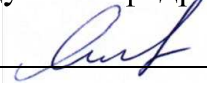


МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"

Завідувач кафедри ІУСТ



к.т.н, доц. Михелєв І.Л.

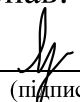
" ____ " _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "бакалавр"

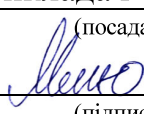
на тему: «Розроблення web-застосунку для залізничного вокзалу»

Виконав: студент групи


_____ Яременко Д.О.
(підпис)

Керівник роботи:

викладач


_____ (посада, науковий ступінь вчене звання)
(підпис) Меліксетов О.І.

м. Миколаїв – 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій
Спеціальність 122 "Комп'ютерні науки"
Освітня програма "Комп'ютерні науки"

"ЗАТВЕРДЖУЮ"

Гарант освітньої програми

к.т.н, доц. Гайда А.Ю.

" ____ " ____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти "бакалавр"

Студенту Яременко Дар'ї Олександрівні

(прізвище, ім'я, по батькові)

1. Тема роботи: Розроблення web-застосунку для залізничного вокзалу

Керівник роботи Меліксетов О.І.

Затверджені наказом ректора № 295-уч від " 27 " березня 2025 року.

2. Термін подання роботи: 20 червня 2025 р.

3. Вихідні дані до проекту (роботи) ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем,

4. Перелік питань, що належать до розробки (найменування розділів) Розробка концепції інформаційної системи залізничного вокзалу, Розробка технічного проекту інформаційної системи залізничного вокзалу, Реалізація проекту інформаційної системи залізничного вокзалу, Охорона праці

5. Перелік презентаційних матеріалів Титульний слайд, Мета роботи, Завдання, Об'єкт та Предмет дослідження, Актуальність, Організаційна структура вокзалу, Аналіз існуючих систем та виявлені недоліки, Модель концепції інформаційної системи для залізничного вокзалу, Формування вимог до інформаційної системи, Архітектура та технології розробки, Концептуальна модель даних, Логічна модель даних, Фізична модель даних, Реалізація web-застосунку, Подальший розвиток, Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	<u>Меліксетов О.І</u>	03.02.2025	22.02.2025
2	<u>Меліксетов О.І</u>	22.02.2025	21.03.2025
3	<u>Меліксетов О.І</u>	21.03.2025	25.04.2025
4	<u>Меліксетов О.І</u>	25.04.2025	07.06.2025

7. Дата видачі завдання 03 лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

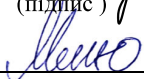
№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1	Аналіз предметної галузі	13.02.2025	
2	Постановка задачі	15.02.2025	
3	Розробка концепції	22.02.2025	
4	Розробка проекту ІС	21.03.2025	
5	Реалізація проекту	25.04.2025	
6	Розробка питань з охорони праці	07.06.2025	
7	Розробка графічних матеріалів	12.06.2025	
8	Подання роботи на перевірку на плагіат	10.06.2025	
9	Подання роботи рецензенту	15.06.2025	
10	Подання роботи на захист	20.06.2025	

Студент


(підпис)

Яременко Д.О.
(ПІБ)

Керівник роботи


(підпис)

Меліксетов О.І
(ПІБ)

АНОТАЦІЯ

У кваліфікаційній роботі було проведено дослідження створення web-застосунку для покращення обслуговування клієнтів залізничного вокзалу. Основною метою цієї роботи є розробка та втілення функціонального web-застосунку для комфортного доступу до інформаційних ресурсів та спрощення взаємодії з сервісами. Також визначено завдання як: Аналіз потреб користувача; Проектування архітектури для системи та пошук необхідних заходів; Розробка основних модулів та проведення перевірки на ефективність. Об'єктом виступає інформатизація залізничного вокзалу, а предметом є web-застосунку для оптимізації роботи. Також було використано методи аналізу, розробка програмного забезпечення (а саме HTML, CSS, JavaScript, Java, SpringBoot) та проектування інтерфейсу. Результатом є концепція прототипу web-застосунку з функціоналами для клієнтів(розклад потягів, пошук потягів, купівля або повернення квитків, особистого кабінету та повідомлення про затримки потягів) та адміністраторів(управління даними(потяги, вагони, місця у вагоні, розклад, цінами), придивляння статистики та підтримки клієнтів). Web-застосунок призначений для покращення обслуговування, зменшення черг та підвищення роботи залізничного вокзалу. Також приділено увагу забезпечення безпеки даних та аналізу поведінки в критичних ситуаціях.

Ключові слова: Web-застосунок, залізничний вокзал, клієнт, адміністратор, розклад поїздів, купівля квитків, HTML, CSS, JavaScript, Java, SpringBoot, управління даними.

ANNOTATION

In the qualification work, a study was conducted to create a web application to improve customer service at the railway station. The main goal of this work is to develop and implement a functional web application for comfortable access to information resources and simplify interaction with services. The tasks were also defined as: Analysis of user needs; Designing the architecture for the system and finding the necessary measures; Development of basic modules and conducting an efficiency test. The object is the informatization of the railway station, and the subject is a web application to optimize work. Methods of analysis, software development (namely HTML, CSS, JavaScript, Java, SpringBoot) and interface design were also used. The result is a concept of a prototype web application with functionalities for customers (train schedule, train search, ticket purchase or refund, personal account and notification of train delays) and administrators (data management (trains, cars, seats in the car, schedule, prices), statistics monitoring and customer support). The web application is designed to improve service, reduce queues and increase the work of the railway station. Attention is also paid to ensuring data security and analyzing behavior in critical situations.

Keywords: Web application, railway station, client, administrator, train schedule, ticket purchase, HTML, CSS, JavaScript, Java, SpringBoot, data management.

ЗМІСТ

ЗМІСТ	3
Вступ	4
Розділ 1. Розробка концепції інформаційної системи залізничного вокзалу	6
1.1 Опис об'єкту дослідження	6
1.2 Аналіз предметної області	11
1.3 Недоліки роботи існуючі системи	12
1.4 Існуючі варіанти рішення проблеми	13
1.5 Формування вимог до інформаційної системи	15
1.6 Концепція інформаційної системи	19
1.7 Постановка задачі.....	27
Висновки до розділу 1.....	31
Розділ 2. Розробка технічного проекту інформаційної системи	33
2.1 Загальносистемні рішення	33
2.2 Рішення по інформаційному забезпеченню	44
2.2.1 Розробка концептуальної моделі даних.....	45
2.2.2 Розробка логічної моделі даних.....	49
2.2.3 Розробка фізичної моделі даних	51
Висновки до розділу 2.....	52
Розділ 3. Реалізація проекту інформаційної системи	54
Висновки до розділу 3.....	59
Розділ 4. Охорона праці	60
Висновки до розділу 4.....	70
Висновки.....	72
Список використаних джерел	74
Додаток А – Інструкція користувача	76
Додаток Б – Технічне завдання.....	81
Додаток В – Фрагмент програмного коду	85

ВСТУП

В сучасному світі у розвиток інформаційних технологій та зростання мобільності населення зумовлюють модернізацією сервісів для залізничного вокзалу. Забезпечення клієнтів оперативним та зручним доступом до інформації, наприклад розклад поїздів, послуги залізничного вокзалу та наявність місць у вагоні для продажу квитків, має важливий фактор для підвищення якості обслуговування клієнтів. Часто існуюча система не повною мірою задовольняють потреби клієнтів, які проявляються у довгих чергах до кас та інформаційних стійок, особливо в період високого попиту. Через недостатній рівень цифрового моніторингу клієнтського потоку приводить до нерівномірного розподілу клієнтів (пасажирів) та створення затворів в окремих частинах залізничного вокзалу. Обмежений функціонал в існуючих сервісів, таких як система «Укрзалізниця» яка не забезпечує належний рівень інформування про зміни в розкладі є актуальним завданням. Розробка web-застосунку дозволить покращити якість обслуговування клієнтів для надання розкладу поїздів, онлайн покупки або повернення квитків, а також оптимізує роботу адміністратора через автоматизацію просів для управління даними. Метою є розробка web-застосунку для залізничного вокзалу для покращення обслуговування клієнтів та оптимізації його функціонування, а також зручності управління даними для адміністратора. Для досягнення мети визначенні такі завдання як:

Розкрити та визначити потреби клієнтів та адміністраторів в можливостях web-застосунку;

Розробити архітектури системи яка забезпечує масштабованість, продуктивність та безпеку даних;

Характеризувати та реалізувати основні модулі web-застосунку які включають інтерфейс для клієнта та адміністратора;

Виявити ефективність та відповідність поставленим вимогам для розробленого web-застосунку;

Сформувати підходи для забезпечення безпеки в функціюванні web-застосунку у підвищеної небезпеки.

Об'єктом дослідження є процес інформатизації функціонування залізничного вокзалу, яка включає такі аспекти як взаємодія клієнтів, адміністраторів та інформаційних систем в яких існує проблема застарілих підходів та неефективності.

А предметом дослідження є web-застосунок для оптимізації обслуговування клієнтів та управління даними для адміністраторів на залізничному вокзалі. Він відображає тему кваліфікаційної роботи та є об'єктом аналізу та розробки.

РОЗДІЛ 1. РОЗРОБКА КОНЦЕПЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ЗАЛІЗНИЧНОГО ВОКЗАЛУ

1.1 Опис об'єкту дослідження

Залізничний вокзал є важливим транспортним вузлом, який забезпечує пасажирські послуги та взаємодію між іншими видами транспорту. Це комплекс з багатьма функціями, який включає інфраструктуру, організаційні процеси та система обслуговування клієнтів, які необхідні для безперервної роботи залізничного транспорту.

Соціальне та економічне значення визначає технологію роботи будь-якого вокзалу. Ця технологія поєднує всі завдання, забезпечуючи чіткість, організованість та високий рівень обслуговування пасажирів, а також технічну частину роботи вокзалу. Технічною характеристикою є існуючі пристрої та технічні засоби вокзалу, його генеральний план з усіма приміщеннями, перонними шляхами та платформами, привокзальними площами, спорудами та спорудами. Економічна частина відображає значення вокзалу у житті того геополітичного місця, де він розташований, особливо якщо це столиця держави чи адміністративно-господарський та промисловий центр. Перевезення пасажирів залізницями має переваги перед іншими видами транспорту в галузі безпеки, надійності, регулярності руху, пропускної та провізної спроможності тощо. З урахуванням доступності, масовості та більшого значення в майбутньому ясно, але залізничний вокзал має велике соціально-економічне значення і є базою для координації та взаємодії всіх видів транспорту. І тому забезпечується узгодження розкладу руху поїздів, автобусів, літаків тощо. буд. На вокзалах залізниць знаходяться цілодобово працюючі автостоянки, готелі, підприємства торгівлі, вузли зв'язку, ресторани, банки, бізнес-центри, пункти медичної допомоги та ін. Це дозволяє забезпечити мінімальні втрати часу для пасажирів та створює найбільші зручності при поїздках та пересадках. Виробнича риса відображає кількість

далеких, місцевих і приміських поїздів, що прибувають і вирушають; кількість пасажирів, що прибувають і відправляються за видами сполучення (з виділенням транзитних пасажирів, які чекають подальшого слідування, а також пересадка з поїзда в поїзд); час відправлення та прибуття приміських пасажирів по днях тижня та доби; обсяги обробки багажу, пошти та ручної скарби; кількість оформлених проїзних документів у квиткових касах; число проданих у приміських касах квитків (зокрема оборотних, пільгових, абонементних тощо.); кількість переробленого багажу та вантажобагажу після прибуття, відправлення та транзиту; число місць ручної скарбниці, що приймається до камер зберігання, та інші операції.

Пасажирські перевезення відрізняються за такими видами:

- 1) перевезення в прямому сполученні – це пасажирські перевезення в границях двох і більше залізниць;
- 2) перевезення в місцевому сполученні – це перевезення пасажирів у границях даної залізниці;
- 3) приміські перевезення – це перевезення поблизу крупних промислових центрів;
- 4) міжнародне залізничне сполучення – перевезення між Україною та іноземними державами [1].

При забезпеченні актуальності цифрової трансформації залізничного вокзалу потрібно розглянути такі кроки як: Управління ресурсами залізничного вокзалу варто покращити для контролю пасажирського потоку, платформ та кас; Врахування зростає потреб на пасажирські обслуговування для зручності використання web-сайтів для комфорту та скорочення часу очікування; Значення технологій на розвиток транспортних систем в плані інтернет речей та хмарних сервісів. Це також сприяє автоматизації обслуговувань клієнтів, покращення безпеки інформаційних даних та швидкість обробки запитів.

Але незважаючи на актуальність існують проблеми, наприклад, довгі черги до кас або інформаційних стійок, якщо особливо це під час свят.

Відсутній цифровий моніторинг приводить до нерегулярного розподілу пасажирського потоку, затори у точках на залізничному вокзалі. Застосування автоматичних систем для управління має полегшити процес. А також, багато залізничних вокзалів використовують старі паперові квитки та систему продажів квитків, що зупиняють процес придбання. У переході на електронні сервіси черги скорочуються, послуги доступнішими та витрати на операції зменшаються.

Для розуміння як організовано залізничний вокзал потрібно зрозуміти структуру. На рис. 1.1 зображено організаційна структура залізничного вокзалу.

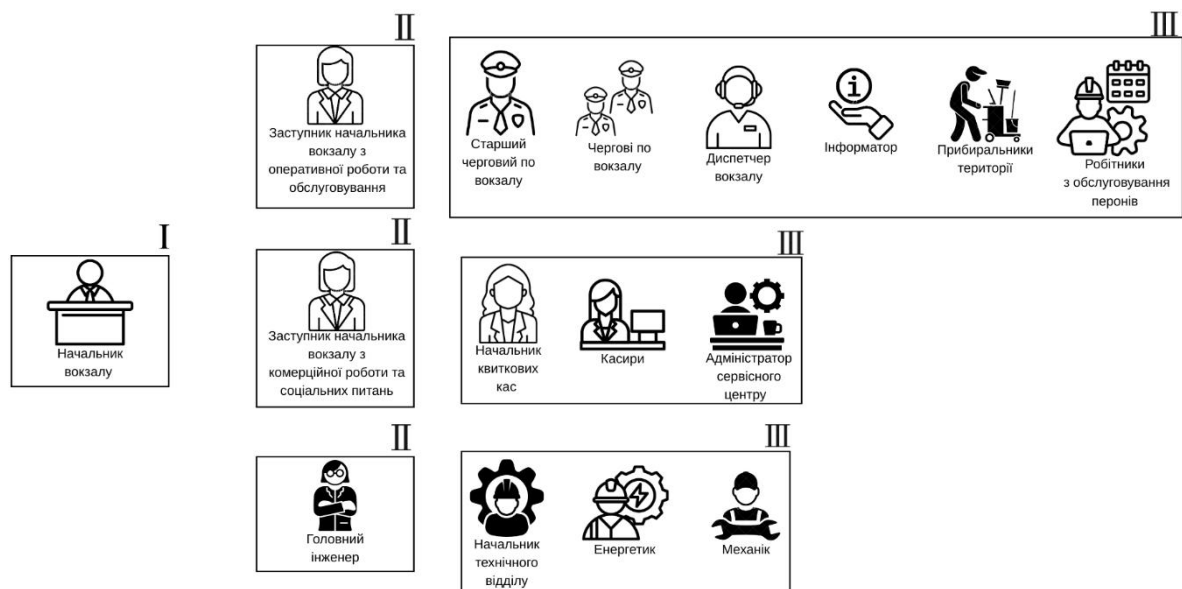


Рисунок 1.1- Організаційна структура залізничного вокзалу.

На рис.1.1 зображено трехрівнева структура с кожним підпорядкуванням. Сама структура залізничного вокзалу є чіткою ієрархією управління.

