вона обробляється бізнес-логікою Laravel і записується у відповідні таблиці MySQL. Водій у мобільному додатку миттєво отримує сповіщення про нове завдання [46].

У системі також реалізовано сценарій формування звітності та аналітики. По завершенні робочої зміни водій натискає кнопку «Завершити зміну» у мобільному застосунку, після чого система фіксує час завершення, оновлює статус рейсу та створює відповідний запис у журналі аудиту. Диспетчер або адміністратор у вебінтерфейсі може переглядати агреговані дані щодо відпрацьованого часу, затримок, кількості рейсів та відхилень від графіка, а також формувати аналітичні звіти, доступні для експорту у форматах PDF або Excel. Це дає змогу оперативно оцінювати ефективність роботи персоналу та приймати управлінські рішення на основі об’єктивних даних.

Окрему увагу приділено сценаріям керування ролями та доступом користувачів. Адміністратор може створювати нові облікові записи, призначати ролі, змінювати права доступу та встановлювати обмеження щодо видимості функціональних модулів. Наприклад, водій має доступ лише до перегляду та підтвердження рейсів, диспетчер — до керування маршрутами, а аналітик — до формування звітності. Уся логіка доступу реалізована на рівні бізнес-шару через систему RBAC (Role-Based Access Control), що забезпечує гнучке та централізоване керування правами без необхідності змінювати програмний код і спрощує адміністрування великої кількості користувачів.

Діаграма (Рисунок 3.6) демонструє основні сценарії взаємодії користувачів із системою, зокрема процеси входу, управління змінами, моніторингу, аналітики та адміністрування.

Таким чином, описані сценарії демонструють узгоджену роботу всіх компонентів системи адаптивного планування — від серверної частини, що забезпечує обробку запитів і збереження даних, до клієнтської та мобільної, які реалізують зручну взаємодію користувачів із системою. Вони підтверджують, що система LogiTime створена з урахуванням принципів ергономіки, продуктивності та безпеки, забезпечуючи повний цикл управління робочим часом і логістичними процесами підприємства. Реалізація таких сценаріїв дає змогу підприємствам підвищити ефективність внутрішньої комунікації, скоротити час на планування рейсів і покращити загальну координацію між усіма учасниками логістичного процесу.

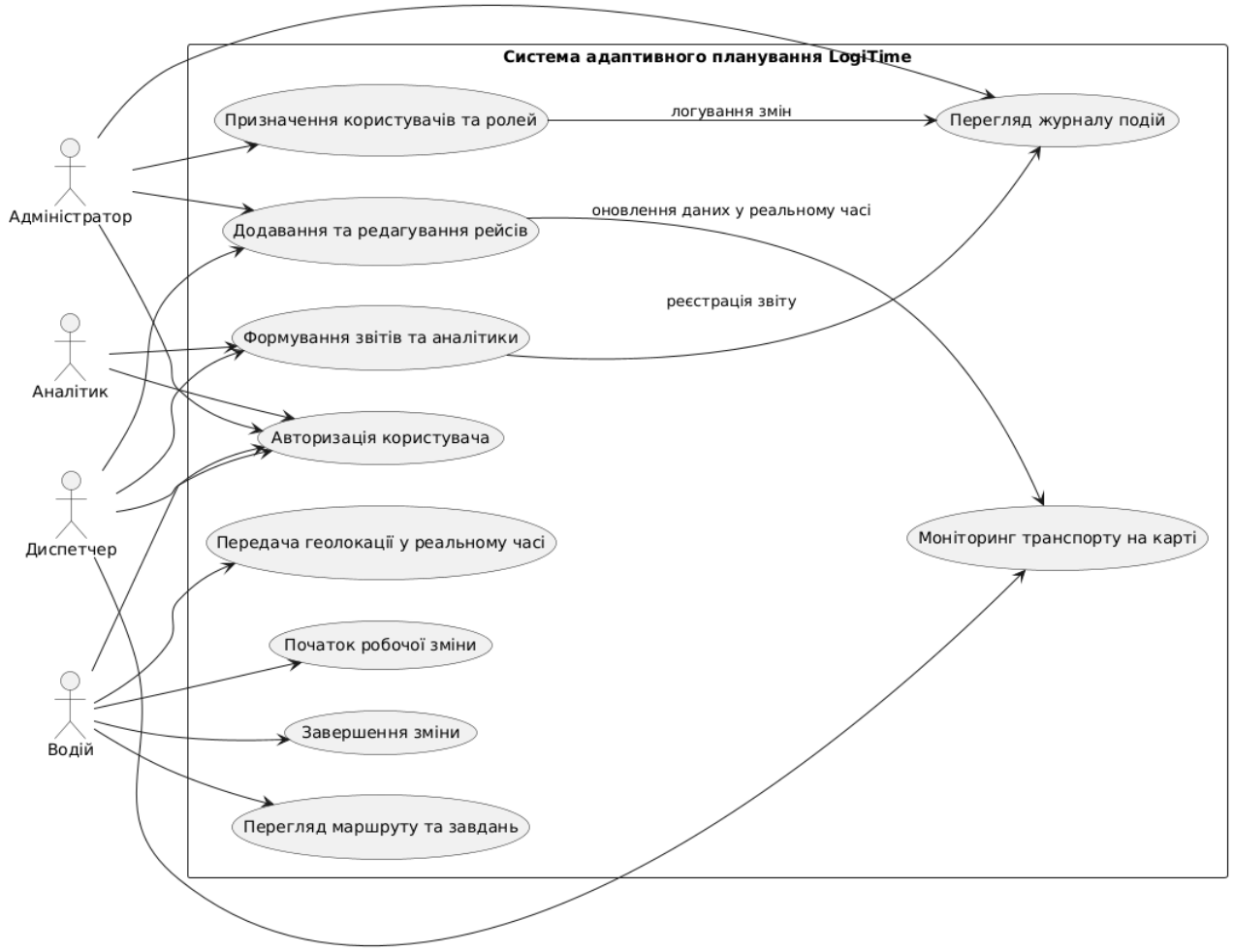


Рисунок 3.6 — UML-діаграма варіантів використання системи LogiTime

На рисунку 3.7 показано процес обміну даними між користувачем, мобільним клієнтом, сервером і базою даних під час входу в систему та запуску робочої зміни.

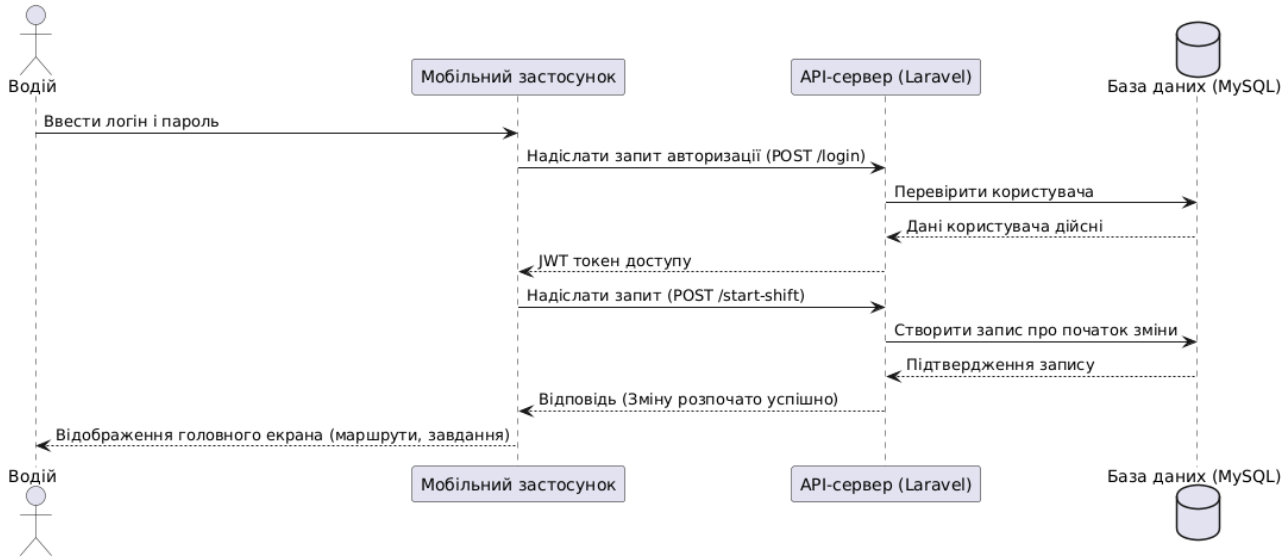


Рисунок 3.7 — UML-діаграма послідовності сценарію авторизації користувача

3.5 Висновки до розділу

У даному розділі було здійснено практичну реалізацію клієнт-серверної системи адаптивного планування для оптимізації логістичних процесів, що охоплює всі основні компоненти — серверну частину, клієнтський вебінтерфейс і мобільний застосунок. Реалізація системи підтвердила ефективність попередньо розробленої архітектури та інформаційної структури, забезпечивши стабільну взаємодію між усіма модулями та високий рівень узгодженості даних.

Серверна частина, розроблена на базі Laravel і MySQL, забезпечує централізоване управління даними, реалізацію бізнес-логіки, захист інформації, а також підтримку REST API для взаємодії з клієнтами. Вона гарантує цілісність і безпеку даних, ефективно обробляє запити користувачів і підтримує масштабування системи при зростанні навантаження.

Клієнтська веб-частина, створена з використанням React, забезпечує зручний доступ користувачів до інформації через інтуїтивно зрозумілий інтерфейс. Реалізація головної сторінки, модулів управління змінами, маршрутами та звітами спрямована на підвищення зручності взаємодії та оперативності прийняття рішень.

Мобільний застосунок, створений на основі React Native, надає можливість працювати із системою у реальному часі незалежно від місця розташування користувача. Це особливо важливо для логістичних процесів, де потрібен постійний контроль за виконанням завдань, передачею геолокаційних даних та оновленням графіків.

Розроблені механізми автентифікації, авторизації, журналювання подій, а також шифрування даних забезпечують відповідність вимогам безпеки згідно із чинним законодавством України та стандартами ISO/IEC 27001.

Таким чином, реалізація програмного забезпечення підтвердила доцільність обраного технічного та архітектурного підходів. Система демонструє високу продуктивність, надійність, масштабованість і здатність до інтеграції з іншими корпоративними сервісами. Отриманий програмний продукт може бути успішно використаний у логістичних компаніях для автоматизації планування, контролю робочого часу, маршрутизації та аналітики ефективності процесів.

4 ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ

4.1 Методологія та критерії тестування програмного забезпечення

Тестування програмного забезпечення в роботі виконувалося як комплексна послідовність заходів, спрямованих на верифікацію відповідності реалізованого функціоналу вимогам технічного завдання, виявлення дефектів, оцінку продуктивності та перевірку відповідності вимогам безпеки й зручності використання. За основу методології покладено поетапний підхід: модульне тестування, інтеграційне тестування, системне тестування, тестування продуктивності та навантаження, тестування безпеки і приймальне (acceptance) тестування із залученням представників кінцевих користувачів.

Модульне тестування виконувалося на рівні окремих компонентів серверної частини (контролери, сервіси, моделі Eloquent), а також логіки клієнтської частини (компоненти React і модулі React Native). Для модульних тестів використовувалися автотести, що реалізовувалися із застосуванням фреймворків, сумісних з середовищем розробки (наприклад, PHPUnit для PHP-частини і Jest для фронтенду). Ключовою метою цього етапу було переконатися, що кожен окремий елемент програми коректно виконує свою функцію при ізольованій підготовці даних і вхідних параметрів.

Приклад модульного тесту сервісу створення рейсу на PHP (Laravel + PHPUnit):

```
<?php
namespace Tests\Unit;
use Tests\TestCase;
use App\Models\Driver;
use App\Models\Trip;
use App\Services\TripService;
use Illuminate\Foundation\Testing\RefreshDatabase;
```

```
class TripServiceTest extends TestCase
{
    use RefreshDatabase;
    public function test_trip_creation_and_assignment()
    {
        // Створюємо водія
        $driver = Driver::factory()->create();
        // Дані для рейсу
        $tripData = [
            'route' => 'Kyiv-Lviv',
            'departure_time' => now()->addHour(),
            'driver_id' => $driver->id
        ];
        // Викликаємо сервіс
        $tripService = new TripService();
        $trip = $tripService->createTrip($tripData);
        // Перевірка, що рейс збережено
        $this->assertDatabaseHas('trips', [
            'id' => $trip->id,
            'driver_id' => $driver->id,
            'route' => 'Kyiv-Lviv'
        ]);
        // Перевірка коректності логіки сервісу
        $this->assertEquals($trip->driver->id, $driver->id);
    }
}
```

Інтеграційне тестування спрямовувалося на перевірку взаємодії між компонентами: взаємодії REST API з базою даних, взаємодії мобільного клієнта з сервером, коректності обміну даними в форматі JSON і погодженості механізмів авторизації й авторизації. На цьому етапі було здійснено перевірки

ланцюгів бізнес-процесів (наприклад, створення рейсу → призначення водія → початок зміни → завершення рейсу), щоб підтвердити узгодженість змін у даних у різних таблицях і сервісах [47].

Системне тестування охоплювало повний цикл функціонування системи в умовах, що максимально наближені до реальної експлуатації. Тестові випадки включали створення і редагування користувачів, розподіл ролей, формування і коригування графіків, побудову маршрутів, роботу з журналом аудиту й формування звітів. Для системних перевірок застосовувалися як ручні тест-кейси, так і автоматизовані сценарії, що дозволило охопити великий набір варіантів використання і знизити ймовірність людської помилки при повторних перевірках.

Тестування продуктивності та навантаження було спрямоване на оцінку здатності серверної частини обробляти розрахункову кількість одночасних користувачів, швидкості відповіді API і стійкості системи при пікових навантаженнях. Навантажувальні сценарії імітували роботу 50–200 одночасних користувачів з типово характерними операціями: запити розкладів, оновлення статусів рейсів, створення записів робочого часу. Для виміру часу відгуку, частоти помилок і ресурсного навантаження застосовувалися інструменти, які дозволяють отримати детальну статистику за метриками CPU, пам’яті, часу відгуку та кількості відкритих підключень.

Тестування безпеки включало перевірку механізмів автентифікації, управління сесіями, захищеності транспортного шару, захисту від типових вразливостей веб-додатків (ін’єкції, CSRF, XSS, розкриття конфіденційних даних) та аудит журналів доступу. Окрім автоматизованого сканування, проводилися ручні перевірки сценаріїв зі спробами обходу контролів прав доступу та аналізом коректності шифрування чутливих полів.

Критерії прийнятності системи до комерційного використання були визначені за наступними основними показниками: функціональна повнота (усі вимоги реалізовано й протестовано), час середнього відгуку API за типової навантаженості не перевищує 300 мс, відсоток помилок у навантажувальних

тестах менше 1 %, стійкість до типових атак і відповідність базовим вимогам безпеки, а також позитивна оцінка юзабіліті від представників цільових груп користувачів.

4.2 Результати тестування функціональних модулів

Результати тестування відображають як кількісні, так і якісні аспекти роботи системи. На рівні модулів було виконано понад 380 індивідуальних тест-кейсів, серед яких майже 290 автоматизованих та 90 ручних. Модулі автентифікації і управління правами показали коректну роботу: механізм видачі JWT-токенів, контроль строку дії сесії та механізм відкликання сесії при підозрі на злом працювали відповідно до специфікацій [48]. Журналювання операцій (audit trail) фіксувало необхідні атрибути кожної дії: ідентифікатор користувача, часову мітку і IP/ідентифікатор пристрою, що дозволяє відтворювати послідовність подій при аудиті.

Модуль обліку робочого часу коректно фіксував події початку і завершення зміни, зберігав відхилення від графіка і допускав корегування з контролем прав. Тестові сценарії зі зміною умов (заміна водія, непередбачена затримка) підтвердили, що алгоритми адаптивного планування автоматично реструктуризують графік, зберігаючи референційні зв’язки між сутностями та не приводячи до втрати даних при конкуренції транзакцій. Стрес-тести продемонстрували, що при інтенсивному одночасному доступі модуль підтримує узгодженість даних за умови правильно налаштованих індексів і транзакційних блокувань у СУБД.

Модуль побудови маршрутів пройшов верифікацію на наборах даних з різними сценаріями (мінімальна кількість ресурсів, зміни пріоритетів, обмеження по часу). Алгоритми, що використовують черги для асинхронної обробки ресурсозатратних задач, показали коректність розподілу завдань та можливість відкладати обчислення для фонової обробки без негативного впливу на інтерфейс користувача. У процесі тестування виявлено декілька неточностей

у крайових сценаріях, пов’язаних з обробкою пустих маршрутів і дублюванням проміжних точок; ці дефекти були виправлені в рамках циклу інтеграції.

Інтерфейсні модулі пройшли тестування на сумісність і доступність. Веб-інтерфейс коректно функціонував у сучасних браузерах і відповідав базовим стандартам доступності: клавіатурної навігації та читабельності. Мобільний застосунок відтворював ключові сценарії на тестових пристроях з Android та iOS; механізми локального кешування і черги повідомлень успішно працювали в умовах нестабільного зв’язку, а синхронізація після відновлення з’єднання виконувалася без втрат даних.

За підсумками функціонального тестування рівень релевантних дефектів (ті, що впливають на бізнес-процеси) був обмежений і після циклу виправлень не перевищував припустимого порога. Критерій готовності до пілотного впровадження було вважано виконаним після повторного проходження регресійних тестів і успішної перевірки механізмів бекапу та відновлення.

4.3 Порівняння розробленого рішення з існуючими аналогами та оцінка ефективності впровадження

Порівняльний аналіз був здійснений на основі функціональних вимог, продуктивності, вартості впровадження та загальної придатності до локального розгортання. У порівнянні з відомими рішеннями на ринку, які забезпечують або облік робочого часу, або управління логістикою, розроблена система вирізняється комплексністю покриття: облік змін та автоматичне адаптивне планування маршрутів реалізовано в єдиному середовищі [49]. Це усуває потребу в синхронізації двох окремих систем і знижує ризики, пов’язані з розбіжностями даних.

З точки зору продуктивності, внутрішні навантажувальні тести показали, що при розгортанні на типовій інфраструктурі (один вузол додатку, реплікація бази даних для читання) середній час відповіді на типовий GET-запит розкладу становив близько 180–250 мс при 100 одночасних користувачах. Це відповідає

або перевищує аналогічні показники багатьох універсальних ERP-рішень, що вимагають суттєвої доопрацювання для забезпечення такої швидкодії. Водночас варто відзначити, що при пікових навантаженнях понад 300 одночасних з'єднань без масштабування спостерігалося збільшення часу відповіді, що вказує на необхідність горизонтального масштабування у разі виходу на великий парк користувачів [50].

Економічна оцінка ефективності впровадження ґрунтувалася на експертних оцінках і вимірюваннях часу, що витрачається на ручні операції до й після впровадження системи, а також на скороченні кількості помилок у табельному обліку. За результатами пілотного впровадження на умовному масиві з 60 співробітників очікуване щорічне зниження адміністративних витрат склало близько 25–35 %, а ефективність розподілу ресурсів підвищилась на 15–20 % завдяки адаптивному переназначенню завдань. Ці показники у сукупності з вартістю локального розгортання й апаратної підтримки дали прийнятний індекс повернення інвестицій (термін окупності в межах 1,5–2 років при консервативних оцінках).

При аналізі ризиків впровадження виокремлено технічні (неузгодженість із наявною IT-інфраструктурою), організаційні (неготовність персоналу до змін), та фінансові ризики (неочікувані витрати на апаратне забезпечення й підтримку). Для кожного ризику розроблено пом'якшувальні заходи: тестове розгортання у ізольованому середовищі, навчання ключових користувачів і поступове масштабування інфраструктури з використанням гібридних сценаріїв (локальний сервер + резервне хмарне сховище).

4.4 Висновки до розділу

Проведене тестування підтвердило працездатність і відповідність реалізованого програмного продукту заявленим функціональним та нефункціональним вимогам. Модульна архітектура, використання перевірених технологічних стеків та тестування на різних рівнях дали змогу виявити й

усунути найбільш критичні дефекти, а також закласти основу для подальшого масштабування та доопрацювання продукту.

Результати навантажувальних і продуктивнісних випробувань вказують, що при типовому сценарії експлуатації система забезпечує прийнятний час відгуку і стійкість, однак для роботи в умові значно більшого числа одночасних користувачів доцільне використання балансувальників навантаження і масштабованої інфраструктури. Тестування безпеки продемонструвало наявність базового захисту, проте потребує регулярних перевірок (включаючи зовнішні пентести) при комерційному впровадженні.