Головний в структурі знаходиться Начальник вокзалу, він здійснює загальний контроль за діяльністю залізнично вокзального комплексу. Ключові обов'язки це забезпечення ефективної роботи всіх служб залізничного вокзалу, гарантування безпеки клієнтів (пасажирів) та працівників, підтримання

порядку на залізничному вокзалі, організація та підтримка співпраці з іншими організаціями.

Другий рівень структури складає з ключових посад управління, який кожен з них відповідає за різні напрямки діяльності залізничного вокзалу.

Перший це заступник начальника вокзалу з оперативної роботи та обслуговування який відповідає за організацію руху поїздів та якість обслуговування пасажирів. Його обов'язок це координація роботи чергових по вокзалу, диспетчера вокзалу, інформаційної служби та служб вокзалу, що відповідають за утримання території та платформ. Якщо поточна ситуація з розкладом може змінитися або кілька поїздів можуть скасуватися чи затриматись, його реакція має заключне значення для порядку на залізничному вокзалі, ефективного завантаження/відправлення та якісне обслуговування з обмеженими ресурсами.

Другий це заступник начальника вокзалу з комерційної роботи та соціальних питань який відповідає за комерційну діяльність вокзалу, а саме очолює каси та здає в оренду комерційні площі та працює у сфері соціального забезпечення працівників вокзалу.

Третій це головний інженер який відповідає за технічний стан будівлі, будов, інженерних мереж вокзального комплексу, опалення, електропостачальних та водопостачальних систем. У випадку пошкодження транспортних засобів інфраструктури виконує заходи для забезпечення бесперебойної роботи терміналу та оперативного відновлення причинених завдань.

Третій рівень структури представляє працівники для керівників другого рівня. У заступника начальника вокзалу з оперативної роботи та обслуговування працюють:

Старший черговий по вокзалу, наглядає за дотриманням правил перебування на території вокзалу та організовує роботу чергових. Його увага приділяється забезпеченню безпеки та оперативному реагуванню на позаштатні ситуації в умовах підвищеної загрози.

Чергові по вокзалу, які відповідають за громадський порядок, стежать за дотриманням пасажирами правил, надають первинні довідки та надають необхідну допомогу.

Диспетчер вокзалу, який відповідає за оперативне управління пасажиропотоками, відповідає за своєчасне інформування про рух поїздів і взаємодію з іншими службами в режимі реального часу.

Інформатор, який надає пасажирам актуальну усну та візуальну інформацію про розклад руху, наявність вільних місць, правила проїзду та інші важливі питання.

Прибиральники вокзалу відповідають за підтримання чистоти та порядку на привокзальній площі та інших зовнішніх територіях вокзалу, забезпечуючи належні санітарні умови.

Робітники з обслуговування перонів підтримують порядок і безпеку на перонах, а також допомагають пасажирам під час посадки та висадки з вагонів.

У заступника начальника вокзалу з комерційної роботи та соціальних питань працюють:

Начальник квиткових кас, який відповідає за організацію квиткових кас, контроль за процесом продажу квитків і облік грошей.

Касири, які здійснюють продаж квитків та допомагають пасажирам з будь-якими питаннями, пов'язаними з проїздом.

Адміністратор сервісного центру, відповідальний за надання пасажирам послуг.

У головного інженера працюють:

Начальник технічного відділу який відповідає за справність усього технічного обладнання та інженерних систем вокзалу, а також за організацію та контроль роботи технічного персоналу.

Енергетик, відповідає за безперебійне та надійне електропостачання вокзального комплексу, а також за обслуговування електричних мереж і обладнання.

Механік, який відповідає за справність механічного обладнання, такого як ліфти, ескалатори, вентиляція та опалення (за її наявності).

Завдяки цієї організаційної структури, залізничний вокзал ефективно працює та пропонує обслуговування клієнтів. У випадку нестабільності важливо чітко координувати захід всього управління та реагувати на виклики, які пов'язані з безпекою та обслуговуванням клієнтів.

Можна зробити висновок що залізничний вокзал є складною та багатофункціональним об'єктом, який відіграє роль у транспортному інфраструктурі. Дослідження показує, що його робота забезпечує організаційні процеси, систему клієнтського обслуговування та інтеграцію інфраструктури. Для виявлення можливостей для оптимізації діяльності транспортного розширення, наступним етапом є проведення аналізу предметної області, який дозволить розглянути процеси, технології та взаємозв'язок між різними елементами системи для залізничного вокзалу.

1.2 Аналіз предметної області

Для виявлення слабких сторін та можливостей покращенню системи web-застосунку залізничного вокзалу, буде проведено аналіз предметної області.

В Україні базовим інструментом для купівлі квитків через онлайн є система «Укрзалізниця». Укрзалізниця - державне акціонерне товариство залізничного транспорту загального користування, національний перевізник вантажів та пасажирів, метою діяльності якого є задоволення потреб у безпечних та якісних залізничних перевезеннях у внутрішньому та міжнародному сполученні, забезпечення ефективного функціонування та розвитку залізничного транспорту та інше [2]. Функції цієї платформи є пошук потяга та придбання квитків, перегляд розкладу потягів, та доступ до особистого кабінету клієнта. Але в той же час в Україні впроваджено кілька автоматизованих систем для керування рухом потягів, централізація та робота

сигналізації. Ці системи були створені для того, щоб підвищити безпеку та зробити роботу залізничного транспорту ефективніше.

Було проведений аналіз предметної області який показує наявність базової інфраструктури для купівлі квитків через онлайн в Україні, представленою системою «Укрзалізниця». Але було виявлено обмеження у функціях та інтеграційних можливостях, що свідчать про неповну відповідність сучасним потребам клієнтів та вимогам управлінню даними. З цих виявлених проблем, наступним кроком є розгляд недоліків роботи існуючих систем, що дозволить зрозуміти шляхи для вирішення та розробки доскональних рішень проблеми.

1.3 Недоліки роботи існуючі системи

Після проведення аналізу предметної області було виявлено наявність базової інфраструктури для купівлі квитків в Україні. Але, було ідентифіковано декілька обмежень, які негативно впливають на комфорт клієнтів та ефективність управління. Для рішення цього питання потрібно детальний огляд недоліків роботи існуючої системи «Укрзалізниці», які потребують усунення для підвищення якості обслуговування клієнтів та ефективності управління.

Одним із суттєвих недоліків є обмеження функції управління розкладом поїздів та інформування пасажирів при інших змінах. В умовах динамічної транспортної сфери, де частіше виникають необхідність в коригуванні графіків, система не часто забезпечує бажану гнучкість та своєчасне доведення актуальної інформації до клієнтів. Це може призвести до дезорієнтації клієнтів та негативно вплинути на досвід з користуванням сервісом. Також, спостерігається недостатній рівень інтеграції з іншими сервісами, які значно підвищують зручність при плануванні подорожей. Відсутність взаємодії з інформаційними системами міського транспорту, сервіси для бронювання проживання та іншими сервісами, створює складність для клієнтів, які

вимушені використовувати роздільні ресурси на різних етапах організації поїздки. Важливо визначити, що існуюча система недостатньо зважає на індивідуальні потреби та вподобані клієнта. Відсутність механізмів персоналізації контенту створює труднощі в процесі отримання важливої інформації про оптимальні маршрути, спеціальні пропозиції які зацікавляють клієнта та актуальні знижки на основі вказаних переваг або попередньої історії подорожей. Також, потребують покращення ефективності зворотного зв'язку з клієнтами та якості служби технічної підтримки. Клієнти часто стискаються з проблемою при спробі отримати необхідну допомогу або залишити відгук про роботу системи, яка може знижувати рівень лояльності. Недоліком є обмежені можливості системи при зборі, аналізі даних та формування аналітичного звіту. Відсутність інструментів для обробки інформації про популярні напрямки, завантаженість рейсів, пасажиропотоків та інших операційних показників обтяжує процес прийняття рішення про управління на стратегічному рівні та має обмеження у потенціалі для операційної діяльності залізничного вокзалу.

Можна зробити висновок що було проведено аналіз в якому виявлено недоліки в роботі існуючої системи, які негативно впливають як на комфорті клієнтів, так і ефективності управління даними. Для удосконалення цих напрямків, необхідно проаналізувати існуючі варіанти рішення цих проблем, які можуть адаптовані для покращення системи. Та відкриває змогу для подальшого вивчення та розробки підходів до управління залізничним вокзалом.

1.4 Існуючі варіанти рішення проблеми

При проведенні аналізу недоліків існуючої системи було виявлено ключові напрямки, які потребують покращення. Для визначення цих проблем потрібно розглянути існуючі варіанти та підходи, які можуть бути адаптивні для покращення зручності використання та функціонування системи.

Перше це модернізація існуючої інфраструктури на розширення можливостей функцій. Вона включає такі впровадження як розробка та інтеграція модулів, які дозволяють вносити зміни до розкладу, гнучкі інструменти для управління даними, автоматичні сповіщення для клієнтів при поправках в розкладі через різні засоби комунікації (SMS-повідомлення або Email). Розширення програмного інтерфейса з сторонніми сервісами для забезпечення взаємодії з системами міського транспорту та іншими корисними ресурсами для подорожей. Це буде дозволяти клієнтам планувати поїздки заздалегідь та комплексно, використовуючи єдині цифрові сервіси. Також, впровадити систему аналізу даних клієнта про попередні поїздки, їхню активність у системі та їхнім вподобанням для надання інформації про маршрути, спеціальні пропозиції, знижки та індивідуальні рекомендації. Важливим напрямком також є покращення взаємодії з клієнтами та якості обслуговування, таких як покращення зворотного зв'язку, а саме впровадження зручних інструментів для отримання відгуків чи запитань від клієнтів. А також, забезпечення швидкого реагування на запити та надання допомоги. Потрібно зосередитися на розробці та інтеграції системи для збирання та аналізування звітів про потік клієнтів, популярні напрямки, фінансові показники та інші операційні дані. Також, створення інструментів для візуального відображення даних та формування звітів, які допоможуть приймати рішення на стратегічному рівні та оптимізувати операційну діяльність.

Отже, можна зробити висновок що було проведено аналіз існуючих варіантів рішення, який можна сформувавши комплексний підхід для модернізації системи, на усунення недоліків та покращення якості обслуговування клієнтів. Для цього потрібно сформувавши вимоги до системи та розробити план для поетапної реалізації, який надає можливості для подальшого розвитку та функціонування залізничного вокзалу.

1.5 Формування вимог до інформаційної системи

На основі аналіз існуючих варіантів рішення, потрібно сформувати вимоги до інформаційної системи які будуть враховувати потреби клієнта та адміністратора. Також, передбачає розробка web-застосунка з функцією купівлі квитків з безпечною інтеграцією банківської платіжної системи. Інформаційна система для автоматизації роботи залізничного вокзала має відповідати таким вимогам.

До функціональних вимог для Клієнтів входить Перегляд розкладу (можливість пошуку поїздів за різними критеріями (станція відправлення, станція прибуття, дата, час), відображення детальної інформації про поїзд (номер поїзда, час відправлення/прибуття, маршрут, наявність місць, вартість), фільтрація та сортування результатів пошуку), Купівля квитків (вибір місць на схемі вагона, введення даних пасажирів, онлайн-оплата квитків (різні способи оплати), формування електронного квитка, можливість збереження даних пасажирів для швидкого заповнення при наступних покупках), Повернення квитків (можливість повернення куплених квитків відповідно до правил перевізника, розрахунок суми повернення, формування заявки на повернення, відображення статусу повернення коштів), Історія та статус квитків (перегляд історії куплених та заброньованих квитків, відстеження статусу бронювання, можливість завантаження електронного квитка), Повідомлення (отримання повідомлень про затримки поїздів, отримання нагадувань про майбутні поїздки, можливість налаштування отримання повідомлень), Особистий кабінет (можливість редагування особистих даних, збереження історії поїздок).

До функціональних вимог для Адміністратора входить Редагування інформації про поїзди (адміністратор може додавати нові поїзди, редагувати інформацію про існуючі поїзди (маршрут, час, зупинки), видаляти поїзди), Управління станціями (адміністратор може додавати, редагувати, та видаляти станції), Управління вагонами поїздів (адміністратор може додавати,

редагувати, та видаляти вагони), Управління місцями у вагоні (адміністратор може додавати, редагувати, та видаляти місця), Редагування розкладу (адміністратор може змінювати розкладу руху поїздів, додавати/видаляти станцій-зупинок, встановлювати час відправлення/прибуття), Статистика (адміністратор може переглядати статистику продажів квитків, аналіз пасажиропотоку на визначені маршрути), Редагування цін (адміністратор може змінювати ціни на квитки, встановлювати тарифи, встановлювати знижки), Відповіді на запитання клієнтів (адміністратор може переглядати запитання від клієнтів, надавати відповіді на запитання).

Для розробки та функціонування системи автоматизації роботи залізничного вокзалу потрібно обрати технологічні та технічні засоби, а також програмне забезпечення для web-застосунка. При виборі критерій потрібно враховувати масштабованість (мати здатність системи для обробки клієнтів та даних), продуктивність (для швидкої роботи системи), безпека (захист даних та платіжної системи), зручність при розробці та підтримки (наявність розробників для підтримки системи та документацій для технологій).

При розробці користувацького інтерфейсу, який має зрозумілий та забезпечений швидким доступом до інформації використовується наступні технології:

HTML (стандартизована мова розмітки документів для перегляду вебсторінок у браузері [4].): Створення структури web-сторінок;

CSS (спеціальна мова стилю сторінок, що використовується для опису їхнього зовнішнього вигляду [5].): Візуального оформлення інтерфейсу та забезпечення адаптивного дизайну;

JavaScript (динамічна, об'єктно-орієнтована прототипна мова програмування. Використовується для створення сценаріїв веб-сторінок, що надає можливість на боці клієнта взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки [6].): Забезпечення інтерактивності, обробки подій користувача, асинхронних запитів до сервера.

Для забезпечення обробки бізнес-логіки, зберігання даних та взаємодії з клієнтською частиною будуть використані наступні технології:

Java (об'єктно-орієнтована мова програмування, випущена 1995 року компанією «Sun Microsystems» як основний компонент платформи Java [7].): Надійна та масштабована платформа для розробки великих корпоративних систем;

Spring Framework (програмний фреймворк з відкритим кодом та контейнери з підтримкою інверсії керування для платформи Java [8].) Для організації серверної частини, керування взаємодією між різними бізнес-компонентами (наприклад, обробка замовлень, управління даними про поїзди) та забезпечення надійного програмного забезпечення для клієнтської частини, спрощуючи розробку, тестування та подальшу підтримку системи.

MySQL (одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування [9].) Ефективного зберігання та управління структурованою інформацією про поїзди, розклади, квитки, користувачів та інших критично важливих даних, забезпечуючи цілісність та доступність цієї інформації для всіх компонентів системи.

Kafka (розподілене сховище подій і платформа для їх багатопотокового оброблення [10].) Підходить для обробки великих потоків даних у реальному часі.

Середовище розробки буде використовуватися IntelliJ IDEA (комерційне інтегроване середовище розробки для різних мов програмування від компанії JetBrains [11].) яка буде забезпечувати необхідну функціональність для ефективної розробки на обраних мовах програмування.

При розробці та забезпечення функціонування системи автоматизації роботи залізничного вокзалу використовується комплекс технічних засобів таких як:

Сервер (для розміщення серверної частини інформаційної системи, бази даних та web-застосунків використовується сервер з відповідними технічними характеристиками (продуктивність процесора, обсяг оперативної пам'яті, дисковий простір) для забезпечення стабільної роботи та обробки очікуваного навантаження).

Операційна система сервера (На сервері буде встановлено операційну систему (Windows), яка забезпечить необхідне середовище для роботи серверного програмного забезпечення (Java, Spring Framework, MySQL, Kafka)).

Web-сервер (для обробки HTTP-запитів від клієнтських браузерів та забезпечення доступу до web-застосунку буде налаштовано web-сервер).

Мережеве обладнання (для забезпечення зв'язку між клієнтами, сервером та зовнішніми системами буде використано відповідне мережеве обладнання (маршрутизатори, комутатори)).

Система резервного копіювання (для запобігання втраті даних у разі збоїв або аварій буде налаштовано автоматичну систему резервного копіювання важливих даних (бази даних, конфігураційні файли тощо).

Безпека (Для захисту даних користувачів, адміністраторів та платіжної інформації буде впроваджено комплекс заходів безпеки, використання SSL-сертифікатів для шифрування трафіку, регулярні оновлення програмного забезпечення та моніторинг системи на предмет загроз).

Система моніторингу (для контролю за працездатністю системи, виявлення потенційних проблем та реагування на них).

Комп'ютери (з необхідним програмним забезпеченням для адміністраторів та касирів, що забезпечать ефективне виконання їхніх функціональних обов'язків, орієнтовно 5-7 комп'ютерів, достатньо продуктивним процесором для продуктивної роботи).

Кваліфікований персонал (для підтримки та подальшого розвитку інформаційної системи буде команда кваліфікованих фахівців, включаючи адміністраторів серверів, адміністраторів баз даних та розробників).

Обраний комплекс технологічних та технічних засобів є основою для безпечного, ефективного та стабільного функціонування для розробки інформаційної системи автоматизації роботи залізничного вокзалу.

Отже, робимо висновок що було сформульовано функціональні вимоги до інформаційної системи. Для реалізації були обрані комплекс технологій та техніки що відповідають критеріям та відповідають поставленій меті для створення інформаційної системи для автоматизації роботи залізничного вокзалу. Та для подальшого розробки концепції інформаційної системи.

1.6 Концепція інформаційної системи

Також для розробки інформаційної системи необхідно визначити межі та функції. Для цього потрібно розробити концепцію. На рис. 1.2 зображено модель концепції інформаційної системи залізничного вокзалу.

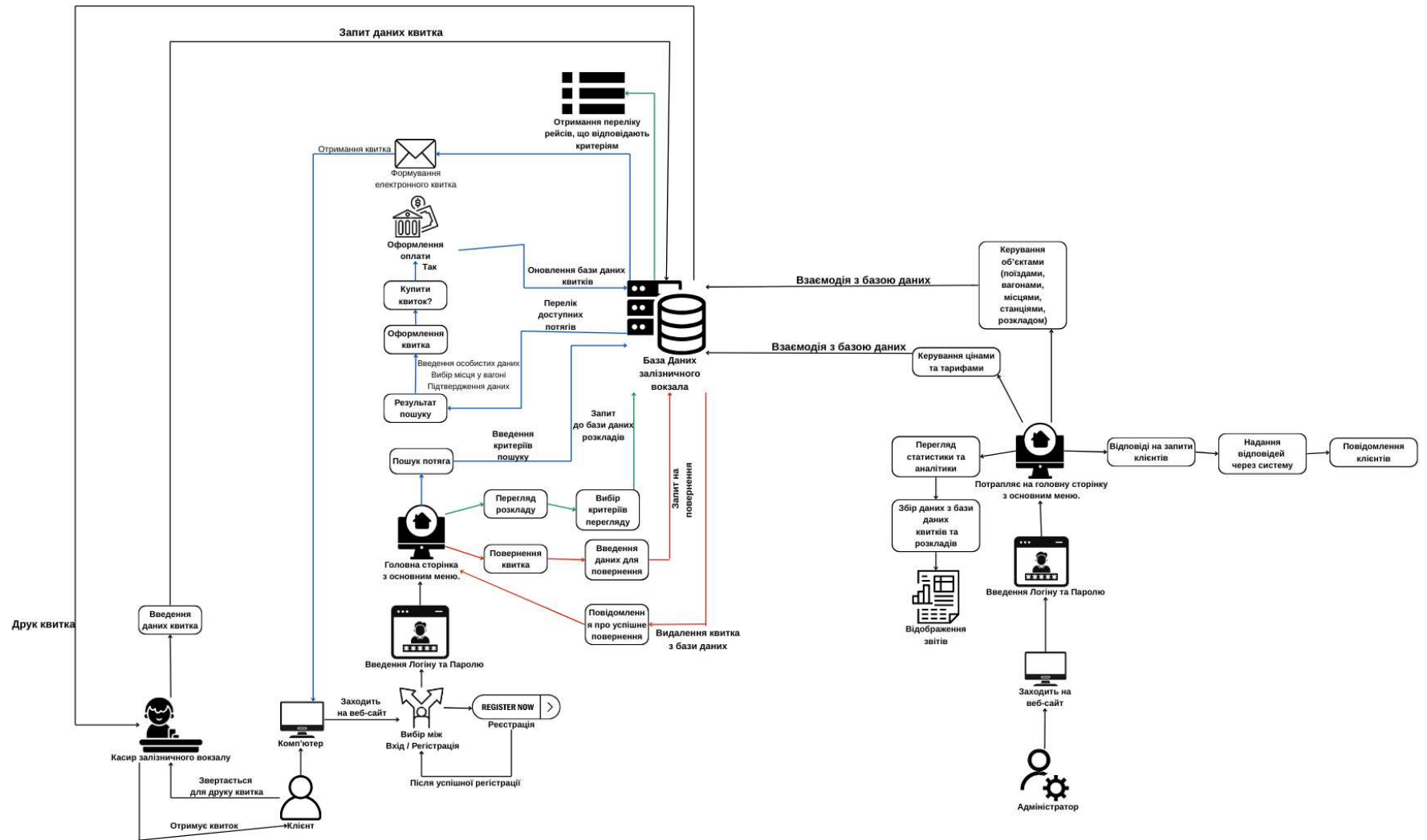


Рис.1.2 - Модель концепції інформаційної системи залізничного вокзалу

Клієнт, зайшовши на web-застосунок для бронювання білетів, спочатку бачить сторінку Вхід та Регістрація. При реєстрації він вводить дані (ПІБ, Тип Документа на якій він буде бронювати квитки, сам номер документа для підтвердження особи, електронна пошта для відправки повідомлення, номер телефону також для відправки повідомлення в форматі SMS, логін для входу, пароль також для входу до особистого кабінету), далі після успішної реєстрації переходить до частини Входу та клієнт вводить дані Логін та Пароль на який він зареєстрований.

Після входу до головної сторінки бачить меню з блоками для пошуку потяга, пошуку розкладу для вказаного потяга, також купівлі квитка, повернення квитка та перегляд статусу квитка. Головна сторінка містить поле пошуку квитків з можливістю вибору станцій, дат та часу, актуальну інформацію про розклад та зміни, швидкий доступ до популярних маршрутів та послуг, а також посилання на особистий кабінет та інших розділи сайту.

У блоці з пошуку потяга він може ввести бажані критерії поїздки а саме: станцію відправлення, станцію прибуття, бажану дату та час відправлення. Після введення критеріїв та їх підтвердження, система відправляє запит до сервера, де зберігається база даних розкладів. У відповідь клієнт отримує перелік доступних потягів, що відповідають його запиту.

Ознайомившись з переліком, клієнт може вибрати потрібний потяг, натиснувши на нього. Після цього він переходить на сторінку з детальною інформацією про обраний потяг, де відображається повний маршрут, час у дорозі, наявність вільних місць у різних типах вагонів та їх ціни. Сторінка пошуку квитків надає користувачу розширену форму для пошуку необхідних рейсів. Форма включає поля для вибору станцій відправлення та прибуття, бажаної дати та часу подорожі, а також додаткові фільтри, такі як клас вагона. Залежно від доступності для обраного потяга, користувачу може бути надана можливість вибору конкретних місць на інтерактивній схемі вагона. Після застосування критеріїв пошуку користувачу відображається список доступних результатів. Кожен результат містить детальну інформацію про поїзд,

включаючи маршрут прямування, час відправлення та прибуття, вартість проїзду та актуальну інформацію про наявність вільних місць.

Переконавшись у правильності вибору, клієнт переходить до оформлення квитка. На сторінці введення особистих даних він заповнює необхідну інформацію: ПІБ, контактний номер телефону, адресу електронної пошти. Також клієнт може вибрати тип вагона та конкретне місце, якщо така опція доступна. При необхідності придбання квитків для кількох пасажирів, клієнт може додати їх дані.

На наступному кроці клієнту надається сторінка для остаточного підтвердження вибраного рейсу та введених даних. Він уважно перевіряє всю інформацію та обирає дію: "Купити квиток".

Якщо клієнт обирає "Купити квиток", система перенаправляє його на сторінку платіжної системи, де він вводить дані своєї банківської картки для здійснення оплати. Після успішної оплати, інформація про покупку заноситься до бази даних, а на вказану клієнтом електронну пошту надходить лист з електронним квитком у форматі PDF. Електронний квиток містить деталі поїздки та дані пасажирів, необхідні для посадки в потяг. Сторінка бронювання квитків автоматично заповнюється даними про обраний поїзд, маршрут, вагон та місце після того, як користувач зробив свій вибір на попередніх етапах. Для зручності користувачів передбачена можливість збереження особистих даних для їх швидкого автоматичного заповнення при наступних бронюваннях. На цій сторінці також надається перелік доступних додаткових послуг, таких як постільна білизна, харчування та напої, з відображенням вартості кожної з них. Система автоматично розраховує загальну вартість квитка з урахуванням обраних додаткових послуг.

Клієнт також має можливість переглянути історію своїх замовлень та статус придбаних або заброньованих квитків у відповідному розділі особистого кабінету на сайті. Особистий кабінет надає користувачам можливість переглядати історію своїх придбаних квитків, оформляти їх

повернення, редагувати особисту інформацію, а також налаштовувати параметри отримання важливих повідомлень та сповіщень від системи.

У випадку затримки рейсу або інших важливих змін, клієнт може отримати автоматичне повідомлення на свою електронну пошту або номер телефону, вказаний при реєстрації.

Клієнт має можливість переглядати розклад не лише на поточну дату, але й на будь-які інші доступні дати в майбутньому чи минулому, використовуючи зручний календарний інтерфейс. Це дозволяє планувати поїздки заздалегідь або перевіряти розклад на конкретний день. Сторінка розкладу надає користувачам актуальну інформацію про рух поїздів. На цій сторінці передбачено відображення розкладу як для конкретно обраного маршруту (між двома станціями), так і для визначеної залізничної станції (інформація про всі поїзди, що відправляються або прибувають на цю станцію). Уся інформація про зміни та затримки відображається чітко та помітно, щоб користувачі могли своєчасно отримувати актуальні дані та коригувати свої плани у разі необхідності.

Також у випадку якщо клієнт передумав їхати, він може скористатися блоком Повернути квиток. Для повернення квитка йому потрібно ввести Id номер квитка та номер карточки для повернення коштів. Далі після цього в нього з'являється повідомлення що повернення коштів прийде на вказану карточку при покупці за 24 години та натикає на ОК. Квиток який Клієнт бронював видаляється з бази даних, а в свою чергу звільняється місце у вагоні. Сторінка повернення квитка надає користувачам можливість скасувати придбані квитки у випадку зміни планів. Для ініціювання процесу повернення клієнту необхідно скористатися блоком "Повернути квиток". На сторінці повернення квитка вводиться ID номер квитка, який зазначений в електронному квитку або історії замовлень в особистому кабінеті, а також номер банківської картки, що використовувалася для оплати квитка, для здійснення повернення коштів.

Взаємодія Адміністратора до Системи:

Так само для адміністративної панелі повинна забезпечувати зручне та ефективне управління контентом а також користувачами веб-застосунку. Загальні вимоги до панелі мають бути такими: Безпечний доступ та авторизація, зручний та інтуїтивно зрозумілий інтерфейс для адміністраторів, швидка навігація та доступ до необхідних функцій, можливість налаштування прав доступу для різних користувачів.

Як клієнт, так і адміністратор також заходить на сторінку Логін та переходить на головну сторінку для Адміністратора.

Адміністратор має доступ до спеціального інтерфейсу для керування інформаційною системою. Основні функції АРМ включають:

Редагування інформації про потяги: Адміністратор може додавати нові потяги, змінювати їх характеристики (номер та назву маршруту, тип потяга, загальна кількість вагонів та статус), а також видаляти. Сторінці "Управління поїздами" має можливість переглядати список усіх зареєстрованих у системі поїздів з їх основною інформацією (номер, назва маршруту, тип, кількість вагонів, статус). Також містить функціонал для Додавання нового поїзда (введення даних нового поїзда, де адміністратор зазначає всі необхідні характеристики), Редагування поїзда (редагування інформації про обраний поїзд, де можна змінювати його характеристики), Видалення поїзда (видалення обраного поїзда з системи (з попереднім підтвердженням)).

Редагування інформації про вагонів: Адміністратор може додавати нові вагони, змінювати їх характеристики (номер потяга для якого буде присвоєно вагон, тип вагону, загальна кількість місць у вагоні та статус), а також видаляти. Сторінці "Управління вагонами" відображається перелік усіх вагонів, зареєстрованих у системі, з інформацією про їхній номер потяга, тип, загальну кількість місць та статус. Також доступний функціонал для Додавання вагона(введення даних нового вагона, де необхідно вказати номер потяга, тип вагона, кількість місць та статус), Редагування вагона(редагування характеристик обраного вагона), Видалення вагона(видалення обраного вагона з системи).

Редагування інформації про місце у вагоні: Адміністратор може додавати нові місця для кожного вагону, змінювати їх характеристики (номер вагону для якого буде присвоєно місце та статус), а також можна видаляти. Сторінка "Управління місцями у вагонах" надає інструменти для керування окремими місцями у вагонах. Відображається перелік вагонів, при виборі якого адміністратор може переглядати список місць у ньому з їхніми номерами та статусами. Також доступний функціонал для Додавання місць (введення даних нових місць для обраного вагона із зазначенням їхніх номерів та статусів), Редагування місця (можливість зміни статусу обраного місця (наприклад, доступне, зайняте)), Видалення місць(видалення окремих місць з вагона).

Редагування інформації про станції: Адміністратор може додавати нові станції, змінювати їх характеристики (назву, місто де знаходиться ця станція, область, статус). Сторінці "Управління станціями" адміністратор бачить список усіх залізничних станцій, зареєстрованих у системі, з їхніми назвами, містом розташування, областю та статусом. Також доступний функціонал для Додавання станції(введення даних нової станції з необхідною інформацією), Редагування станції(редагування інформації про обрану станцію), Видалення станції(видалення обраної станції з системи).

Редагування розкладу: Адміністратор може додавати нові рейси, змінювати існуючі (час відправлення/прибуття, станції), а також скасовувати рейси у разі потреби. Сторінці "Управління розкладом" представлений розклад руху поїздів. Адміністратор має можливість Додавання рейсу (введення даних нового рейсу, де вказується поїзд, маршрут, час відправлення та прибуття на кожній станції), Редагування рейсу (редагування інформації про обраний рейс (час, станції тощо)), Скасування рейсу (скасування запланованого рейсу).

Перегляд статистики: Система надає адміністратору можливість переглядати різні статистичні звіти, такі як кількість проданих квитків за певний період, найбільш популярні напрямки, завантаженість рейсів тощо. Ці звіти можуть бути представлені у вигляді графіків для кращого аналізу. Сторінці "Звітність та аналітика" адміністратору доступні різні статистичні

звіти та аналітичні дані. Залежно від обраного звіту, відображається інформація про кількість проданих квитків за певний період, популярність маршрутів, завантаженість рейсів у вигляді таблиць та графіків для полегшення аналізу.