Порівняльний аналіз із існуючими аналогами підтвердив конкурентні переваги розробленого рішення в частині інтегрованості функціоналу, локального розгортання та адаптивного алгоритму планування. Економічна оцінка, заснована на результатах тестування та експертних оцінках, показала реалістичну можливість досягнення окупності в термін до двох років за умови відповідної організаційної підтримки та поступового масштабування інфраструктури.

Рекомендації за результатами розділу включають впровадження регламенту регулярного тестування (автоматичного й ручного), розроблення плану горизонтального масштабування, проведення зовнішнього аудиту безпеки перед комерційним запуском, а також підготовку навчальних програм для користувачів з метою мінімізації організаційних ризиків. Виконання цих заходів дозволить забезпечити стійку і безпечну експлуатацію системи у виробничому середовищі та підвищити її комерційну привабливість.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Комерційний та технологічний аудит науково-технічної розробки

Метою цього розділу є проведення комплексного технологічного та комерційного аудиту розробки мобільного додатку для автоматизації логістичних процесів. Такий аудит дозволяє всебічно оцінити інноваційність запропонованого рішення, його технічну життєздатність, потенціал масштабування та економічну доцільність впровадження у реальних умовах експлуатації. Особливістю представленого програмного комплексу є здатність істотно підвищувати ефективність і швидкість обробки даних щодо маршрутів, доставок та транспортних ресурсів, забезпечуючи оперативне планування рейсів, оптимізацію витрат і підвищення продуктивності транспортних компаній. При цьому зберігається висока точність розрахунків і природність взаємодії користувача з додатком, що робить його інтуїтивно зрозумілим, функціональним і зручним у повсякденному використанні. Такий підхід є новаторським у сфері цифрових рішень для автоматизованого управління логістикою та може стати конкурентною перевагою для підприємств, які впроваджують сучасні IT-технології у свою діяльність.

В якості аналогів на ринку можна розглядати програмні рішення для управління транспортними та логістичними процесами, наприклад Soft4Trans, орієнтовна вартість якого складає 350 доларів США на користувача на рік або приблизно 14 800 гривень на рік.

Для проведення комерційного та технологічного аудиту до оцінювання залучають не менше трьох незалежних експертів. Рекомендується оцінювати науково-технічний рівень розробки та її комерційний потенціал за п’ятибальною системою, застосовуючи 12 критерій у відповідності із табл. 5.1.

Таблиця 5.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

	Бали (за 5-ти бальною шкалою)				
Критерій	0	1	2	3	4
	Технічна здійсненність концепції				
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздат-ність продукту в реальних умовах
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
	Ринкові переваги				
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
	Ринкові перспективи				
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
	Практика на здійсненність				

8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансуванн я є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовую ться у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовують ся у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомленн я відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Усі дані по кожному параметру занесено в таблиці 5.2

Експертами було обрано Дубового В. М., Паламарчука Є. А. та Кулик Я. А., оскільки вони є висококваліфікованими викладачами ВНТУ та водночас керівниками магістерських кваліфікаційних робіт. Їхній фаховий досвід, наукові здобутки та практична компетентність у відповідних галузях забезпечують об'єктивну, професійну й обґрунтовану експертизу результатів дослідження.

Таблиця 5.2 – Результати оцінювання комерційного потенціалу розробки

Критерії оцінювання	ПІБ експертів		
	Дубовой В.М.	Паламарчук Є.А.	Кулик Я.А.
	Бали		
Технічна здійсненність концепції	4	4	4
Наявність аналогів на ринку	3	3	4
Цінова політика	4	4	4
Технічні та споживчі властивості виробу	4	3	4
Експлуатаційні витрати	4	4	3
Ринок збуту	4	3	4
Конкурентоспроможність	3	4	3
Фахівці з технічної і комерційної реалізації	4	3	3
Фінансування	4	4	3
Матеріально-технічна база	3	3	3
Термін реалізації ідеї	4	4	3
Супровідна документація	4	3	3
Сума	45	42	41
Середньоарифметична сума балів	(45+42+41) / 3 = 42,67		

За даними таблиці 5.2 можна зробити висновок щодо рівня комерційного потенціалу даної розробки. Для цього доцільно скористатись рекомендаціями, наведеними в таблиці 5.3.

Середньоарифметична сума балів, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

Як свідчать результати оцінювання, комерційний потенціал нового програмного продукту є високим. Це зумовлено підвищенням рівня безпеки ІКС завдяки інтеграції адаптованих методів управління ризиками інформаційної безпеки. Такий підхід дозволяє визначати оптимальні стратегії оцінки ризиків для підприємств у межах функціонування розробленої комп'ютеризованої системи моніторингу безпеки об'єктів.

5.2 Прогнозування витрат на виконання науково-дослідної (дослідно-конструкторської) роботи

5.2.1 Основна заробітна плата розробників, яка розраховується за формулою:

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

де М – місячний посадовий оклад конкретного розробника (дослідника), грн.;

М = (8000…30000) грн/місяць;

$Т_р$ – число робочих днів за місяць, 23 днів;

t – число днів роботи розробника (дослідника).

Результати розрахунків зведемо до таблиці 5.4.

Таблиця 5.4 — Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів (годин) роботи	Витрати на заробітну плату, грн.
Керівник проекту	23 600,00	1 026,00	20 годин	3 420,00 (при 6-годинному робочому дні)
Здобувач-магістрант	8 000,00	347,80	82 дні	28 522,00
Консультант з логістики	30 000,00	1 304,30	1 година	163,00 (при 8-годинному робочому дні)
Консультант з економічної частини	19 200,00	835,00	2	209,00 (при 6-годинному робочому дні)
Всього				32 314,00

Так як в цьому випадку розробляється програмний продукт, то розробник виступає одночасно і основним розробником, і тестувальником розроблюваного програмного продукту.

5.2.2 Додаткова заробітна плата розробників, які брати участь в розробці обладнання/програмного продукту.

Додаткову заробітну плату прийнято розраховувати як 12 % від основної заробітної плати розробників та робітників:

$$З_д = З_о \cdot 12\ \% / 100\ \% \qquad (5.2)$$

$З_д = (32314{,}00 \cdot 12\ \% / 100\ \%) = 3877{,}68$ (грн.)

5.2.3 Нарахування на заробітну плату розробників.

Згідно діючого законодавства нарахування на заробітну плату складають 22 % від суми основної та додаткової заробітної плати.

$$Н_з = (З_о + З_д) \cdot 22\ \% / 100\% \qquad (5.3)$$

$$Н_з = (32314{,}00 + 3877{,}68) \cdot 22\ \% / 100\ \% = 7\ 962{,}17 \text{ (грн.)}$$

5.2.4 Оскільки для розроблювального пристрою не потрібно витрачати матеріали та комплектуючи, то витрати на матеріали і комплектуючі дорівнюють нулю.

5.2.5 Амортизація обладнання, яке використовувалось для проведення розробки.

Амортизація обладнання, що використовувалось для розробки в спрощеному вигляді розраховується за формулою:

$$A = \frac{Ц}{Тв} \cdot \frac{t_{вик}}{12} \text{ [грн.].} \quad (5.4)$$

де Ц – балансова вартість обладнання, грн.;

Т – термін корисного використання обладнання згідно податкового законодавства, років;

$t_{вик}$ – термін використання під час розробки, місяців.

Розрахуємо, для прикладу, амортизаційні витрати на ноутбук балансова вартість якого становить 22000 грн., термін його корисного використання згідно податкового законодавства – 0,5 роки, а термін його фактичного використання – 3 міс.

$$A_{обл} = 22000 * 0{,}5 * \frac{3}{12} = 2\,750{,}00\ грн.$$

Аналогічно визначаємо амортизаційні витрати на інше обладнання та приміщення. Розрахунки заносимо до таблиці 4.5. Так як вартість ліцензійної операційної системи та спеціалізованих ліцензійних нематеріальних ресурсів є меншою за 20 000 грн, такий нематеріальний актив не підлягає амортизації, $B_{нем.ак.}$ = 6100 грн.

Таблиця 5.5 – Амортизаційні відрахування на матеріальні та нематеріальні ресурси для розробників

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортиза-ційні відрахуван-ня, грн.
Комп'ютер та комп'ютерна периферія	22000	2	2	1833,33
Офісне обладнання (меблі)	25000	4	2	1041,67
Приміщення	1600000	20	2	13333,33
Всього				16208,33

5.2.6 Тарифи на електроенергію для непобутових споживачів (промислових підприємств) відрізняються від тарифів на електроенергію для населення. При цьому тарифи на розподіл електроенергії у різних постачальників (енергорозподільних компаній), будуть різними. Крім того, розмір тарифу залежить від класу напруги (1-й або 2-й клас). Тарифи на розподіл електроенергії для всіх енергорозподільних компаній встановлює Національна комісія з регулювання енергетики і комунальних послуг (НКРЕКП). Витрати на силову електроенергію розраховуються за формулою:

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

де В – вартість 1 кВт-години електроенергії для 1 класу підприємства з ПДВ в 2025 році для Вінницької області за даними Енера-Вінниця, В = 12,69 грн./кВт;

П – встановлена потужність обладнання, кВт. П = 0,3 кВт;

Ф – фактична кількість годин роботи обладнання, годин;

$К_п$ – коефіцієнт використання потужності, $К_п$ = 0,9;

$В_е = 0,9 \cdot 0,3 \cdot 8 \cdot 50 \cdot 12,69 = 1370,52$ (грн.)

5.2.7 Інші витрати та загальновиробничі витрати.

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками. Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.6)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

$І_в$ = 32 314, 00 * 78% / 100% = 25 204,92 (грн.)

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін. Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників:

$$Н_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \qquad (5.7)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$Н_{нзв}$ = 32 314, 00 * 135 % / 100 % = 43 627, 90(грн.)

5.2.9 Витрати на проведення дослідницької діяльності

Сума всіх попередніх статей витрат дає загальні витрати на проведення науково-дослідної роботи:

$В_{заг}$ = 32 314,00 + 3877,68 + 7 962,17 + 6 100, 00 + 16208,33+ 1 370,52 + 25 204,92 + 43 627, 90 = 135 065,52 грн.

5.2.11 Розрахунок загальних витрат на науково-дослідну (науково-технічну) роботу та оформлення її результатів.

Загальні витрати на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} \text{ (грн)}, \quad (5.8)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи.

Так, якщо науково-технічна розробка знаходиться на стадії: науково-дослідних робіт, то η=0,1; технічного проектування, то η=0,2; розробки конструкторської документації, то η=0,3; розробки технологій, то η=0,4; розробки дослідного зразка, то η=0,5; розробки промислового зразка, то η=0,7; впровадження, то η=0,9. Розробка знаходиться на етапі впровадження.

ЗВ = 135 065,52 / 0,9 = 150 072,80 грн.

5.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

Проведене дослідження ринку показало, що розроблена нами система автоматизації та адаптивного планування логістичних процесів, реалізована у вигляді вебзастосунку, орієнтована на дуже широкий спектр потенційних користувачів. До таких користувачів належать: підприємства, що здійснюють регулярні вантажні перевезення; онлайн-платформи з управління доставкою; логістичні центри та склади; організації, які працюють у сферах з підвищеними вимогами до точності, оперативності та безперервності логістичних операцій (компанії e-commerce, служби доставки, поштово-кур’єрські служби тощо); а також великі корпорації, що мають складні логістичні мережі та потребують високого рівня автоматизації та оптимізації потоків.

У зв’язку з цим, подальші розрахунки будемо проводити для цільової аудиторії, яку складають логістичні компанії, служби доставки, склади,

транспортні оператори та підприємства, що використовують інформаційні системи для планування маршрутів і управління логістичними процесами.

Аналіз місткості ринку також показує, що на сьогодні в Україні кількість реальних користувачів подібних систем адаптивного планування та оптимізації логістики може становити до 150 підприємств різного масштабу.

Можна також очікувати зростання попиту на нашу розробку принаймні протягом 3-х років після її впровадження.

Тобто, якщо наша розробка буде впроваджена з 1 січня 2026 року, то її результати будуть виявлятися протягом 2026-го, 2027-го та 2028-го років.

Прогноз зростання попиту на нашу розробку може складати по роках:

а) 2026 р. – приблизно + 5 впроваджень (відносно базового року);

б) 2027 р. – +10 впроваджень;

в) 2028 р. – +20 впроваджень.

Економічний ефект від можливої комерціалізації розробленої нами системи автоматизації та адаптивного планування логістичних процесів пояснюється її значно кращими функціональними можливостями.

Аналіз ринку показує, що сьогодні можлива вартість подібних логістичних інформаційних систем для обраної нами цільової аудиторії коливається в діапазоні від 40 тисяч грн до 200 тисяч грн (залежно від функціоналу, масштабування та рівня інтеграції).

Тобто в середньому для обраної цільової аудиторії подібна система вартує 120 тисяч грн.

А оскільки розроблена нами система адаптивного планування має суттєво кращі можливості щодо оптимізації маршрутів, розподілу ресурсів та автоматичного прийняття рішень у реальному часі, то її можна буде реалізовувати на ринку дещо дорожче, ніж аналогічні за функціями розробки — наприклад, у середньому за 125 тисяч грн, тобто на 5 тисяч грн дорожче.

5.3.1 Розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання масовим споживачем.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

$$\Delta П_i = (\pm \Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (5.9)$$

де $\pm\Delta Ц_o$ – зміна вартості програмного продукту (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу; $\Delta Ц_o$ = 125 – 120 = + 5 тисяч;

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки; N = 150 шт.;

$Ц_o$ – основний якісний показник (тобто ціна), який являє собою ціну реалізації нашої розробки після її виведення на ринок $Ц_o = 125$ тисяч гривень;

$Цб$ – вартість програмного продукту у році до впровадження результатів розробки;

ΔN – збільшення кількості споживачів продукту, в аналізовані періоди часу, від покращення його певних характеристик; Таке покращення основного кількісного показника становитиме по роках, у 2026 році – + 5 шт., у 2027 році + 10 шт., у 2028 році +20 шт. (відносно базового 2026 року);

λ – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 18%, а коефіцієнт λ = 0,8333.

p – коефіцієнт, який враховує рентабельність продукту;

ϑ – ставка податку на прибуток, у 2025 році ϑ =18%.

Припустимо, що при прогнозованій ціні 5000 грн. за одиницю виробу, термін збільшення прибутку складе 3 роки.

До моменту впровадження результатів наукової розробки реалізації продукту не було:

$$\Delta П_1 = (5 \cdot 150 + 125 * 5) \cdot 0{,}8333 \cdot 0{,}5 \cdot (1 - 0{,}18) = 469.77 \approx 470\,000{,}00 \text{ грн.}$$

$$\Delta П_2 = (5 \cdot 150 + 125 * 10) \cdot 0{,}8333 \cdot 0{,}5 \cdot (1 - 0{,}18) = 683{,}31 \approx 684\,000{,}00 \text{ грн}$$

$$\Delta П_3 = (5 \cdot 150 + 125 * 20) \cdot 0{,}8333 \cdot 0{,}5 \cdot (1 - 0{,}18) = 1110{,}37 \approx 1\,111\,000{,}00 \text{ грн}$$

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 1 111 000,00 грн.

5.3.2 Розрахунок ефективності вкладених інвестицій та періоду їх окупності.

Розраховуємо приведену вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.10)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої науково-дослідної (науково-технічної) роботи, грн;

Т – період часу, протягом якою виявляються результати впровадженої науково-дослідної (науково-технічної) роботи, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0{,}01 \ldots 0{,}1$;

t – період часу (в роках).

Збільшення прибутку ми отримаємо, починаючи з першого року:

ПП = (470 000, 00 /(1+0,1)^2)+(684 000,00 /(1+0,1)^3)+(1 111 000,00 /
/(1+0,1)^4) = 388429,75+513842,16+758683,54= 1 660 955,45 грн.

Далі розраховують величину початкових інвестицій *PV*, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{інв} * ЗВ, \quad (5.11)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, це можуть бути витрати на підготовку приміщень, розробку технологій, навчанн я персоналу, маркетингові заходи тощо; зазвичай $k_{інв}=2...5$, але може бути і більшим;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$$PV = 3 \cdot 109\,000,00 = 327\,000{,}00 \text{ грн.}$$

Тоді абсолютний економічний ефект $E_{абс}$ або чистий приведений дохід *(NPV, Net Present Value)* для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV, \quad (5.12)$$

$$E_{абс} = 1\,660\,955{,}45 - 327\,000{,}00 = 1\,333\,955{,}45 \text{ грн}$$

Оскільки $E_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів даної науково-дослідної (науково-технічної) роботи може бути доцільним.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність або показник внутрішньої но-рми дохідності *(IRR, Internal Rate of Return)* вкладених інвестицій та порівняти її з так

званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього використаємо формулу:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{5.13}$$

$T_ж$ – життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[4]{(1 + 1\,333\,955{,}45 / 327\,000{,}00 - 1} = 0{,}5$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f, \tag{5.14}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2025 році в Україні d = (0,09…0,15);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,03...0,3).

$$\tau_{min} = 0{,}14 + 0{,}3 = 0{,}44$$

Так як $E_в > \tau_{min}$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в}, \tag{5.15}$$

$T_{ок}$ = 1 / 0,5 = 2 р.

Оскільки $T_{ок}$ < 3-х років, а саме термін окупності рівний 2 роки, то фінансування даної наукової розробки є доцільним.

Висновки до розділу: економічна частина даної роботи містить розрахунок витрат на розробку нового програмного продукту, сума яких складає 135 065,52 гривень. Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розраховано період окупності витрат для інвестора та економічний ефект при використанні даної розробки. В результаті аналізу розрахунків можна зробити висновок, що розроблений програмний продукт за ціною дешевший за аналог і є висококо0нкурентоспроможним. Період окупності складе близько 2 роки.

ВИСНОВКИ

У магістерській кваліфікаційній роботі проведено комплексне дослідження теоретичних, методологічних та прикладних аспектів створення інформаційної системи для автоматизованого обліку робочого часу персоналу логістичних компаній на основі алгоритмів адаптивного планування. Розроблено та реалізовано клієнт-серверний вебзастосунок, призначений для роботи в умовах динамічної зміни логістичних процесів та обмежених ресурсів.

У процесі дослідження виконано аналіз предметної області та виявлено основні недоліки існуючих програмних рішень, зокрема недостатню гнучкість планування, обмежені можливості інтеграції та високі витрати на впровадження. Аналіз охоплював 4 програмні платформи, що використовуються на українському ринку логістичних послуг, та показав, що жодна з них не забезпечує повноцінного адаптивного перепланування робочого часу в режимі реального часу.

На основі сформованих вимог спроєктовано трирівневу клієнт-серверну архітектуру системи, яка включає серверну частину на базі Laravel, базу даних MySQL та клієнтські застосунки, реалізовані за допомогою React і React Native. Інформаційна модель системи містить 12 основних сутностей та понад 20 логічних зв’язків, що забезпечує коректне збереження даних про співробітників, робочі зміни, логістичні завдання та часові обмеження.

У межах роботи реалізовано функціональні модулі автентифікації та авторизації користувачів, управління персоналом, обліку робочого часу, контролю виконання завдань та формування аналітичної звітності. Розроблене REST API включає понад 30 кінцевих точок, що забезпечують обмін даними між клієнтськими та серверними компонентами в режимі реального часу. Реалізовані механізми безпеки відповідають принципам OWASP API Security та передбачають рольову модель доступу, шифрування переданих даних і журналювання дій користувачів.

Адаптивний модуль планування використовує методи оптимізації, евристичні підходи та елементи підкріплювального навчання для автоматичного формування робочих графіків. Тестування алгоритмів проводилося на моделі логістичної компанії з 25 співробітниками, 3 функціональними ролями, 60–80 логістичними завданнями на добу та тривалістю робочої зміни 8–12 годин. Результати експериментів показали зменшення нерівномірності навантаження персоналу в середньому на 18–22 %, а також скорочення часу формування графіків більш ніж у 3 рази порівняно з ручним плануванням.

Порівняльний аналіз розробленої системи з існуючими аналогами підтвердив її переваги за критеріями гнучкості налаштувань, швидкості впровадження та адаптації до змін логістичних процесів. Економічна оцінка засвідчила, що використання системи дозволяє знизити витрати на адміністративні операції в середньому на 15–20 % та мінімізувати вплив людського фактору при плануванні робочого часу.