Редагування цін: Адміністратор має можливість змінювати вартість квитків залежно від типу вагона, часу до відправлення, попиту та інших факторів. На сторінці "Редагування цін" адміністратор має інструменти для керування цінами на квитки. Доступна можливість зміни вартості залежно від типу вагона, часу до відправлення, попиту та інших налаштованих параметрів.

Відповіді на питання клієнтів: У випадку, якщо клієнти звертаються через форму зворотного зв'язку або іншим способом, адміністратор може переглядати їхні запитання та надавати відповіді безпосередньо через АРМ. Система може передбачати сповіщення про нові запитання. На сторінці "Підтримка користувачів" адміністратор може переглядати список звернень від користувачів, що надійшли через форму зворотного зв'язку. Для кожного запиту передбачена можливість перегляду детальної інформації та надсилання відповіді безпосередньо з інтерфейсу. Система може мати функціонал сповіщень про нові запити.

Для забезпечення стабільної роботи сайту, адміністратор може використовувати сервіси моніторингу для відстеження доступності та продуктивності веб-сайту.

Для обробки онлайн-платежів використовується інтеграція з платіжною системою, налаштованою відповідно до вимог залізничної компанії.

Для автоматичного надсилання електронних листів (підтвердження бронювання/покупки, електронні квитки, повідомлення) використовується сервіс розсилки електронної пошти, також налаштований згідно з потребами компанії.

Дивлячись на те, чи має клієнт можливість придбати квиток онлайн, він завжди може звернутися до касира в залізничному вокзалі. Якщо касир використовує веб-сайт для оформлення, він спочатку авторизується в системі.

Потім, отримавши від клієнта інформацію про бажану поїздку, вводить у відповідні поля: станції відправлення та прибуття, дату та час. Система показує доступні варіанти, і касир обирає потрібний поїзд, а за потреби - конкретні місця за схемою вагона. Далі він вносить дані клієнта з наданих документів, додає необхідні додаткові послуги, перевіряє замовлення та підтверджує його. Після вибору способу оплати (готівка чи картка), касир проводить оплату через систему або POS-термінал. Веб-сайт створює електронний квиток, який касир друкує та видає клієнту.

Отже, при розробці концепції інформаційної системи залізничного вокзалу, було визначено ключові можливості та межі майбутньої системи. Було охоплено основні аспекти взаємодії як клієнтів, так і адміністратора з системою. Розроблена концепція системи є комбінованим та має ключові потреби як для клієнта (зручно та швидко придбати квиток) так для адміністратора (ефективні інструменти для керування всіма об'єктами роботи залізничного вокзалу). Ця концепція призначається основою для подальшого проектування та розробки інформаційної системи для досягнення поставлених цілей.

1.7 Постановка задачі

Після побудови концепції та формулювання функціональних вимог потрібно сформулювати постановку задачі майбутньої системи для автоматизації процесів роботи в залізничному вокзалі. Основна мета є формулювання задач, які потрібно рішити в процесі створення системи. Головним аспектом є розробка зрозумілого та функціонального web-застосунку, який забезпечить клієнтам зручний та швидкий доступ до всіх необхідних інформацій та послуг, а адміністрація надає ефективні інструменти для управління цілої системи.

Для досягання цієї поставленої мети потрібно вирішити задачі, які пов'язані з розробкою інтерфейсу та функціональністю системи для клієнтів, такі як:

Розробка візуального та привабливого інтерфейсу для клієнтів, який буде відповідати вимогам дизайну та зручному використанню;

Реалізація реєстрації та авторизації для клієнтів;

Розробка головного меню web-застосунку, який має забезпечити поле для пошуку потяга з вибором станції відправлення та станції прибуття, дати та час, відображення актуальної інформації про розклад або змін у розкладі, доступ до популярних маршрутів та послуг, посилання на особистий кабінет та інші розділи;

Реалізація функціоналу пошуку поїздів, який дозволяє клієнту вводити критерії поїздки (станція відправлення, станція прибуття, дата та час), отримувати перелік доступних поїздів, що відповідають запиту, перегляд інформації про обраний поїзд (а саме маршрут, час у дорозі, наявність вільних місць у різних типах вагонів, ціни);

Розробка функціоналу пошуку квитків з розширеними можливостями, який дає розширену форму для вибору станцій, дати та часу, а також тип вагону, вибір конкретних місць на інтерактивній схемі вагона, відображення списку результатів з інформацією про поїзд, час відправлення та прибуття, вартість та наявність місць;

Реалізація процесу оформлення квитка який буде забезпечувати автоматичного заповнення даних про обраний поїзд, маршрут, вагон та місце, забезпечувати можливість додавання даних кількох пасажирів та надавання сторінки для остаточного підтвердження вибраного потяга та введення даних;

Перенаправляти клієнта на сторінку платіжної системи після підтвердження замовлення, забезпечити для нього безпечне введення даних банківської картки, обробляти успішні платежі та заносити інформацію про покупку до бази даних, автоматично надсилати електронний квиток на вказану електронну пошту у форматі PDF, забезпечити можливість збереження

особистих даних клієнта для швидкого автоматичного заповнення при наступних купівлях;

Надання інформації про доступні додаткові послуги (постільна білизна, харчування, напої) з відображенням вартості та забезпечення автоматичного розрахунку загальної вартості квитка з урахуванням обраних послуг;

Розробка особистого кабінету клієнта, який буде надавати можливість як, перегляд історії придбаних квитків, редагувати особисту інформацію (наприклад номер документа, прізвище та ім'я), реалізувати систему повідомлень для інформування клієнтів про затримки поїздів через електронну пошту або SMS, забезпечення можливості перегляду розкладу на будь-які доступні дати за допомогою зручного календарного інтерфейсу, як для конкретних маршрутів, так і для визначених станцій, з чітким відображенням змін та затримок;

Розробити функціонал повернення квитків, що дозволяє клієнту вводити номер квитка та номер банківської картки для повернення коштів, отримувати повідомлення про успішне повернення та орієнтовний термін зарахування коштів, автоматично видаляти повернутий квиток з бази даних та звільняти відповідне місце;

Забезпечити можливість оформлення квитків через касира на залізничному вокзалі з використанням web-застосунку, включаючи авторизацію касира, пошук рейсів, вибір місць, введення даних клієнта, додавання послуг, вибір способу оплати та друк електронного квитка.

Також, для досягання цієї поставленої мети потрібно вирішити задачі, які пов'язані з розробкою інтерфейсу та функціональністю системи для адміністратора, такі як:

Розробка безпечної системи авторизації адміністраторів для доступу до адміністративної панелі;

Створення зручного та зрозумілого інтерфейсу адміністративної панелі, що буде забезпечувати швидку навігацію та доступ до всіх необхідних

функцій, візуально зрозуміле представлення даних, також можливість налаштування права доступу для різних адміністраторів;

Реалізувати функціонал управління інформацією про поїзди, а саме, перегляд списку усіх зареєстрованих поїздів з їхньою основною інформацією (номер, назва маршруту, тип, кількість вагонів, статус), додавання нових поїздів, вказуючи всі необхідні характеристики, редагування інформацію про існуючі поїзди (номер, назва маршруту, тип, загальна кількість вагонів, статус), видалення поїзди з системи (з попереднім підтвердженням);

Розробити функціонал управління вагонами поїздів, що дозволяє адміністратору переглядати перелік усіх вагонів, зареєстрованих у системі, з інформацією про їхній номер потяга, тип, загальну кількість місць та статус, додавання нових вагонів, вказуючи номер потяга, тип вагона, кількість місць та статус, редагування характеристики обраного вагона та видалення вагонів з системи;

Реалізувати функціонал управління місцями у вагонах, що дозволяє адміністратору переглядати список вагонів та місця в обраному вагоні з їхніми номерами та статусами, додавати нові місця для обраного вагона із зазначенням їхніх номерів та статусів, редагувати статус обраного місця (наприклад, доступне, заброньоване), видаляти окремі місця з вагона;

Розробити функціонал управління інформацією про станції, що дозволяє адміністратору переглядати список усіх залізничних станцій, зареєстрованих у системі, з їхніми назвами, містом розташування, областю та статусом, додавати нові станції з необхідною інформацією, редагувати інформацію про обрану станцію, видаляти станції з системи;

Реалізувати функціонал управління розкладом руху поїздів, що дозволяє адміністратору додавати нові рейси, вказуючи поїзд, маршрут, час відправлення та прибуття на кожній станції, редагувати інформацію про існуючі рейси (час, станції тощо), скасовувати заплановані рейси;

Розробити функціонал перегляду статистики та аналітики, що надає адміністратору можливість переглядати звіти про кількість проданих квитків

за певний період, аналізувати популярність маршрутів, також оцінювати завантаженість рейсів та відображати статистичні дані у вигляді графіків;

Реалізувати функціонал редагування цін на квитки, що дозволяє адміністратору змінювати вартість квитків залежно від типу вагона, попиту та інших налаштованих параметрів та встановлювати тарифи та знижки;

Розробити функціонал обробки запитів клієнтів, що дозволяє адміністратору переглядати список звернень від користувачів, що надійшли через форму зворотного зв'язку, переглядати детальну інформацію за кожним запитом, надавати відповіді на запитання безпосередньо з інтерфейсу адміністративної панелі;

Налаштувати сервіс відправлень через email для автоматичного надсилання повідомлень клієнтам.

Отже, після детальної розробки концепції та формування функціональних вимог було сформульовано постановку задачі яка потрібна для створення інформаційної системи автоматизації роботи залізничного вокзалу. Також постановка задачі охоплює всі деталі для розробки системи, та є основою для подальшого проектування та реалізації інформаційної системи.

Висновки до розділу 1

При дослідженні предметної області концепцією інформаційної системи було отримано важливі теоретичні та практичні результати, які забезпечать основу для проектування та реалізації системи. Було детально описано об'єкт дослідження, а саме залізничний вокзал як складний об'єкт з багатьма функціями та об'єднує організаційні процеси та інфраструктуру. Було досліджено його організаційну структуру, що дозволяє виявити його ключові елементи та взаємозв'язки в роботі. Також було проведено аналіз предметної області, де було досліджено існуюча система для купівлі квитків в Україні, а саме «Укрзалізниця». Та було виявлено недоліки цієї системи, що вказує про неповну відповідальність потребам клієнтам та управління даних. Але було

сформульовано існуючі варіанти для вирішення цих недоліків, який дозволить визначити підхід до модернізації системи. Ще було сформульовано вимоги до інформаційної системи які враховують потреби клієнта та адміністратора, який буде основою для подальшої розробки. Також на основі сформульованих вимог було створено концепції інформаційної системи, який показує основне бачення майбутньої системи, основні деталі та принцип взаємодії між ними. Та вирішальним етапом є постановка задачі де було визначено список задач які потрібно для реалізації концепції та функціональних вимог. Отже в цьому розділі було реалізовано поставлену мету, а саме розробити концепцію інформаційної системи залізничного вокзалу для подальшого проектування, яка буде покращувати рівень обслуговування клієнтів та оптимізує роботу управління даними для адміністраторів.

РОЗДІЛ 2. РОЗРОБКА ТЕХНІЧНОГО ПРОЕКТУ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Загальносистемні рішення

Основою досягнення цілей, які були визначені постановкою задачі є опис загальносистемних рішень які були прийняті на етапі проектування web-застосунку для залізничного вокзалу. Вибір цих параметрів є остаточне значення для реалізації проекту. Вони формують основу для створення структури системи, її функціональних можливостей, продуктивності, масштабованості, надійності та тривалого обслуговування. До цього підходу є комплексний аналіз діаграм використання та послідовності, робиться оцінка обраних технологій, механізми безпеки, методи обробки помилок та реєстрації подій та процес конфігурування. Це формує основу для створення якісний та стабільний програмний продукт, який здатний ефективно відповідати на встановлені вимоги та забезпечує комфорт клієнту. На рис. 2.1 зображено діаграма варіантів використання.

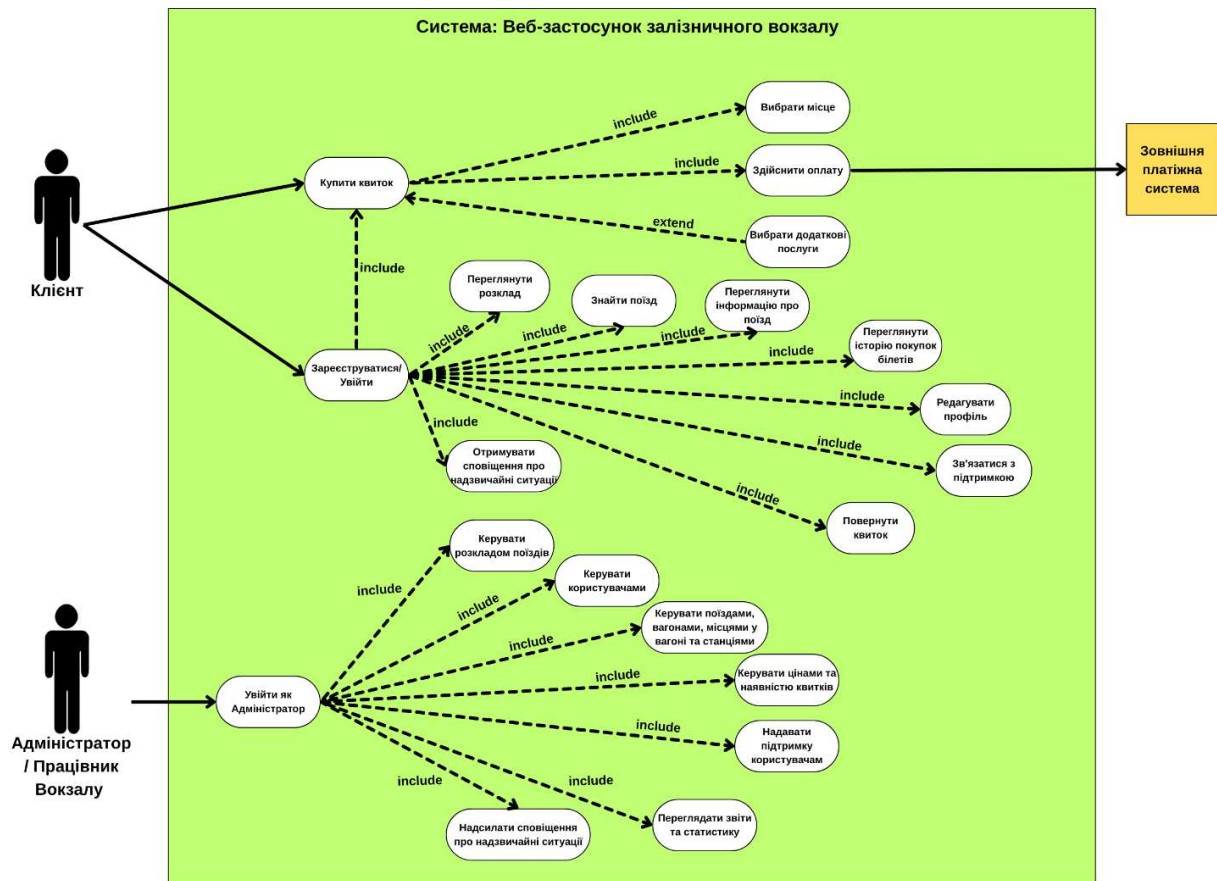


Рис.2.1- Діаграма варіантів використання

Для відображення функціональних вимог web-застосунку залізничного вокзалу було застосовано діаграму варіантів використання. Ця діаграма описує функціональність системи з погляду зовнішніх користувачів (акторів) та їх цілі.