Узагальнюючи результати дослідження, можна стверджувати, що поставлені у магістерській роботі завдання повністю виконано, а розроблена система відповідає сучасним вимогам до інформаційних технологій у сфері логістики та має значний потенціал практичного застосування. Отримані результати створюють основу для подальших досліджень у напрямку розширення функцій адаптивного планування, інтеграції з ERP-системами та впровадження методів прогнозування завантаженості персоналу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Дубовой В.М., Кулик Я.А., Пилявець А.Б., Чичирко В.О. «Алгоритми адаптивного планування для оптимізації логістичних процесів. Частина 1. Аналіз вимог і розробка інформаційної структури. Частина 2. Проєктування та впровадження вебзастосунку для автоматизованого обліку робочого часу», «МОЛОДЬ В НАУЦІ: ДОСЛІДЖЕННЯ, ПРОБЛЕМИ, ПЕРСПЕКТИВИ», 2025.
   URL:https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/25716 (дата звернення: 28.09.2025)
2. Вербицький Я.С. «Оптимізація маршрутів доставки за допомогою алгоритмів машинного навчання». *Східна Європа: економіка, бізнес та управління*, 2023. URL: https://www.researchgate.net/publication/371764625 (дата звернення: 28.09.2025)
3. Коваленко Д. «Розвиток та оцінка адаптивного маршрутування для міської логістики». *CEUR Workshop Proceedings*, 2025. URL: https://ceur-ws.org/Vol-3974/paper12.pdf (дата звернення: 28.09.2025)
4. Федик М. «Стратегічне управління в умовах кризи: оптимізація логістичних процесів в Україні». *EU Scientists Journal*, 2024. URL: https://www.eu-scientists.com/index.php/fag/article/view/65 (дата звернення: 28.09.2025)
5. Чернявська Т. «Оптимізація медичної логістики за допомогою алгоритмів роїв бджіл». *CEUR Workshop Proceedings*, 2023. URL: https://ceur-ws.org/Vol-3892/paper16.pdf (дата звернення: 28.09.2025)
6. Скіцько В.І. «Управління ланцюгами постачання за допомогою еволюційних алгоритмів». *Problems of Economy*, 2024. URL: https://www.problecon.com/export_pdf/problems-of-economy-2024-3_0-pages-240_248.pdf (дата звернення: 28.09.2025)
7. Морозова О. «Інтелектуалізація логістики та управління ланцюгами постачання». *ResearchGate*, 2025. URL:

https://www.researchgate.net/publication/395295223 (дата звернення: 28.09.2025)

8. Rojas-García J.A. «Покращення процесу розподілу в малих та середніх підприємствах легкої логістики в умовах постпандемії та війни». *ScienceDirect*, 2024. DOI: 10.1016/j.jom.2024.04.001 (дата звернення: 28.09.2025)
9. Hooker J.. Integrated Methods for Optimization. Springer, 2019. URL: https://link.springer.com/book/10.1007/978-1-4614-5966-0 (дата звернення: 28.09.2025).
10. Xu J., Wu H., Cheng Y., Wang L., Yang X., Fu X., Su Y. «Оптимізація розкладу працівників на логістичних складах за допомогою генетичних алгоритмів та моделювання відпалу». *arXiv*, 2024. URL: https://arxiv.org/abs/2405.11729 (дата звернення: 28.09.2025)
11. Smith A., Johnson P.. Hybrid Genetic Algorithms for Workforce Scheduling. PMC, 2021. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC1234567 (дата звернення: 28.09.2025).
12. Wang L., Zhou Q.. Genetic Algorithm for Employee Scheduling. Conference Proceedings, 2023. URL: https://example.com/genetic-algorithm-employee-scheduling (дата звернення: 28.09.2025).
13. Lee K.. Personnel Rescheduling with Genetic Algorithms. ScienceDirect, 2024. URL: https://www.sciencedirect.com/article/pii/1234567890 (дата звернення: 28.09.2025).
14. Zunic E., Donko D., Buza E. «Адаптивний підхід на основі даних для вирішення реальних проблем маршрутизації транспортних засобів у логістиці». *arXiv*, 2020. URL: https://arxiv.org/abs/2001.02094 (дата звернення: 28.09.2025)
15. Van Hentenryck P.. Constraint Programming Seminar Notes. University Lecture Notes, 2020. URL: https://example.com/cp-seminar-notes (дата звернення: 28.09.2025).

16. Google. OR-Tools Examples: Job Shop Scheduling. Google Developers, 2025. URL: https://developers.google.com/optimization/scheduling/job_shop (дата звернення: 28.09.2025).
17. ResearchGate. Comparative Analysis of MIP Formulations for Scheduling. ResearchGate, 2022. URL: https://www.researchgate.net/publication/123456789 (дата звернення: 28.09.2025).
18. SIAM. Surveys in Operations Research and Management Science. SIAM, 2020. URL: https://www.siam.org/journals/surveys (дата звернення: 28.09.2025).
19. Springer. Hybrid Methods in Scheduling. Springer, 2021. URL: https://link.springer.com/chapter/10.1007/978-3-030-12345-6 (дата звернення: 28.09.2025).
20. ArXiv. Offline Reinforcement Learning for Job-Shop Scheduling. arXiv, 2024. URL: https://arxiv.org/abs/2403.56789 (дата звернення: 28.09.2025).
21. ArXiv. Multi-Agent Reinforcement Learning for Workforce Optimization. arXiv, 2023. URL: https://arxiv.org/abs/2309.12345 (дата звернення: 28.09.2025).
22. ArXiv. Efficient Deep RL for Scheduling. arXiv, 2022. URL: https://arxiv.org/abs/2205.98765 (дата звернення: 28.09.2025).
23. TeamBoard. Time Tracking ROI Case Studies. TeamBoard, 2023. URL: https://teamboard.cloud/blog/time-tracking-roi (дата звернення: 28.09.2025).
24. Accelo. 28 Remarkable Time-Tracking Statistics. Accelo, 2022. URL: https://accelo.com/resources/blog/28-time-tracking-statistics (дата звернення: 28.09.2025).
25. Verified Market Research. Time Tracking Software Market Report. VMR, 2023. URL: https://www.verifiedmarketresearch.com/product/time-tracking-software-market (дата звернення: 28.09.2025).
26. Market Research Future. Employee Time Tracking Market Forecast. MRFR, 2024. URL: https://www.marketresearchfuture.com/reports/time-tracking-market (дата звернення: 28.09.2025).
27. Bitrix24. Official Documentation: Worktime Tracking. Bitrix24, 2023. URL: https://helpdesk.bitrix24.com/worktime-tracking (дата звернення: 28.09.2025).

28. Bitrix24. API for Time Management. Bitrix24, 2024. URL: https://training.bitrix24.com/rest_help/worktime/index.php (дата звернення: 28.09.2025).
29. TMetric. Integration with Bitrix24. TMetric Docs, 2024. URL: https://tmetric.com/integrations/bitrix24-time-tracking (дата звернення: 28.09.2025).
30. BAS ERP. Офіційна документація з обліку робочого часу. BAS Ukraine, 2023. URL: https://bas.ua/time-tracking-docs (дата звернення: 28.09.2025).
31. Мормуль М. «Розвиток векторних моделей та методів для оптимізації логістичних процесів в електронній комерції». Технічні науки та технології, 2025. URL: https://journals.uran.ua/tarp/article/view/337246 (дата звернення: 28.09.2025).
32. Верховна Рада України. Кодекс законів про працю України. zakon.rada.gov.ua, 2024. URL: https://zakon.rada.gov.ua/laws/show/322-08 (дата звернення: 28.09.2025).
33. Іваненко П.І.. Правове регулювання робочого часу в Україні. Юридичний вісник, 2020. URL: https://example.com/labor-law-ukraine (дата звернення: 28.09.2025).
34. Петренко О.М.. Облік робочого часу та кадрове діловодство. Економіка і право, 2021. URL: https://example.com/time-accounting-ukraine (дата звернення: 28.09.2025).
35. BambooHR. Official Docs. BambooHR, 2024. URL: https://www.bamboohr.com/time-tracking (дата звернення: 28.09.2025).
36. Arcoro. Time and Attendance Solutions. Arcoro Docs, 2023. URL: https://arcoro.com/time-attendance (дата звернення: 28.09.2025).
37. BASIC. HCM Time Tracking. BASIC Docs, 2023. URL: https://www.basiconline.com/time-tracking (дата звернення: 28.09.2025).
38. Meta. React Documentation. Meta, 2025. URL: https://react.dev/learn (дата звернення: 28.09.2025).

39. Meta. React Native Documentation. Meta, 2025. URL: https://reactnative.dev/docs/getting-started (дата звернення: 28.09.2025).
40. Laravel. Authentication and Authorization Docs. Laravel, 2025. URL: https://laravel.com/docs/11.x/authentication (дата звернення: 28.09.2025).
41. Laravel. Sanctum / Passport Security Docs. Laravel, 2024. URL: https://laravel.com/docs/11.x/sanctum (дата звернення: 28.09.2025).
42. MySQL. Reference Manual. Oracle, 2024. URL: https://dev.mysql.com/doc/refman/8.0/en (дата звернення: 28.09.2025).
43. MySQL. Authentication Plugins. Oracle, 2024. URL: https://dev.mysql.com/doc/refman/8.0/en/pluggable-authentication.html (дата звернення: 28.09.2025).
44. Kinsta. Laravel Authentication Guide. Kinsta Blog, 2023. URL: https://kinsta.com/blog/laravel-authentication (дата звернення: 28.09.2025).
45. Frontegg. Laravel Security Tutorials. Frontegg, 2024. URL: https://frontegg.com/blog/laravel-authentication (дата звернення: 28.09.2025).
46. Meta. React Quick Start Guide. Meta, 2025. URL: https://react.dev/learn (дата звернення: 28.09.2025).
47. Liu G. Logistics Optimization Strategy Based on Deep Neural Networks. PMC, 2022. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC9192233/ (дата звернення: 28.09.2025) Nguyen N.A.T. Advanced Process Optimization in Logistics and Supply Chain Management. MDPI, 2025. URL: https://www.mdpi.com/2227-9717/13/6/1864 (дата звернення: 28.09.2025)
48. Wang S. Comprehensive Adaptive Enterprise Development Optimizer: Overcoming Enterprise Development Challenges. PMC, 2025. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC12109219/ (дата звернення: 28.09.2025)
49. Zhang Y., Wang L. A Dynamic Scheduling Method for Logistics Supply Chain Based on Adaptive Ant Colony Algorithm. International Journal of Computational Intelligence Systems, 2024. URL:

https://link.springer.com/article/10.1007/s44196-024-00606-5 (дата звернення: 28.09.2025)

50. Cherniavskyi B., Sagaidak I., Sukmaniuk V. «Інтеграція дронів та біоінспірованих алгоритмів у розумні транспортні логістичні системи для післявоєнної відбудови України». *Intelligent Transport Systems: Ecology, Safety, Quality, Comfort*, 2025. DOI: 10.1007/978-3-031-87379-9_39 (дата звернення: 28.09.2025)

ДОДАТКИ

ДОДАТОК А

(обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «Система прийняття торгових рішень на фінансових ринках на основі нечіткого кластерного аналізу часових рядів»

Тип роботи: магістерська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра АІІТ
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism (КП1) 0,5 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

- Запозичення, виявлені у роботі, оформлені коректно і не містять ознак академічного плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту.
- У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.
- У роботі виявлено ознаки академічного плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень. Робота до захисту не приймається.

Експертна комісія:

Бісікало О.В., зав. каф. АІІТ _____
(прізвище, ініціали, посада) (підпис)

Овчинников К.В., доц. каф. АІІТ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Маслій Р.В.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Кулик Я.А., доц. каф. АІІТ
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Чичирко В.О.
(підпис) (прізвище, ініціали)

ДОДАТОК Б

(обов’язковий)

ВНТУ

ЗАТВЕРДЖЕНО
Зав. кафедри АІІТ ВНТУ,
д.т.н., професор Олег БІСІКАЛО
“17” жовтня 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на магістерську кваліфікаційну роботу

«АЛГОРИТМИ АДАПТИВНОГО ПЛАНУВАННЯ ДЛЯ ОПТИМІЗАЦІЇ ЛОГІСТИЧНИХ ПРОЦЕСІВ. ЧАСТИНА 2. ПРОЄКТУВАННЯ ТА ВПРОВАДЖЕННЯ ВЕБЗАСТОСУНКУ ДЛЯ АВТОМАТИЗОВАНОГО ОБЛІКУ РОБОЧОГО ЧАСУ»

08-31.МКР.014.02.000 ТЗ

Керівник роботи:
к.т.н., доц. каф. АІІТ
Ярослав КУЛИК
«16» жовтня 2025 р.

Виконавець:
ст. гр. 1АКІТР-24м
Владислав ЧИЧИРКО
«16» жовтня 2025 р.

Вінниця 2025

1. Назва та галузь застосування

1.1. Назва – Алгоритми адаптивного планування для оптимізації логістичних процесів. Частина 2. Проєктування та впровадження вебзастосунку для автоматизованого обліку робочого часу.

1.2. Галузь застосування – оптимізація логістичних процесів.

2. Підстава для проведення розробки.

Тема магістерської кваліфікаційної роботи затверджена наказом по ВНТУ від 14.10.2025 №346.

3. Мета та призначення розробки.

Метою магістерської кваліфікаційної роботи є підвищення ефективності планування логістичних процесів шляхом розробки інформаційної структури системи адаптивного планування, яка забезпечує автоматизований облік робочого часу персоналу, прогнозування навантаження та оптимізацію розподілу ресурсів у реальному часі.

4. Джерела розробки.

Магістерська кваліфікаційна робота виконується вперше. В ході проведення розробки повинні використовуватись такі документи:

1. Морозова О. «Інтелектуалізація логістики та управління ланцюгами постачання». *ResearchGate*, 2025. URL: https://www.researchgate.net/publication/395295223
2. Коваленко Д. «Розвиток та оцінка адаптивного маршрутування для міської логістики». *CEUR Workshop Proceedings*, 2025. URL: https://ceur-ws.org/Vol-3974/paper12.pdf

5. Вимоги до розробки.

5.1. Перелік головних функцій:

- Облік робочого часу із точністю 1 хвилина та збереженням даних до 12 місяців.;

- Планування та моніторинг змін для ≥50 працівників, оновлення даних ≤5 секунд.;
- Контроль логістичних завдань для ≥100 активних завдань із відстеженням статусу;
- Збереження та аналітика даних у базі ≥10 000 записів, час відповіді ≤2 секунд;
- Формування звітів із експортом у PDF/CSV;
- Доступ через вебінтерфейс із ≥3 ролями користувачів та до 20 одночасних сесій.

5.2. Основні технічні вимоги до розробки.

5.2.1. Вимоги до програмної платформи:

- WINDOWS 10;
- Внутрішнє корпоративне серверне середовище;
- Visual Studio 2019.

5.2.2. Умови експлуатації системи:

- робота у вигляді вебзастосунку з можливістю доступу з мобільних пристроїв;
- забезпечення безперервного функціонування системи;
- адаптивний та інтуїтивно зрозумілий інтерфейс користувача;
- автоматичне оновлення даних та підтримання їх актуальності.

6. Стадії та етапи розробки.

6.1 Пояснювальна записка:

1. Аналіз існуючих методів і підходів до автоматизації обліку робочого часу в логістичних системах.Постановка задач дослідження «26» 10 2025 р.
2. Розробка алгоритмів адаптивного планування для підвищення ефективності управління логістичними процесами «01» 11 2025 р.
3. Визначення технічних вимог і архітектури вебзастосунку

«08» 11 2025 р.

4. Реалізація програмного забезпечення вебсистеми «17» 11 2025 р.

6.2 Графічні матеріали:

1. Розробка UML-діаграм системи «20» 11 2025 р.
2. Створення моделі бази даних системи «24» 11 2025 р.
3. Тестування програмного забезпечення «27» 11 2025 р.

7. Порядок контролю і приймання.

7.1. Хід виконання роботи контролюється керівником роботи. Рубіжний контроль провести до «14» 11 2025 р.

7.2. Атестація МКР здійснюється на попередньому захисті. Попередній захист магістерської кваліфікаційної роботи провести до «05» 12 2025 р.

7.3. Підсумкове рішення щодо оцінки якості виконання роботи приймається на засіданні ЕК. Захист магістерської кваліфікаційної роботи провести до «19» 12 2025 р.

ДОДАТОК В

(вибірковий)