Актори – це зовнішні сутності, які взаємодіють з web-застосунком. Обрані основні категорії акторів як Клієнт - це основний користувач системи, який отримує інформацію про розклад поїздів, придбати квитки, переглядати історію своїх поїздок або звертатися за підтримкою, Адміністратор або Працівник вокзалу - це персонал залізничного вокзалу, який виконує функції управління системними даними, забезпечення актуальності інформації, допомоги користувачам і моніторингу роботи системи та Зовнішня платіжна система - це автоматизована зовнішня система, що відповідає за обробку фінансових транзакцій під час процесу придбання квитків.

Дивлячись з погляду Клієнта існує варіанти використання систем:

Зареєструватися/Увійти який дозволяє клієнту створити новий обліковий запис або увійти вже в існуючий, щоб скористатися персоналізованими послугами;

Перегляд розкладу, де клієнт може ознайомитися з актуальним графіком руху потягів, користуючись різними критеріями, такими як дата та маршрут;

Пошук поїзда який надає можливість знайти потрібний поїзд за номером, маршрутом або іншими параметрами;

Інформація про поїзд дає клієнту доступну інформацію щодо обраного потяга, включаючи маршрут, зупинки та час відправлення та прибуття;

Купівля квитка дає клієнту можливість обрати маршрут, місце в вагоні та оформити покупку квитка;

Вибір місця це обов'язкова складова при купівлі квитка, що надає можливість вибору місця в вагоні;

Оплата є необхідним кроком у процесі купівлі, що дозволяє провести оплату через інтегровану платіжну систему, взаємодіючи із зовнішньою платіжною платформою;

Історія покупок дає клієнту переглядати список раніше придбаних квитків і свою історію поїздок;

Зв'язок з підтримкою дає можливість клієнту надіслати запит або питання в службу підтримки вокзалу;

Повернення квитка є процесом ініціації повернення вже придбаного квитка;

Редагування профілю дає змогу клієнтам оновлювати персональні дані у його обліковому записі;

Додаткові послуги є необов'язковим вибором, що дозволяє додати до покупки квитка такі послуги, як постільна білизна, харчування, напій;

Сповіщення про надзвичайні ситуації надається клієнту можливість отримувати сповіщення про затримки, скасування або інші важливі зміни.

Дивлячись з погляду Адміністратора існує варіанти використання системи:

Увійти як Адміністратор відкриває доступ до адміністративної системи за допомогою персональних облікових даних;

Керувати розкладом поїздів дає змогу додавати нові рейси, змінювати маршрути, коригувати час відправлення та прибуття, визначати платформи, а також видаляти рейси з розкладу;

Керувати розкладом поїздів дає можливість встановлювати ціни на квитки залежно від категорії поїздів, вагонів або місць, редагувати ціни та вручну регулювати наявність місць;

Переглядати звіти та статистику забезпечує доступ до звітів та аналітичних даних, включаючи обсяг проданих квитків, отримані прибутки, популярні напрямки та рівень завантаженості поїздів;

Керувати користувачами дозволяє переглядати список зареєстрованих осіб, створювати нові облікові записи для працівників вокзалу, оновлювати інформацію користувачів;

Надавати підтримку користувачам включає можливість опрацьовувати запити від пасажирів, відповідати на них та надавати необхідну консультацію чи допомогу;

Керувати поїздами, вагонами, місцями у вагоні та станціями має можливість управляти основною базою даних щодо потягів, вагонів, їхньої пасажиромісткості та списку залізничних станцій;

Надсилати сповіщення про надзвичайні ситуації являє собою функцію яка створює й відправляє повідомлення клієнтам у випадках надзвичайних ситуацій, наприклад затримки рейсів, скасування, евакуації або виникнення загроз.

Незважаючи на те, що Клієнт і Адміністратор виконують різні функції та мають окремі ролі в системі, взаємодія між ними забезпечується інтегрованим підходом. Ця взаємодія здійснюється через функціонал web-застосунку та є елементом, що гарантує чітку роботу системи. Клієнт формує запити через функцію Зв'язатися з підтримкою, які обробляються Адміністратором за допомогою опції Надавати підтримку користувачам. В свою чергу,

Адміністратор зможе надсилати загальні чи екстрені сповіщення через функцію Надсилати сповіщення про надзвичайні ситуації, які потім отримують Клієнти через механізм Отримувати сповіщення про надзвичайні ситуації. Адміністратор відповідає за оновлення та регулювання основної інформації системи, яка включає розклад поїздів, ціни, наявність місць, а також відомості про станції. Внесені у систему дані використовуються Клієнтами під час дій, як ознайомлення з розкладом, пошук поїздів чи оформлення квитків.

Отже, такий спосіб до моделювання функціональних вимог дозволяє встановити межі та можливості системи та забезпечити зрозумілу та ефективну взаємодію між її зовнішніми учасниками.

Розуміння дій кожного актора також потребує представлення вигляді діаграми послідовності, який є важливим інструментом в моделюванні програмних систем. Ця діаграма дає можливість візуалізувати взаємодію між об'єктами системи та показує обмін повідомленнями в хронологічному порядку. На рис. 2.2 зображено діаграма послідовності для клієнта.

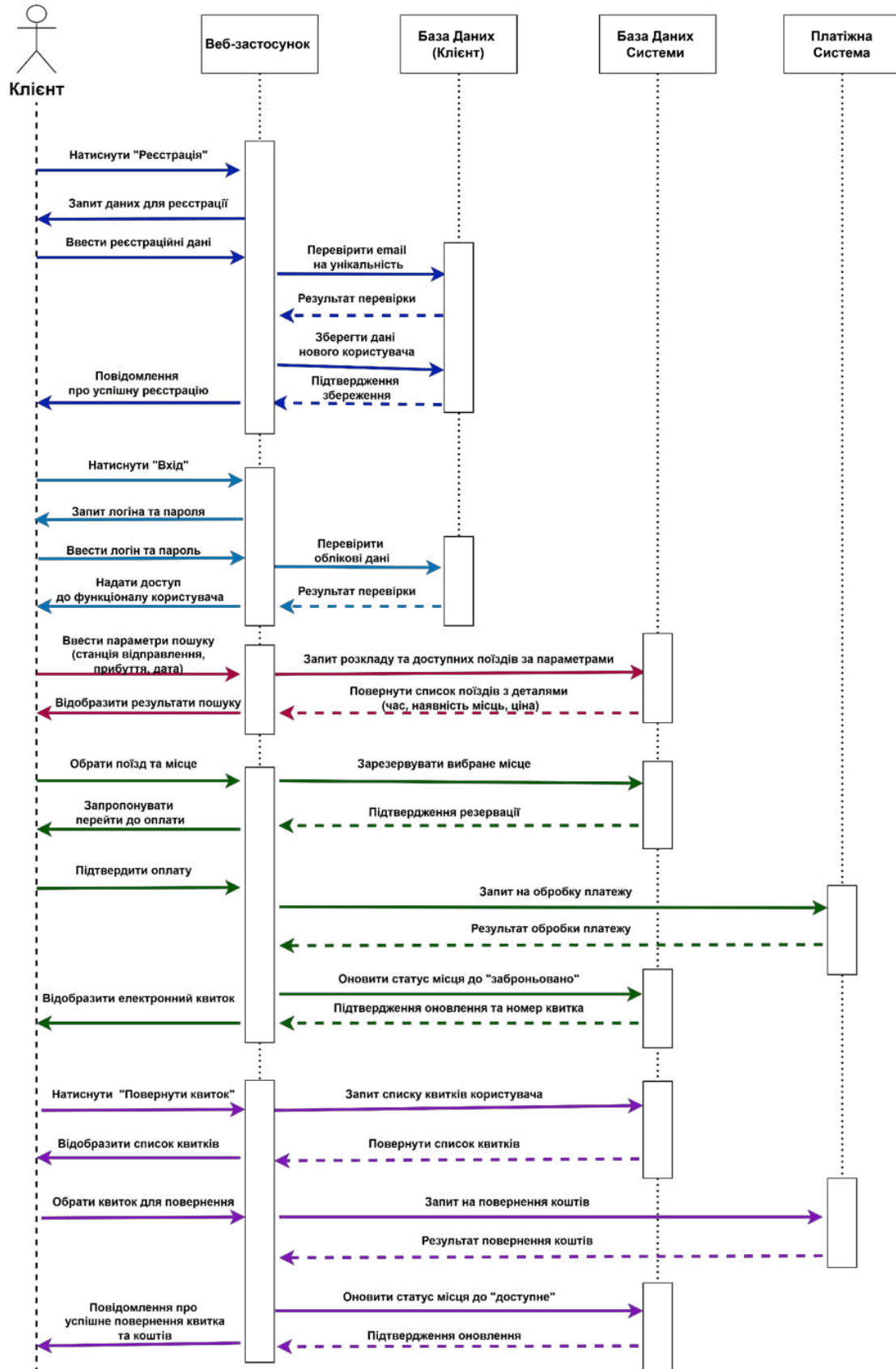


Рис.2.2- Діаграма послідовності для Клієнта

На рис.2.2 зображена діаграма послідовності яка демонструє потік взаємодії між актором Клієнт та основними компонентами системи: Веб-

застосунком, Базою Даних (Клієнта), Базою Даних Системи та Платіжною Системою. Ця модель показує процеси роботи системи, в яких охоплює дії з даними Клієнта, перевірки в системі та фінансові транзакції що підтверджується користувачем. Ці учасники представлені як Клієнт (актор, який взаємодіє з системою через інтерфейс web-застосунку та його роль полягає в ініціюванні операцій, наприклад збереження інформації, запуск платіжного процесу), Web-застосунок(виконує роль клієнтського інтерфейсу, який забезпечує зв'язок між клієнтом та системою, також виступає посередником для передачі запитів між актором та іншими компонентами), База Даних(Клієнт)(відповідає за збереження даних, які стосується Клієнтів та відповідає за актуальність інформації), База Даних Системи (центральне сховище системних даних, яка виконує роль компонента що відповідає за складні обчислення, обробку запитів, та інтеграцію з зовнішніми сервісами), Платіжна Система (зовнішній сервіс, який відповідає за обробку фінансових транзакцій).

Над цими учасниками існує певний аналіз потоку повідомлення між ними за етапами:

Для реєстрації клієнта має такі етапи: Клієнт переходить до Реєстрації, далі Веб-застосунок відправляє запит даних реєстрації до Клієнта, Клієнт вводить реєстраційні дані до Веб-застосунку для створення облікового запису, після чого Веб-застосунок надсилає дані до Баз Даних (Клієнт) для перевірки унікальності електронної пошти, Бази Даних (Клієнт) відправляє Веб-застосунку результати перевірки, далі Веб-застосунок зберігає дані нового користувача до Баз Даних (Клієнт), після чого Бази Даних (Клієнт) відправляє підтвердження збереження, а в свою чергу Веб-застосунок відправляє Клієнту повідомлення про успішну реєстрацію;

Для входу клієнта має такі етапи: Клієнт переходить до Вхід, далі Веб-застосунок відправляє запит на логін та пароль до Клієнта, Клієнт вводить логін та пароль до Веб-застосунку, після чого Веб-застосунок надсилає дані до Баз Даних (Клієнт) для перевірки облікових даних (чи дійсний логін та

пароль), Бази Даних (Клієнт) відправляє Веб-застосунок результати перевірки, далі Веб-застосунок надає доступ до функціоналу для Клієнта;

Для пошуку потяга має такі етапи: Клієнт вводить параметри пошуку (станція відправлення/прибуття, дату та час) до Веб-застосунку, в свою чергу Веб-застосунок надсилає запит розкладу та доступних поїздів за параметрами до Бази Даних Системи, База Даних Системи повертає список поїздів з деталями (час, наявність місць, ціна) до Веб-застосунку, та Веб-застосунок відображає результат пошуку;

Для купівлі квитка має такі етапи: Клієнт після знаходження потяга, обирає потяг та місце у вагоні, далі Веб-застосунок надсилає до Бази Даних Системи команду щоб зарезервувати вибране місце, далі База Даних Системи надсилає підтвердження резервації до Веб-застосунку, в свою чергу Веб-застосунок надсилає Клієнту пропозицію перейти до оплати, Клієнт підтверджує оплату у Веб-застосунку, Веб-застосунок надсилає запит на обробку платежу до Платіжної системи, далі Платіжна система надсилає результат обробки платежу до Веб-застосунку, далі Веб-застосунок відправляє до Бази Даних Системи команду для оновлення статусу місця до "заброньовано", База Даних Системи відправляє підтвердження оновлення та номер квитка до Веб-застосунку, та Веб-застосунок відправляє електронний квиток до Клієнта;

Для повернення квитка має такі етапи: Клієнт переходить до Повернути квиток, далі Веб-застосунок надсилає до Бази Даних Системи запит до списку квитків користувача, далі База Даних Системи повертає список квитків до Веб-застосунку, в свою чергу Веб-застосунок надсилає Клієнту список квитків які він купив, Клієнт обирає квиток для повернення у Веб-застосунку, Веб-застосунок надсилає запит на повернення коштів до Платіжної системи, далі Платіжна система надсилає результат повернення коштів до Веб-застосунку, далі Веб-застосунок відправляє до Бази Даних Системи запит для оновлення статусу місця до "доступне", База Даних Системи відправляє підтвердження

оновлення до Веб-застосунку, та Веб-застосунок відправляє повідомлення про успішне повернення квитка та коштів.

Також для розуміння дій адміністратора кожна його дія потребує представлення у вигляді діаграми послідовності. На рис.2.3 зображено діаграма послідовності для адміністратора.