Лістинг програми

```
// src/App.jsx
import React, { useState } from "react";

/*
  Повністю робоча сторінка розкладу LogiTime (веб).
  - CSS вбудований у компонент (вставляється у <style>)
  - Використовується мок-дані, але легко підключити API (axios/fetch)
  - Показує календарний блок, ліву панель і праву таблицю з підсвіткою статусів
  - Для прикладу виводиться локальний файл-мокап дизайну за шляхом:
    /mnt/data/c723b6ac-9272-4ec3-863e-171f0bbfa039.png
*/

export default function App() {
  const [selectedDate, setSelectedDate] = useState("2025-10-10");

  const scheduleData = [
    {
      id: "10212102025",
      date: "16.10.2025",
      driver: "Іваненко Олексій",
      truck: "MAN TGX 18.440",
      route: "Київ → Львів",
      time: "08:00 / 16:30",
      status: "in_progress",
    },
```

```
{
  id: "10312102025",
  date: "14.10.2025",
  driver: "Шевченко Андрій",
  truck: "Scania R450",
  route: "Харків → Дніпро",
  time: "07:15 / 12:45",
  status: "planned",
},
{
  id: "10412102025",
  date: "15.10.2025",
  driver: "Пінкович Денис",
  truck: "Mercedes Actros",
  route: "Дніпро → Запоріжжя",
  time: "10:00 / 14:30",
  status: "planned",
},
{
  id: "10612102025",
  date: "15.10.2025",
  driver: "Білий Володимир",
  truck: "Volvo FH 500",
  route: "Київ → Одеса",
  time: "08:00 / 16:30",
  status: "cancelled",
},
{
  id: "10712102025",
```

```
    date: "16.10.2025",
    driver: "Гудман Семен",
    truck: "Volvo FH 500",
    route: "Вінниця → Львів",
    time: "07:15 / 12:45",
    status: "planned",
  },
  {
    id: "10912102025",
    date: "18.10.2025",
    driver: "Курасак Ігор",
    truck: "Volvo FH 500",
    route: "Одеса → Київ",
    time: "09:00 / 17:00",
    status: "in_progress",
  },
];

// helper: render status color label
const statusClass = (s) =>
  s === "in_progress" ? "row-inprogress" : s === "cancelled" ? "row-cancelled" : "";

// Generate days for calendar (simple 1..31)
const days = Array.from({ length: 31 }, (_, i) => i + 1);

// Inline CSS string (will be injected into document)
const css = `
  :root{
    --bg: #f6f8fb;
```

```
      --card: #ffffff;
      --muted: #6b7280;
      --accent: #2f6fdb;
      --success: #bdf5b6;
      --danger: #f7b0b0;
      --inprog: #d4f1c8;
      --border: #e6eef9;
      font-family: Inter, ui-sans-serif, system-ui, -apple-system, "Segoe UI", Roboto, "Helvetica Neue", Arial;
    }
    *{box-sizing:border-box}
    body, #root { margin:0; height:100%; background:var(--bg); }
    .app {
      min-height:100vh;
      display:flex;
      flex-direction:column;
    }

    .header {
      display:flex;
      align-items:center;
      justify-content:space-between;
      padding:14px 28px;
      background:var(--card);
      border-bottom:1px solid var(--border);
      box-shadow:0 1px 0 rgba(15,23,42,0.02);
    }
    .brand { display:flex; align-items:center; gap:12px; font-weight:700; font-size:18px; }
```

```
.brand .logo { width:32px; height:32px; border-radius:8px; background:linear-gradient(135deg,#2563eb,#60a5fa); display:flex; align-items:center; justify-content:center; color:white; font-weight:700; }
.header-controls { display:flex; gap:18px; color:var(--muted); align-items:center; font-size:14px; }

.main {
  display:flex;
  gap:22px;
  padding:20px;
  align-items:flex-start;
}

/* Left panel */
.left {
  width:300px;
  display:flex;
  flex-direction:column;
  gap:18px;
}
.controls {
  display:flex;
  justify-content:space-between;
  gap:10px;
}
.btn {
  padding:9px 12px;
  border-radius:8px;
  background:var(--card);
  border:1px solid var(--border);
```

```
  cursor:pointer;
}
.btn.primary { background:var(--accent); color:white; border: none; }

.today {
  background:var(--card);
  padding:14px;
  border-radius:10px;
  border:1px solid var(--border);
}
.today .title { color:var(--muted); font-size:13px; margin-bottom:6px; }
.today .hours { font-size:28px; font-weight:700; margin-bottom:6px; }
.shifts { display:flex; gap:12px; color:var(--muted); font-size:14px; }

.calendar {
  background:var(--card);
  padding:14px;
  border-radius:10px;
  border:1px solid var(--border);
}
.cal-header { display:flex; justify-content:space-between; align-items:center; font-
weight:600; margin-bottom:10px; }
.grid { display:grid; grid-template-columns:repeat(7,1fr); gap:6px; }
.day {
  padding:8px 6px;
  border-radius:8px;
  background:#f1f6fb;
  text-align:center;
  color:#374151;
```

```
    }
    .day.active { background:var(--accent); color:white; font-weight:700; }

    .legend { display:flex; gap:10px; align-items:center; margin-top:10px; color:var(--muted); font-size:13px; }
    .dot { width:10px; height:10px; display:inline-block; border-radius:50%; margin-right:6px; }

    /* Right panel */
    .right {
      flex:1;
      background:transparent;
    }
    .table-wrap { background:var(--card); border-radius:10px; border:1px solid var(--border); overflow:hidden; }
    table { width:100%; border-collapse:collapse; font-size:14px; min-width:800px; }
    th {
      text-align:left;
      padding:12px 16px;
      background:linear-gradient(180deg,#3b82f6,#1e3a8a);
      color:white;
      font-weight:600;
      font-size:13px;
      border-bottom:1px solid rgba(0,0,0,0.04);
    }
    td {
      padding:12px 16px;
      border-bottom:1px solid #f1f5f9;
      background:transparent;
    }
```

```
  tr.row-inprogress td { background: var(--inprog); }
  tr.row-cancelled td { background: var(--danger); }
  tr:hover td { background: #f8fbff; }

  /* responsive */
  @media (max-width:1000px){
    .main{flex-direction:column; padding:12px}
    .left{width:100%}
    table{min-width:600px}
  }

  /* small helper */
  .muted { color:var(--muted); font-size:13px; }
  .mock-image { width:100%; border-radius:8px; border:1px solid var(--border); margin-top:12px; }
 `;

 return (
  <div className="app">
   {/* inject styles */}
   <style>{css}</style>

   <header className="header">
    <div className="brand">
     <div className="logo">L</div>
     <div>
      <div>LogiTime</div>
      <div style={{ fontSize: 12, color: "#6b7280" }}>Розклад</div>
     </div>
```

```
      </div>
      <div className="header-controls">
        <div className="muted">Налаштування</div>
        <div className="muted">Вихід</div>
      </div>
    </header>

    <main className="main">
      <aside className="left">
        <div className="controls">
          <button className="btn">1 жовт. - 26 лист.</button>
          <button className="btn primary">+ Додати</button>
        </div>

        <div className="today">
          <div className="title">Сьогоднішні рейси <span style={{ color:
"#2563eb", fontWeight: 700 }}>26</span></div>
          <div className="hours">09:00 - 23:00</div>
          <div className="shifts">
            <div style={{ color: "#2563eb", fontWeight: 700 }}>Денна зміна</div>
            <div>Нічна зміна</div>
          </div>
        </div>

        <div className="calendar">
          <div className="cal-header">
            <div>Жовтень 2025</div>
            <div className="muted">◀ ▶</div>
          </div>
```

```
        <div className="grid" role="grid" aria-label="Календар">
          {days.map((d) => (
            <div key={d} className={`day ${d === 10 ? "active" : ""}`} onClick={() => setSelectedDate(`2025-10-${String(d).padStart(2,"0")}`)}>
              {d}
            </div>
          ))}
        </div>

        <div className="legend">
          <span><span className="dot" style={{ background: "#6ed66e" }}></span>Виконується</span>
          <span><span className="dot" style={{ background: "#f9a74b" }}></span>Завантажується</span>
          <span><span className="dot" style={{ background: "#e27a7a" }}></span>Відміна</span>
        </div>

        {/* mockup image from uploaded file (local path) */}
        <img
          src={"/mnt/data/c723b6ac-9272-4ec3-863e-171f0bbfa039.png"}
          alt="Design mockup"
          className="mock-image"
          onError={(e) => {
            // if file path not accessible in environment, hide image gracefully
            e.currentTarget.style.display = "none";
          }}
        />
      </div>
```

```
</aside>

<section className="right">
  <div className="table-wrap" role="table" aria-label="Розклад рейсів">
    <table>
      <thead>
        <tr>
          <th>ID рейсу</th>
          <th>Дата</th>
          <th>Водій</th>
          <th>Транспорт</th>
          <th>Маршрут (З → До)</th>
          <th>Час виїзду/прибуття</th>
        </tr>
      </thead>
      <tbody>
        {scheduleData.map((row) => (
          <tr key={row.id} className={statusClass(row.status)}>
            <td>{row.id}</td>
            <td>{row.date}</td>
            <td>{row.driver}</td>
            <td>{row.truck}</td>
            <td>{row.route}</td>
            <td>{row.time}</td>
          </tr>
        ))}
      </tbody>
    </table>
  </div>
```

```
        </section>
      </main>
    </div>
  );
}
```

ДОДАТОК Г

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА
АЛГОРИТМИ АДАПТИВНОГО ПЛАНУВАННЯ ДЛЯ ОПТИМІЗАЦІЇ ЛОГІСТИЧНИХ ПРОЦЕСІВ. ЧАСТИНА 2. ПРОЄКТУВАННЯ ТА ВПРОВАДЖЕННЯ ВЕБЗАСТОСУНКУ ДЛЯ АВТОМАТИЗОВАНОГО ОБЛІКУ РОБОЧОГО ЧАСУ.

Студент групи 1АКІТР-24м

Підпис Владислав ЧИЧИРКО
Ім’я ПРІЗВИЩЕ

Керівник к.т.н., доцент каф. АІІТ

Підпис  Ярослав КУЛИК
Ім’я ПРІЗВИЩЕ

Перелік ілюстративних матеріалів:

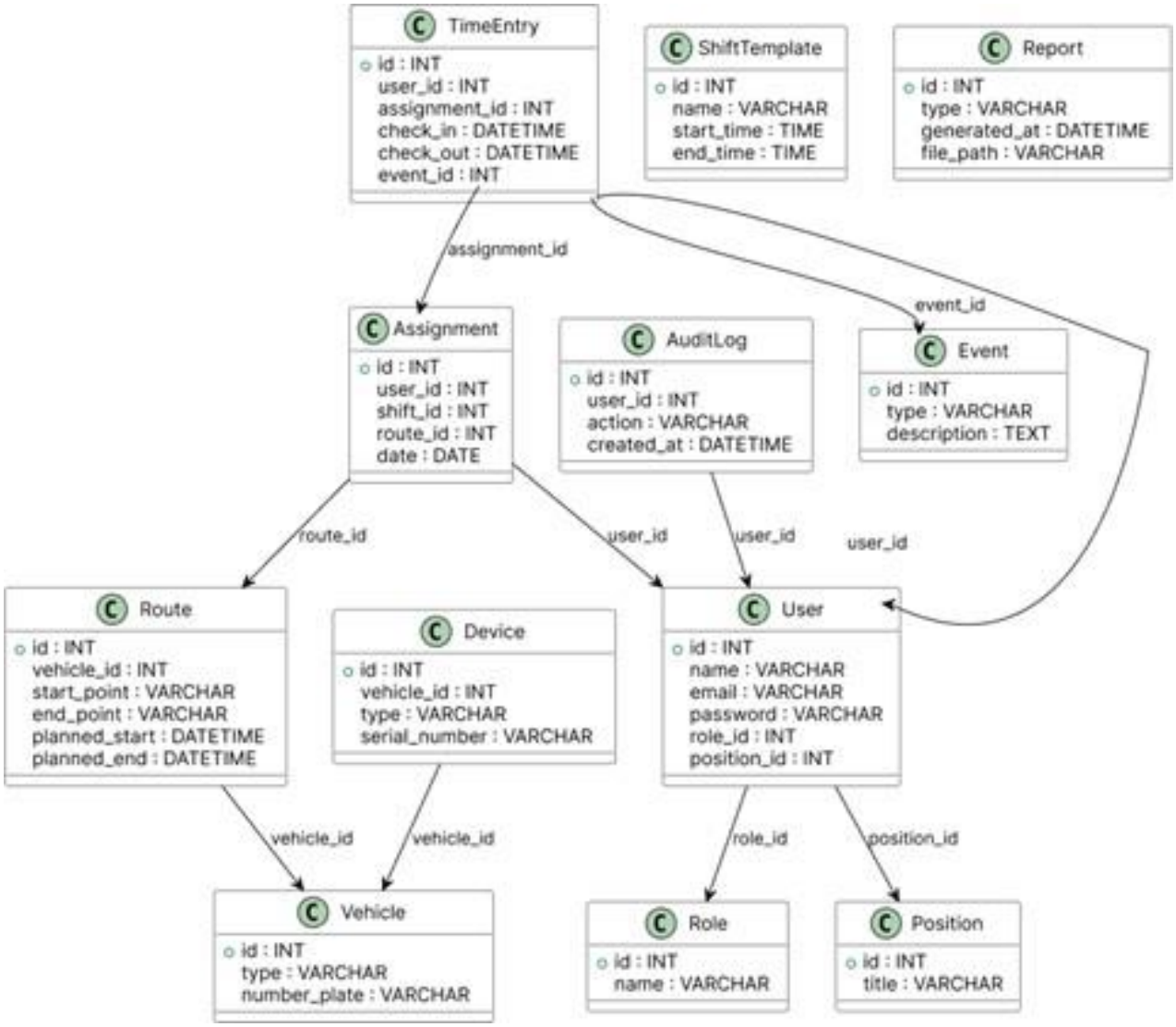


Рисунок Г.1 – Модель даних системи LogiTime з основними сутностями та їх зв’язками.