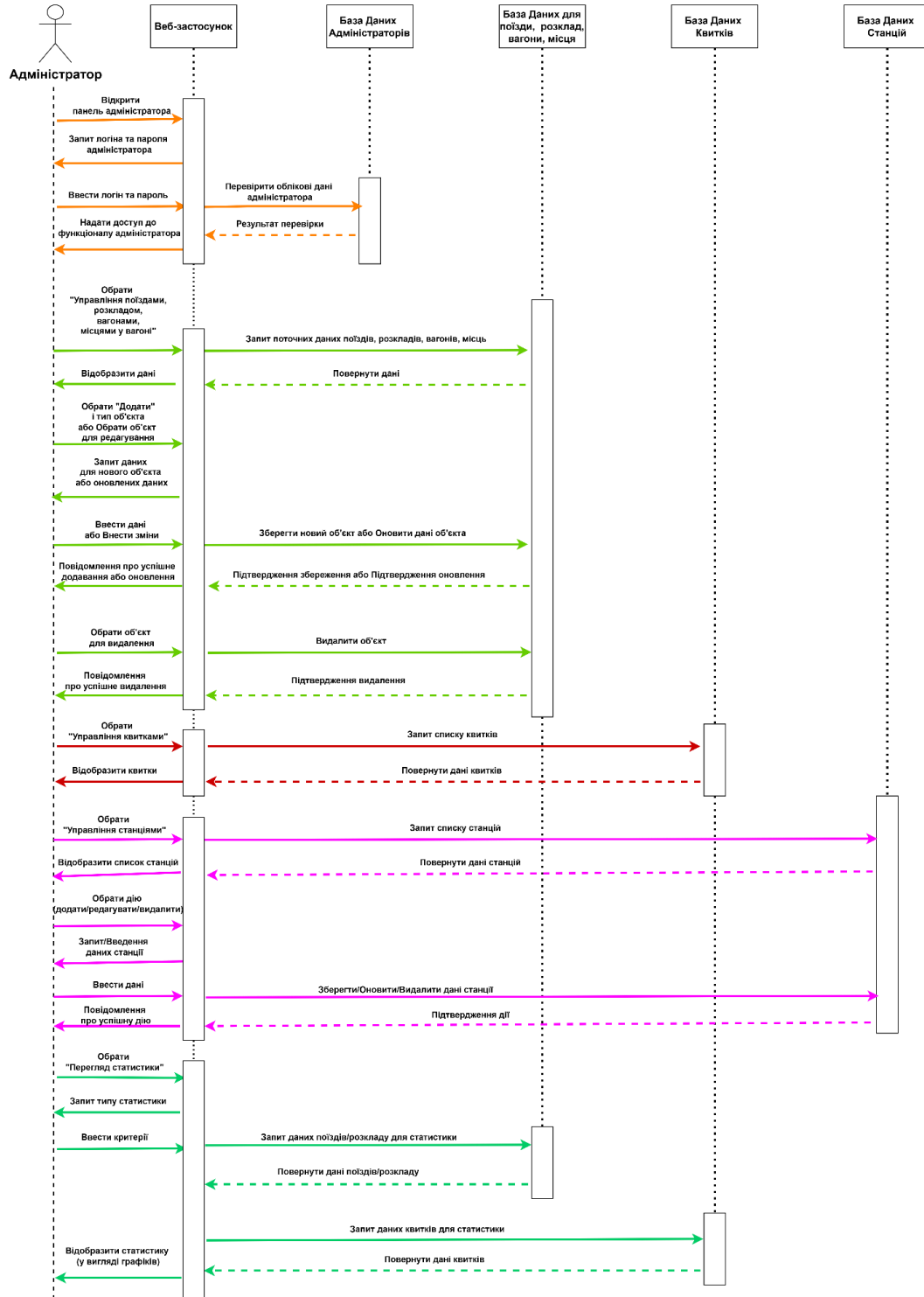


Рис.2.3- Діаграма послідовності для Адміністратора

На рис.2.3 зображена діаграма послідовності яка демонструє потік взаємодії між актором Адміністратором та основними компонентами системи: Веб-застосунком, Базою Даних Адміністраторів, Базою Даних для поїздів, вагонів, місць, Базою Даних Квитків та Базою Даних Станцій. Ця модель показує процеси роботи системи, в яких охоплює дії з керування даних про проїзд, вагонами, місцями та квитками, що ініціює Адміністратор через інтерфейс. Ці учасники представлені як Адміністратор (актор, який взаємодіє з системою через інтерфейс веб-застосунку та його роль полягає в ініціюванні операцій, наприклад керування даними поїздів, вагонів, місць, квитків та станцій), Веб-застосунок (виконує роль основного інтерфейсу, який забезпечує зв'язок між адміністратором та внутрішньою системою, також виступає посередником для передачі запитів між актором та іншими компонентами), База Даних(Адміністратор)(відповідає за збереження даних, які стосуються адміністратора та їхніми правами доступу), База Даних для поїздів, вагонів, місць (центральне сховище системних даних про поїзди, маршрути, типи вагонів, доступні місця та статус (доступні, заброньовані)), База Даних Квитків (сховище, який відповідає за зберігання інформації про випущені квитки, їхній статус, дані пасажирів, історію транзакцій, пов'язаних з квитками), База Даних Станцій (сховище що містить дані про залізничні станції, їхні назви, розташування та іншу необхідну інформацію для маршрутизації поїздів).

Над цими учасниками існує певний аналіз потоку повідомлень між ними за етапами:

Для входу до веб-застосунку має такі етапи: Адміністратор переходить до Вхід, далі Веб-застосунок відправляє запит на логін та пароль до Адміністратора, Адміністратор вводить логін та пароль до Веб-застосунку, після чого Веб-застосунок надсилає дані до Бази Даних Адміністратор для перевірки облікових даних (чи дійсний логін та пароль), Бази Даних Адміністратор відправляє Веб-застосунку результати перевірки, далі Веб-застосунок надає доступ до функціоналу адміністратора для Адміністратора;

Для управління даними по поїздам, розкладів, вагонів, місцями у вагоні (ці об'єкти можуть бути окремі) має такі етапи: Адміністратор переходить до Управління поїздами, розкладом, вагонами, місцями у вагоні в Веб-застосунку, в свою чергу Веб-застосунок надсилає запит на поточні дані поїздів, розкладів, вагонів, місць до Бази Даних для поїздів, вагонів, місць, далі База Даних повернути дані до Веб-застосунку, та Веб-застосунок відправляє дані Адміністратору. Якщо Адміністратор хоче обрати "Додати" і тип об'єкта який потрібно або обрати об'єкт для редагування, то він переходить до цих операцій у Веб-застосунку, Веб-застосунок надсилає запит даних для нового об'єкта або оновлених даних Адміністратору, після чого Адміністратор вводить дані або вносить зміни в Веб-застосунку, Веб-застосунок надсилає запит зберегти новий об'єкт або оновити дані об'єкта у Базу Даних, після чого База Даних надсилає підтвердження збереження або підтвердження оновлення в Веб-застосунок, та Веб-застосунок надсилає повідомлення про успішне додавання або оновлення Адміністратору. Якщо Адміністратор хоче обрати об'єкт для видалення то також він переходить до цих операцій у Веб-застосунку, Веб-застосунок надсилає запит у Базі Даних про видалення об'єкта, після чого База Даних надсилає підтвердження видалення в Веб-застосунок, та Веб-застосунок надсилає повідомлення про успішне видалення Адміністратору;

Для управління квитками має такі етапи: Адміністратор переходить до Управління квитками в Веб-застосунку, в свою чергу Веб-застосунок надсилає запит списку квитків до Бази Даних квитків, далі Бази Даних квитків повертає дані квитків до Веб-застосунку, та Веб-застосунок відображає квитки Адміністратору;

Для управління станціями має такі етапи: Адміністратор переходить до Управління станціями в Веб-застосунку, в свою чергу Веб-застосунок надсилає запит списку станцій до Бази Даних станцій, далі Бази Даних станцій повертає дані станцій до Веб-застосунку, та Веб-застосунок відображає список станцій Адміністратору. Якщо також Адміністратор хоче обрати дію (додати/редагувати/видалити) то він також переходить до цих операцій у Веб-

застосунку, в свою чергу Веб-застосунок надає запит Адміністратору про введення даних станції, Адміністратор вводить дані в Веб-застосунку, Веб-застосунок надсилає запит на зберегти/оновити/видалити дані станції до Бази Даних станції, далі База Даних надсилає підтвердження дії в Веб-застосунку, та Веб-застосунок надсилає повідомлення про успішну дію Адміністратору;

Для перегляду статистики має такі етапи: Адміністратор переходить до Перегляд статистики у Веб-застосунку, Веб-застосунок надсилає запит на критерії які хоче подивитися Адміністратор, Адміністратор вводить критерії для статистики у Веб-застосунку, в свою чергу Веб-застосунок надсилає запит даних поїздів/розкладу/квитків для статистики, далі База Даних повертає дані поїздів/розкладу/квитків до Веб-застосунка, та Веб-застосунок відображає статистику (у вигляді графіків) для Адміністратора.

Отже, було визначено архітектурні та функціональні принципи для проектування web-застосунку залізничного вокзалу. Для досягнення цих цілей проведено аналіз функціональних вимог системи за допомогою діаграми варіантів використання, які дозволили визначити взаємодію акторів з системою. Також було детально описано варіант використання для кожної ролі взявши всі аспекти взаємодії. Для візуалізації хронології взаємодії між об'єктами системи, розроблено діаграми послідовності для кожного актора (Клієнт, Адміністратор), яка відображає потік повідомлень та етапи роботи з погляду кожного актора. Таким чином, побудова діаграми варіантів використання та діаграм послідовності візуалізує та формує функціональні вимоги web-застосунку, які встановлюють межі та можливості системи та забезпечує ефективну взаємодію між компонентами та акторами.

2.2 Рішення по інформаційному забезпеченню

Для функціонування системи потрібно створити продуману та реалізовану систему інформаційного забезпечення. Потрібно розглянути основні аспекти які стосуються збирання, обробки та надання даних. Для цього

потрібно спроектувати концептуальну, логічну та фізичну модель даних для майбутньої бази даних.

2.2.1 Розробка концептуальної моделі даних

Для проектування web-застосунку потрібно визначити структуру даних, яка буде обробляти та зберігати дані в системі. Концептуальна модель даних є початковим та важливим етапом процесу, яка дозволяє визначити основні сутності предметної області та їх взаємовідносини на рівні абстракції. Вона показує розуміння даних всіх учасників проекту, також визначає об'єкти та їх характеристики. Ця модель буде представлена через ER-діаграму. На рис. 2.4 зображено концептуальна модель даних для залізничного вокзалу.

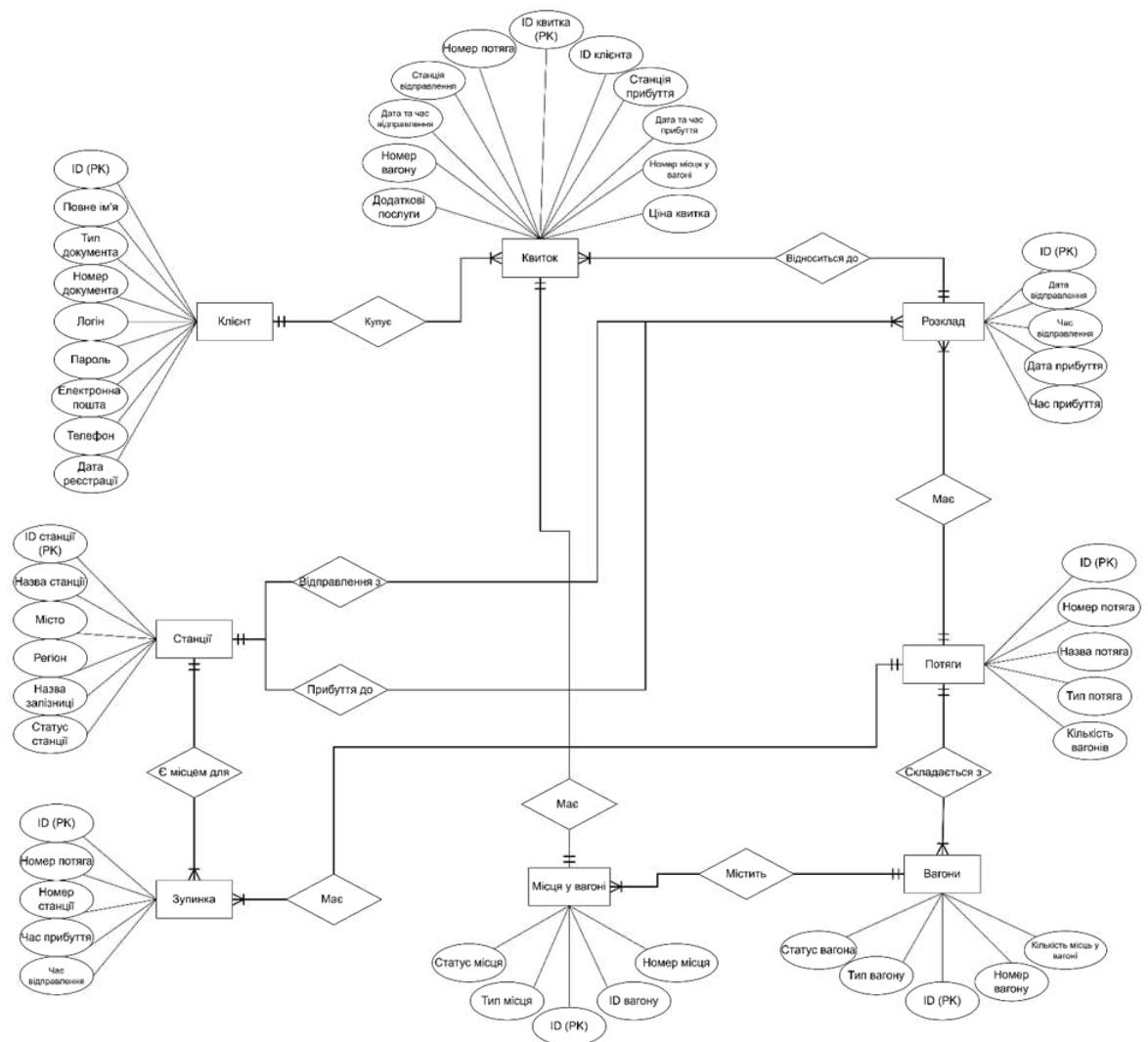


Рис.2.4- Концептуальна модель даних для залізничного вокзалу

У таблиці 2.1 наведено розподіл сутностей та атрибутів, які використовуються для створення концептуальної моделі даних.

Таблиця 2.1

Сутності та Атрибути концептуальної моделі даних

Сутність	Атрибути
Клієнт	ID
	Повне ім'я
	Тип документа
	Номер документа
	Логін
	Пароль
	Електронна пошта
	Телефон
	Дата реєстрації
Квиток	ID квитка
	Номер клієнта
	Номер потяга
	Станція відправлення
	Дата та час відправлення
	Станція прибуття
	Дата та час прибуття
	Номер вагону
	Номер місця у вагоні
	Ціна квитка
Розклад	ID
	Дата відправлення
	Час відправлення
	Дата прибуття

	Час прибуття
Потяги	ID
	Номер потяга
	Назва потяга
	Тип потяга
	Кількість вагонів
Вагони	ID
	Номер вагону
	Тип вагону
	Кількість місць у вагоні
	Статус вагона
Місця у вагоні	ID
	Номер вагону
	Номер місця
	Тип місця
	Статус місця
Станції	ID станції
	Назва станції
	Місто
	Регіон
	Назва залізниці
	Статус станції
Зупинка	ID
	Номер потяга
	Номер станції
	Час прибуття
	Час відправлення

Між цими сутностями є зв'язки, а саме:

Клієнт і Квиток - це 1:N (один до багатьох), оскільки один клієнт може купити багато квитків, однак кожен квиток купує лише один клієнт;

Квиток і Розклад - це N:1 (багато до одного), оскільки багато квитків належать до одного розкладу, однак в одному розкладі може бути багато квитків;

Розклад і Потяги - це N:1 (багато до одного), оскільки багато розкладів мають один потяг, однак один поїзд може мати багато розкладів;

Потяги і Вагони - це 1:N (один до багатьох), оскільки один потяг складається з багатьох вагонів, однак кожен вагон належить лише одному потягу;

Вагони і Місця у вагоні - це 1:N (один до багатьох), оскільки один вагон містить багато місць, але кожне місце належить лише одному вагону;

Місця у вагоні і Квиток - це 1:N (один до багатьох), оскільки одне місце у вагоні може мати у багатьох квитків, але кожен квиток має лише одне місце;

Квиток і Станції - це N:1 (багато до одного), оскільки багато квитків мають одну станцію відправлення, але одна станція може бути станцією відправлення для багатьох квитків;

Квиток і Станції - це також N:1 (багато до одного), оскільки багато квитків мають одну станцію прибуття, але одна станція може бути станцією прибуття для багатьох квитків;

Станції і Зупинка - це N:1 (багато до одного), оскільки багато зупинок є місцем для однієї станції, але одна станція може бути місцем для багатьох зупинок;

Зупинка і Потяги - це N:1 (багато до одного), оскільки багато зупинок мають один потяг, але один потяг може мати багато зупинок.

Отже, було розроблено концептуальну модель даних для залізничного вокзалу, який є етапом в проектуванні системи. Його основними теоретичними та практичними результатами є визначення головних сутностей, формування переліку атрибутів для кожної сутності, визначення зв'язки між сутностями. Це міцна основа для подальшого проектування бази даних для логічної та

фізичної моделі даних. Сформульовані сутності з атрибутами та зв'язками між ними забезпечать для майбутньої бази даних цілісність та інформаційне забезпечення системи.

2.2.2 Розробка логічної моделі даних

Після формування концептуальної моделі даних з своїми сутностями та взаємозв'язками, наступним етапом є розробка логічної моделі даних. Логічна модель даних перетворює концептуальні сутності та атрибути в таблиці для бази даних. Також, визначають первинні та зовнішні ключі, що буде гарантувати зв'язність та інформація буде цілою. Метою є створення архітектури бази даних, яка буде структурована та стане також основою для фізичної моделі даних. На рис.2.5 зображено логічна модель даних для залізничного вокзалу.