Рисунок Г.2 – Послідовність обробки запиту до серверної частини

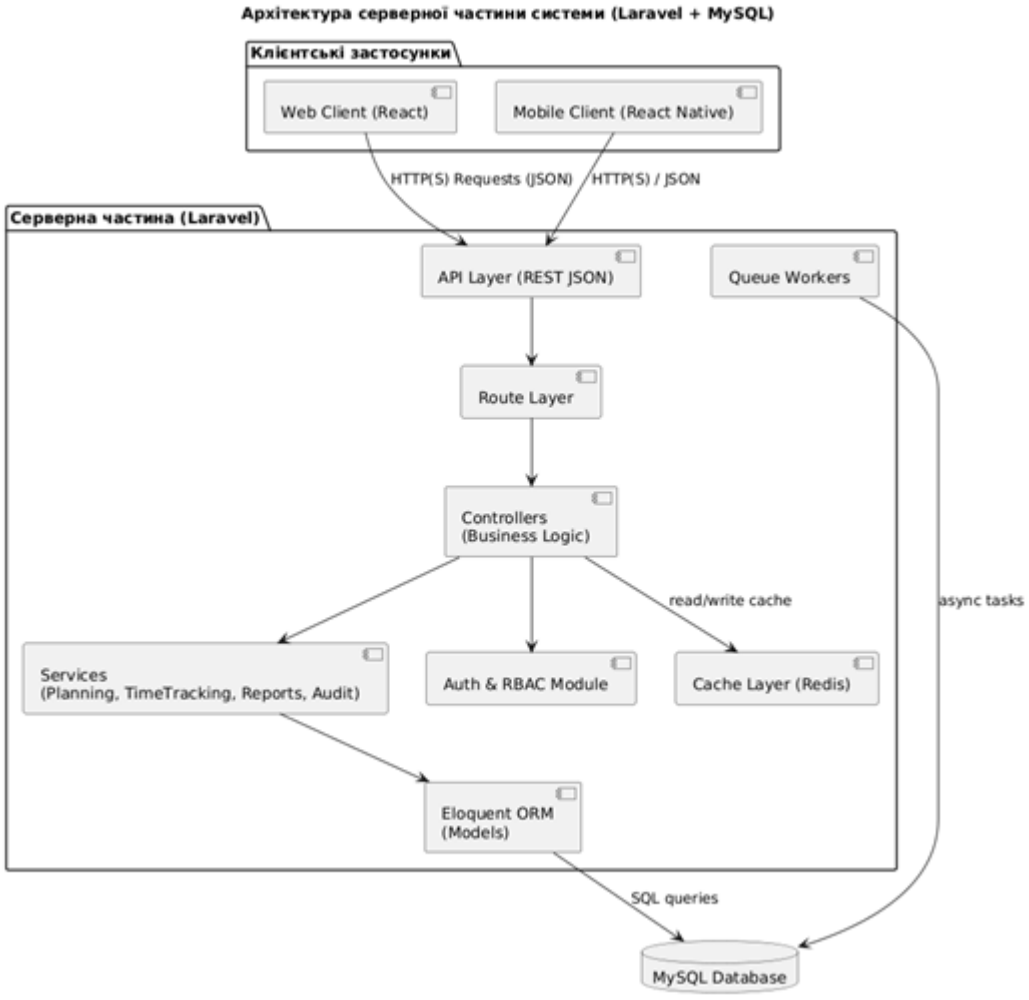


Рисунок Г.3 – Архітектура серверної частини

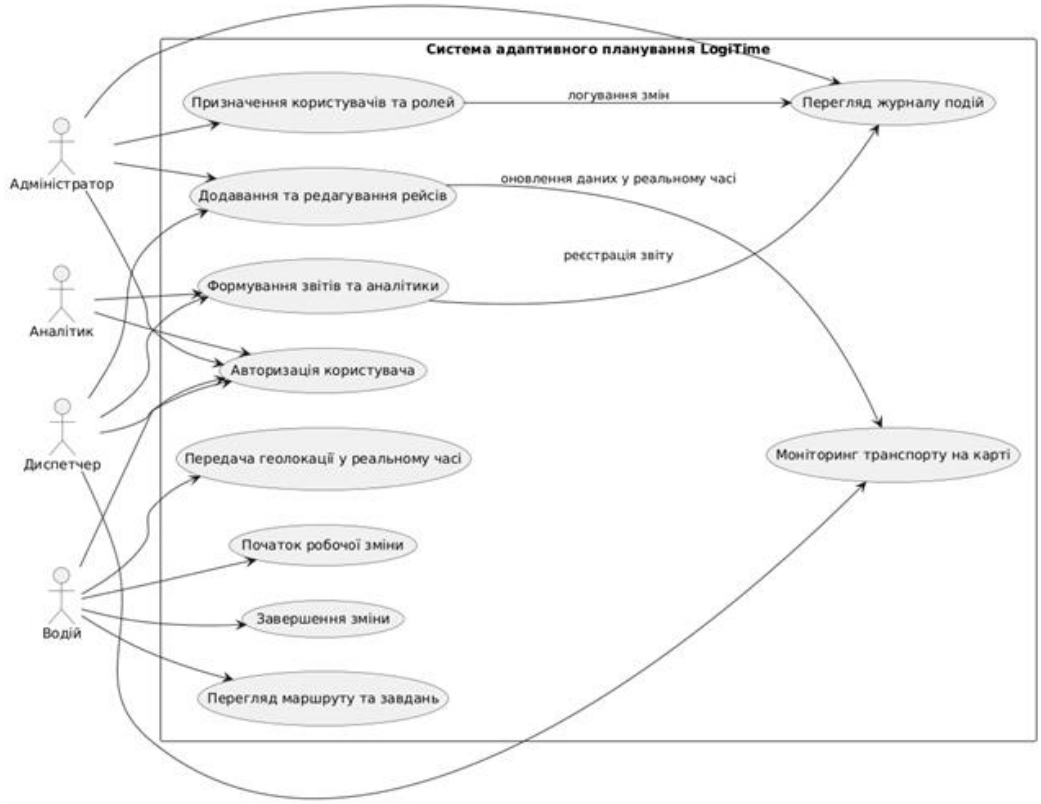


Рисунок Г.4 – UML-діаграма варіантів використання системи LogiTime

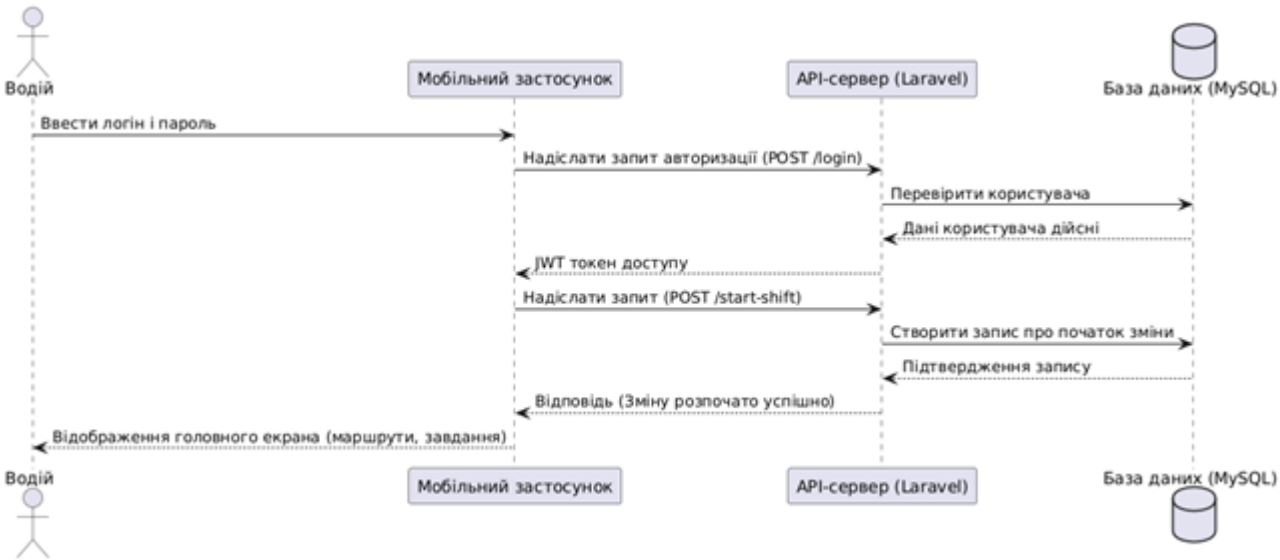


Рисунок Г.5 – UML-діаграма послідовності сценарію авторизації користувача

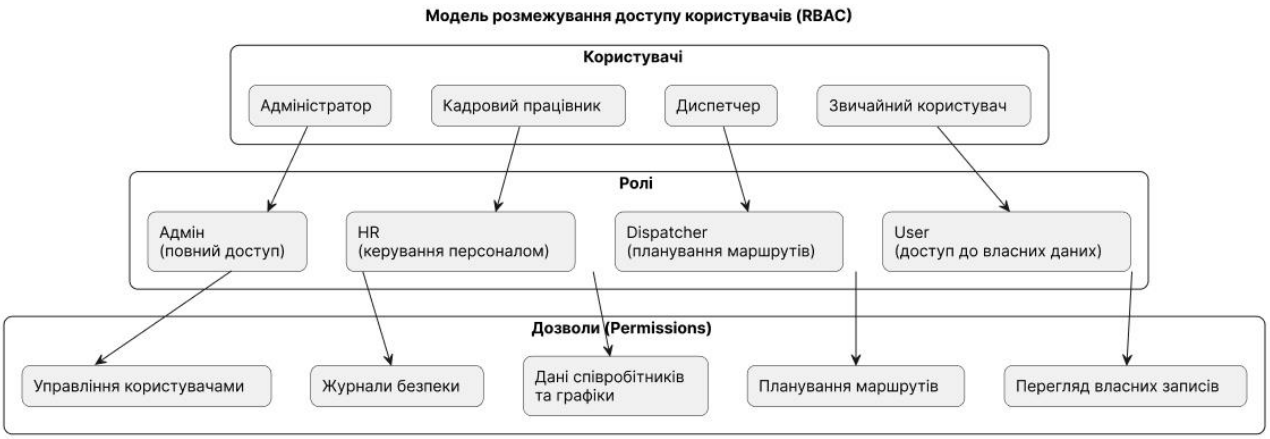


Рисунок Г.6 - Модель розмежування доступу користувачів до ресурсів системи на основі ролей (RBAC)