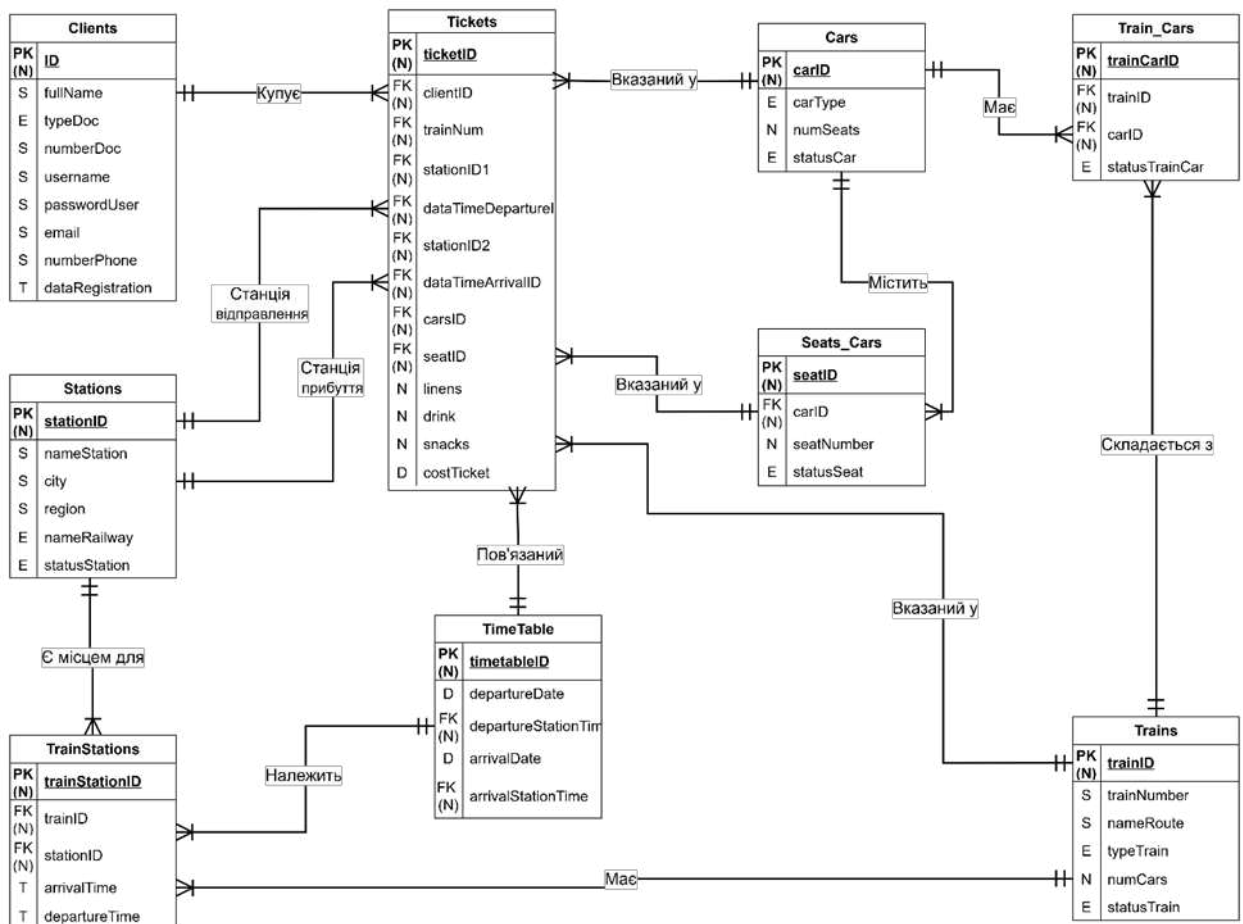


Рис.2.5 - Логічна модель даних для залізничного вокзалу

В таблиці Tickets є зовнішні зв'язки з Clients (зовнішній ключ clientID у таблиці Tickets посилається на первинний ключ ID у таблиці Clients), Stations (зовнішні ключі stationID1 та stationID2 у таблиці Tickets посилаються на первинний ключ stationID у таблиці Stations), Cars (зовнішній ключ carsID у таблиці Tickets посилається на первинний ключ carID у таблиці Cars), Seats_Cars (зовнішній ключ seatID у таблиці Tickets посилається на первинний ключ seatID у таблиці Seats_Cars), TimeTable (зовнішній ключ dateTimeDepartureID та dateTimeArrivalID у таблиці Tickets посилається на первинний ключ timetableID у таблиці Timetable), Trains (зовнішній ключ trainNum у таблиці Tickets посилається на первинний ключ trainID у таблиці Trains).

В таблиці TimeTable є зовнішні зв'язки з TrainStations (зовнішні ключі departureStationTimeID та arrivalStationTimeID у таблиці Timetable посилаються на первинний ключ trainStationID у таблиці TrainStations).

В таблиці Train_Cars є зовнішні зв'язки з Trains (зовнішній ключ trainID у таблиці Train_Cars посилається на первинний ключ trainID у таблиці Trains), Cars (зовнішній ключ carID у таблиці Train_Cars посилається на первинний ключ carID у таблиці Cars).

В таблиці TrainStation є зовнішні зв'язки з Trains (зовнішній ключ trainID у таблиці TrainStation посилається на первинний ключ trainID у таблиці Trains), Stations (зовнішній ключ stationID у таблиці TrainStation посилається на первинний ключ stationID у таблиці Stations).

В таблиці Seats_Cars є зовнішні зв'язки з Cars (зовнішній ключ carID у таблиці Seats_Cars посилається на первинний ключ carID у таблиці Cars).

Отже, було розроблено логічну модель даних, що є продовження концептуальній моделі. Основними результати є перетворення сутностей та атрибутів з концептуальної моделі у таблиці для логічної моделі даних, визначено первинні та зовнішні ключі для кожної таблиці та створено структуровану архітектуру для бази даних. Ця логічна модель буде основою для створення фізичної моделі даних та створенню бази даних.

2.2.3 Розробка фізичної моделі даних

Також після формування логічної моделі що представляє собою структурованим описом сутностей системи, наступним етапом є розробка фізичної моделі даних для залізничного вокзалу. Він передбачає практичну реалізацію логічної структури в форматі об'єктів бази даних, дивлячись на особливості обраної системи управління базами даних. Для створення фізичної моделі даних було обрано реляційну систему MySQL, яка представляє собою продуктивну, надійну та відкрита до рішень з питання побудови. Метою розробки є створення структури бази даних, яка задовільнить функціональним вимогам до системи, а також забезпечить продуктивність, масштабування та стабільність у роботі. На рис.2.6 зображено діаграма фізичної моделі даних для залізничного вокзалу.

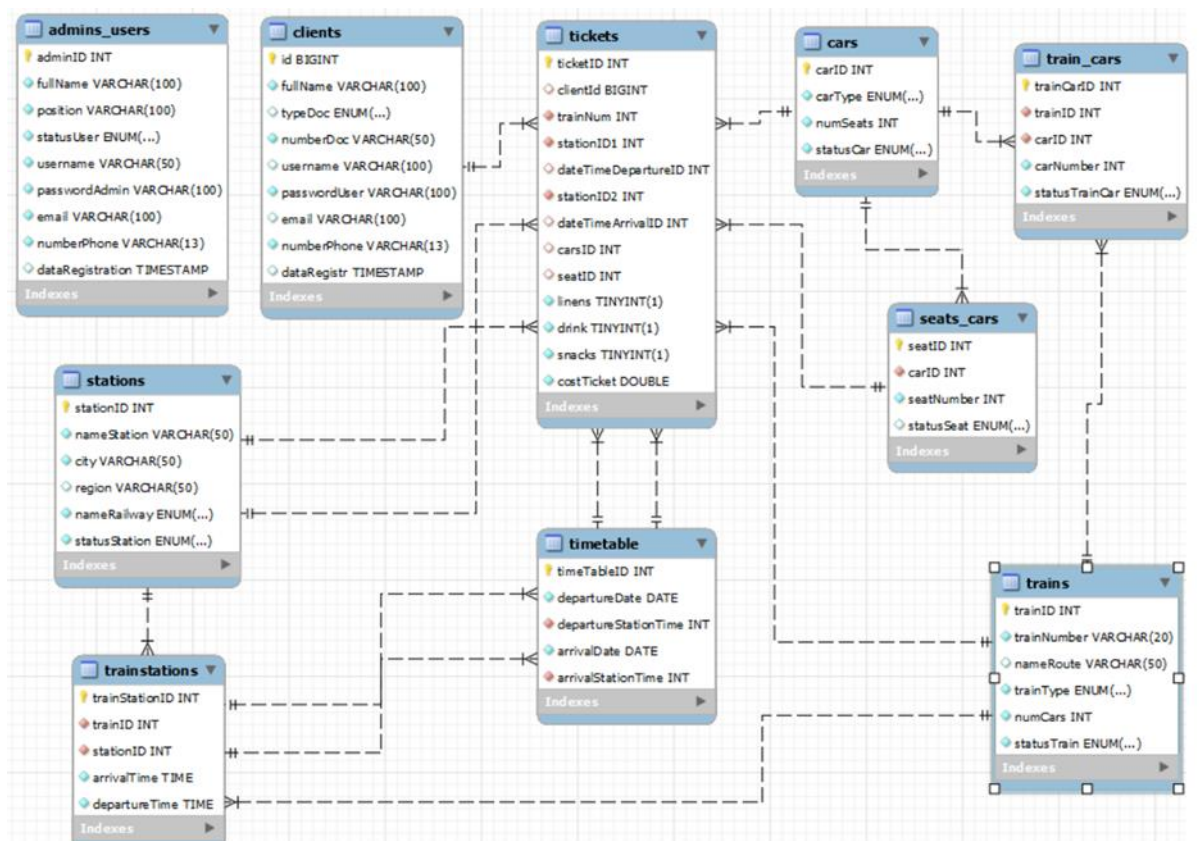


Рис.2.6-Фізична модель даних для залізничного вокзалу.

Ця фізична модель даних показує логічну структуру, яка виконує надання типів даних, їх розміри для кожного атрибуту та визначенні індекси. Взаємозв'язки між ними (первинні та зовнішні ключі) було визначено в

логічній моделі даних та збережені і реалізовані до синтаксису MySQL. Наприклад, в таблиці tickets, timetable, train_cars, trainstations та seats_cars знаходяться зовнішні ключі, які посилаються на первинні ключі відповідних батьківських таблиць, які забезпечують цілісність даних та можливість об'єднати інформацію. Приклад визначених типів даних включає використання BIGINT для унікальних ідентифікаторів, VARCHAR для текстових полів змінної довжини, DATETIME та TIMESTAMP для збереження інформації про дату та час, а також ENUM для полів з обмеженим набором можливих значень.

Отже, було розроблено фізична модель даних для залізничного вокзалу, як кінцевий етап проектування бази даних, що показує готовність для практичної реалізації логічної структури в системі управління бази даних. Основними теоретичними та практичним результатами є вибір СУБД MySQL, реалізація взаємозв'язки між таблицями, визначення індексів для доступу даних. Фізична модель даних відповідає функціональним вимогам та забезпечує продуктивність, масштабованість та стабільність у роботі web-застосунку для залізничного вокзалу.

Висновки до розділу 2

В цьому розділі було здійснено етап проектування архітектури інформаційні системи для web-застосунку залізничного вокзалу. В загальносистемних рішень було візуалізовано функціональні вимоги системи, а саме діаграма варіантів використання, яка дозволяє описати взаємодію акторів (Клієнт, Адміністратор, Зовнішня платіжна система) з системою. Та діаграми послідовності для розуміння хронологічного потоку подій між компонентами системи для розуміння логіка роботи та взаємодії модулів. У рішенні по інформаційному забезпеченню було спроектовано систему зберігання та обробки даних та використаний підхід до розробки бази даних якій забезпечить послідовність у визначених структури даних. Концептуальна

модель даних дозволила визначити основні сутності та їх логічної взаємозв'язки для розуміння процесу. Логічна модель даних перетворила сутності концептуальної моделі у таблиці з первинними та зовнішніми ключами, яка забезпечить цілісність даних та підготує основу для управління даними. А фізична модель є завершальним проектування бази даних для створення моделі було обрано СУБД MySQL. Також деталізовано таблиці з індексами, типами даних та обмеженнями до обраної системи, яка забезпечить бази даних до реалізації та буде гарантувати продуктивність, надійність та масштабованість. Отримані ці результати є основою для реалізації інформаційні системи, а саме реалізації діаграм та моделі даних яка буде використана як технічне завдання для розробників. А його практичність має зменшити ризики на етапі кодування, тому що всі рішення структури та взаємодії вже будуть прийняті та дозволять розподілити завдання розробникам та забезпечать розуміння архітектури системи. Результат цієї розробки буде використано для подальшого запровадження та розробки web-застосунку для залізничного вокзалу.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОЕКТУ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Після аналізу та створення структури бази даних для web-застосунку для залізничного вокзалу, потрібно практично реалізувати web-застосунок. Для цього потрібно розглянути програмне та технічне забезпечення для системи та визначити для програмного забезпечення архітектуру яка буде забезпечувати функціональні обов'язки між компонентами. Для визначення архітектури програмного забезпечення можна візуалізувати через діаграму розгортання. На рис.3.1 зображено діаграма розгортання для залізничного вокзалу.

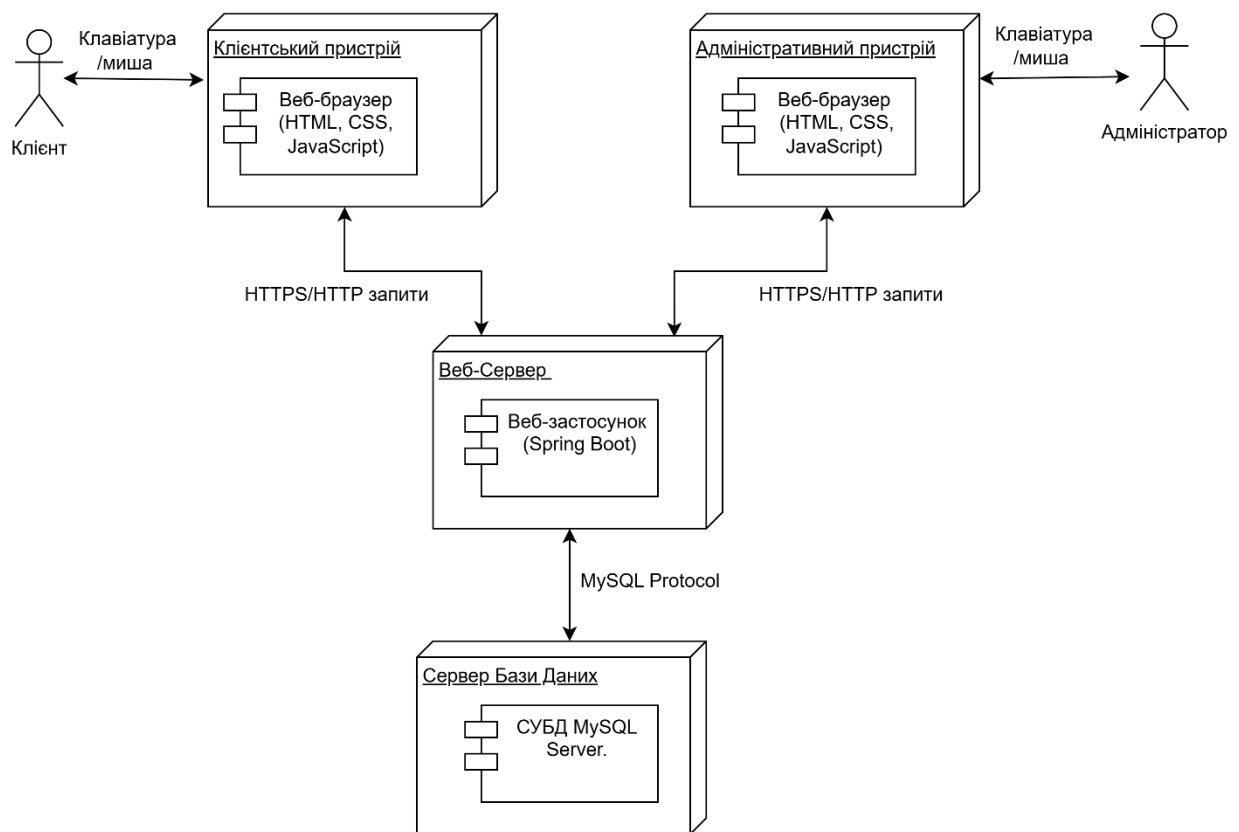


Рис.3.1-Діаграма розгортання для залізничного вокзалу

Діаграма розгортання показує розгортання програмних компонентів на апарати. Складається з основних логічних вузлів (клієнтська частина, адміністративна частина, сервер застосунку, сервер бази даних), які забезпечують розподіл відповідальності, безпеки та масштабованості.

Клієнтський пристрій є апаратним пристроєм який використовує кінцевий користувач, в ньому є веб-браузер для перегляду веб-сторінок, які

розроблені з використанням HTML, CSS, JavaScript. Через інтерфейс можна переглядати розклад поїздів, шукати поїзди та купляти квитки, отримувати повідомлення про затримку, доступ до особистого кабінету.

Адміністративний пристрій є апаратним пристроєм який використовує кінцевий користувач, призначений для персоналу залізничного вокзалу, які відповідають за управління системою та даними. Також використовує веб-браузер для перегляду веб-сторінок, які розроблені з використанням HTML, CSS, JavaScript для доступу адміністративної панелі системи. Через панель можна управляти даними сутностей в базі даних, формувати звіти, редагувати ціни.

Веб-сервер це серверний вузол який відповідає за обробку системи та взаємодії з базою даних. Використовує Веб-застосунок який є основною системою, який обробляє запити від клієнта та адміністратора та курує логікою управління даними, взаємодією бази даних для збереження та отримання даних й обробкою платежів.

Сервер Баз Даних є серверним вузлом для управління та зберігання даних. Використовує систему управління базами даних MySQL Server, яка дає надійне зберігання, цілісність даних та швидкий доступ.

Веб-браузер на пристроях клієнта та адміністратора встановлює зв'язок з Веб-сервером, який відправляє HTTP/HTTPS запити для отримання даних та використовує функції застосунка.

В свою чергу Веб-застосунок який відкритий на Веб-сервері встановлює зв'язок з c\Сервером Баз Даних для використання операцій SQL (SELECT, INSERT, UPDATE, DELETE) для збереження та отримання даними, які пов'язані з залізничним вокзалом.

Програмне забезпечення web-застосунка для залізничного вокзалу буде створена на основі багаторівневої архітектури, яка буде забезпечувати гнучкість та масштабованість. Це дозволяє ефективно розподілити функціональні обов'язки між компонентами, надавати доступ до даних, розділити інтерфейс від бізнес-логіки. Ця структура надає розуміння

архітектурного рішення, функції модулів та їх взаємодії, та надає вибір оптимальних програмних інструментів для реалізації системи. Для сучасних застосунків характерно багаторівнева побудова, яка забезпечує розподіл обов'язків, обслуговування системи та модульність. В проєкті яка буде розроблена на Spring Boot, використовується логічні рівні які структуровані у окремих папок. Презентаційний рівень який відповідає за обробку HTTP-запитів, взаємодіє з інтерфейсом для користувачів (Клієнт, Адміністратор) та формує відповіді. Контролери, які будуть отримувати дані від клієнта або адміністратора передають на наступні етапи для обробки та формування відповідей в форматі HTML-сторінок. Для цього використовується HTML, CSS, JavaScript який створює інтерактивний та адаптований дизайн для web-застосунку. Сервіси будуть реалізовувати бізнес-логіку застосунка. Вони обробляють запити від контролерів, проводять перевірки, координують взаємозв'язки з базами даних та іншими службами і виконують розрахунки. Вони забезпечать послідовність та цілісність бізнес-процесів. Для доступу до даних використовується Spring Data JPA, яка реалізує відображення даних та спрощує операції CRUD (створення, читання, оновлення, видалення) для сутностей. Моделі та сутності знаходяться в класах Java, які будуть відображати структуру таблиці бази даних та слугуватимуть для передачі даними між системою. В конфігурації включають класи з параметрами системи та налаштуванням безпеки. Для управління залежностями між компонентами використовують механізм контролю і впровадження залежності є елементами фреймворку Spring. Модулі автентифікації та авторизації, які відповідають за захищений доступ до системи. Вони мають можливості реєстрації користувачів, зокрема безпечне хешування паролів та зручність входу. Також система розподіляє права доступу до різних функцій для користувачів (Клієнт, Адміністратор) завдяки Spring Security. Модуль управління даними дає адміністраторам легко керувати інформацією у базі даних. Клієнтам надають можливість зручно шукати рейси за напрямком, датою та часом. Функціонал покупки та управління квитками забезпечує весь процес (від пошуку місць до

оформлення та купівлі квитка). Історія квитків доступна для зареєстрованих клієнтів. Користувацький інтерфейс створено для зручності та інтуїтивності. Дизайн на базі CSS дає правильне відображення на всіх типах пристроїв. Завдяки використанню JavaScript на клієнтській стороні, система підвищує інтерактивність, підтримує асинхронні запити та оновлює інформацію без перезавантаження сторінки. Для розробки та підтримки програмного забезпечення використовується ключові інструменти, які ефективно та якісно виконують роботу. Maven забезпечував автоматизацію збирання проєкту, управління залежностями і пакування виконуваних файлів. IntelliJ IDEA надає розширені можливості для написання коду, налагодження та тестування, що оптимізовано для Java та Spring Boot. Система контролю версій Git забезпечила управління вихідним кодом, відстеження змін і підтримала колективну роботу над проєктом.

Для розгортання та забезпечення стабільної роботи web-застосунку для залізничного вокзалу необхідно чітко визначити апаратні та програмні вимоги, які відповідають потребам системи. Ці вимоги мають забезпечити продуктивність, надійність та доступність програми для користувачів. Технічні рішення розробляються з урахуванням архітектури, функціональності системи та прогнозованого навантаження.

Серверне обладнання дає ключову роль, оскільки на ньому базується серверна частина програми та система керування базами даних (СУБД). Для підтримки стабільної роботи з можливістю масштабування встановлено такі мінімальні та рекомендовані вимоги таких як:

Процесор (8-ядерний, здатний ефективно обробляти великий об'єм одночасних запитів і виконання складних операцій);

Оперативна пам'ять (від 16 ГБ для покращеного кешування даних, стабільної роботи за підвищеного навантаження і швидшого виконання операцій із базою даних);

Дисковий простір (SSD-накопичувач обсягом 100 ГБ для високошвидкісного читання і запису даних, що критично важливо для продуктивності бази даних та оперативного завантаження веб-додатка);

Мережевий інтерфейс (Ethernet з пропускною здатністю 1 Гбіт/с, що гарантує швидкий і стабільний обмін даними між сервером і клієнтами);

Інтернет-з'єднання (широкопasmуговий доступ із мінімальною пропускною здатністю 100 Мбіт/с і високою стабільністю).

Для належного функціонування веб-програми також потрібно відповідне програмне забезпечення такі як:

Операційна система (Microsoft Windows Server 2019/2022, що забезпечує стабільність, високу продуктивність серверних додатків, безпеку та широкі можливості підтримки);

Середовище виконання (Java OpenJDK 17 для запуску програми, побудованої з використанням Spring Boot);

СУБД (MySQL, яка використовується для зберігання даних програми);

Spring Boot має вбудований сервер Tomcat, тому немає потреби встановлювати його окремо.

Для промислових середовищ використовується проксі-сервер для таких задач, як: балансування навантаження, кешування статичних файлів, шифрування трафіку (SSL), а також загальна оптимізація безпеки програми. Система контролю версій (Git для управління кодом і автоматизації розгортання).

Клієнтська частина веб-додатку функціонує у веб-браузері користувача, що дозволяє знизити вимоги до обладнання. Для роботи підходить будь-який комп'ютер, ноутбук, планшет або смартфон із доступом до Інтернету. Для комфортного використання сучасного браузера необхідно щонайменше 4 Гб оперативної пам'яті, а також достатньо вільного місця на диску для зберігання кешу браузера. Програмне забезпечення має включати актуальну версію сучасного браузера, який підтримує HTML, CSS і має ввімкнений JavaScript,

необхідний для коректного впливу інтерфейсу та його інтерактивності. Крім того, важливе значення має стабільний доступ до мережі Інтернет.

Висновки до розділу 3

Отже, було здійснено технічну реалізацію web-застосунку для залізничного вокзалу, якою є частиною дослідження роботи та досягнуті результати які відображають реалізацію поставленої мети. Основними практичними та теоретичними результатами є побудова архітектури системи для оптимального рішення та розподілу відповідальності. Діаграма розгортання демонструє розподіл програмних компонентів між апаратними вузлами. Було визначено вимоги до програмного забезпечення, а саме визначення технологій та середовище розробки як IntelliJ IDEA та контролю версій. Було визначені вимоги до технічного забезпечення, а саме сформульовано апаратні та програмні вимоги для серверної та клієнтської частини. Визначенні ці вимоги будуть слугувати готовим керівництвом для підготовки інфраструктури. Подальші вивчення теми можуть включати розширення функціоналу, наприклад реалізація мобільного додатку. Та перспектива є хмарних збережень для підвищення масштабності та доступності системи.

РОЗДІЛ 4. ОХОРОНА ПРАЦІ

У світі, де розробка програмного забезпечення є однією з сфер, питання охорони праці набуває особливого значення. Хоча робота за комп'ютером може здатися «безпечною», вона несе ризики для здоров'я, такі як напруга очей, проблеми з опорно-руховим апаратом та перевантаження нервів. Забезпечення безпечних умов праці має значення для продуктивності та добробуту розробників. Оскільки вокзали є місцями, де збираються люди, питання безпеки є особливо важливими. Мета це висвітлити основні аспекти охорони праці для розробників, правила поведінки в надзвичайних ситуаціях та принципи екологічної безпеки, актуальні для IT-галузі.

Сама розробка програмного забезпечення це професійна діяльність із конкретними ризиками для здоров'я та благополуччя фахівців. Не звертаючи уваги на організацію робочих місць та графіків роботи, це може призвести до розвитку професійних захворювань, що в свою чергу може зменшити продуктивність та загальний дискомфорт. Щоб забезпечити велику ефективність та підтримувати здоров'я розробника web-застосунку для залізничного вокзалу, необхідно врахувати та мінімізувати основні ризики.

Наприклад, навантаження на зір, при довгої та інтенсивної роботи з візуальною інформацією на екрані, що спричиняє значне навантаження на око. Це може спричинити зорову втому, що проявляється в відчутті печіння, різі в очах, головних болях, а також призводить до поступового погіршення гостроти зору, розвитку або прогресування короткозорості та синдрому сухого ока.

Сидячий спосіб життя та тривалий час у постійній позі, коли погана організоване робоче місце є головним фактором ризику опорно-рухової системи. Це може призвести до розвитку остеохондрозу, сколіозу, болю в шії, спині, попереку, а також синдрому зап'ястного каналу, що впливає на нерви руки через безперервне напруження та погане положення зап'ястя при роботі з клавіатурою та мишею.

Висока інтенсивність розумової роботи, безперервна концентрація, стрес від термінах, складних завдань та монотонність операцій можуть

спричинити хронічну перевтому, неврологічні розлади, безсоння, дискомфорт, зменшення здатності до фокусування та емоційне вигорання.

Робота з комп'ютерною технікою та іншими для роботи обладнанням завжди пов'язана з ризиком ураження електричним струмом. Причинами можуть бути несправність ізоляції дроту, використання пошкодженого обладнання, перевантаження електромережі, відсутність заземлення або порушення правил безпеки при підключенні або відключенні обладнання. Це може спричинити електротравму різної тяжкості та в гіршому випадку - до смерті.

Параметри без внутрішніх мікрокліматних значень (температура, вологість) можуть впливати на здоров'я, комфорт і продуктивність. Температура занадто висока або при низьких температурах (особливо в кондиціонерах або нагрівальному обладнанні) може спричинити дискомфорт, головний біль, сухість слизових оболонок та зменшення концентрації.

Для мінімізації ризиків та забезпечення оптимальних умов праці під час розробки web-застосунку для залізничного вокзалу необхідно дотримуватися рекомендацій і норм охорони праці. Зокрема, ергономічна організація робочого місця відіграє вирішальну роль у запобіганні професійним захворюванням та підвищенні комфорту співробітників.

Робоче місце потребує якісного освітлення, яке має бути рівномірним, достатнім і без відблисків. Найкраще використовувати комбіноване освітлення: загальне верхнє світло для всього приміщення разом із додатковим місцевим освітленням, наприклад настільною лампою з можливістю регулювання яскравості та напрямку світла. Особливо важливо, щоб настільна лампа не створювала відблисків на дисплеї монітора. Природне світло повинно проникати збоку, бажано зліва для праворуких осіб, аби уникнути тіней на робочій поверхні. Регулювання інтенсивності сонячного світла можна здійснювати за допомогою штор або жалюзі, що також допоможе зменшити відблиски.

Правильне розташування монітора є надзвичайно важливим для збереження зору та підтримки правильної постави. Оптимальна відстань від очей до екрана повинна складати 60-70 см, що забезпечує комфортне фокусування. Верхня частина екрана має бути на рівні очей або трохи нижче, що сприяє природному положенню голови та шиї, зменшуючи їх напруження. За потреби використовуйте підставки для монітора. Екран слід розташовувати перпендикулярно до вікна, аби уникнути прямих сонячних променів або інтенсивного освітлення. Яскравість і контрастність монітора потрібно регулювати залежно від умов освітлення в приміщенні. Важливо, щоб екран був чистим від пилу та відбитків пальців, також рекомендується використовувати матове покриття.

Вибір та налаштування офісного крісла є однією з ключових ергономічних вимог. Варто обирати ергономічне офісне крісло з можливістю регулювання висоти сидіння, кута нахилу спинки та положення підлокітників. Спинка крісла повинна підтримувати поперековий відділ хребта, відповідно до його природного вигину. Сидіння має бути достатньо глибоким для підтримки стегон, але не надмірно стискати кровоносні судини під колінами.

Ноги мають бути поставлені на підлогу або на спеціальну підставку під кутом 90 градусів у колінах. Лікоть має утворювати кут 90-100 градусів при роботі з клавіатурою. Правильне розташування миші та клавіатури допомагає уникнути перенапруження зап'ясть та рук. Клавіатура повинна бути розміщена на відстані 10-15 см від краю столу, щоб залишалось місце для підтримки передпліччя. Руки мають знаходитися у природному положенні, без зайвого згинання зап'ясть вгору чи вниз. Доцільно використовувати ергономічні килимки для миші з підтримкою під зап'ястя, що допомагає утримувати їх у нейтральній позиції.

Рациональне дотримання режиму праці та відпочинку відіграє ключову роль у збереженні здоров'я, зокрема зору, фізичного стану та психоемоційного благополуччя. Під час роботи за комп'ютером варто робити короткі активні перерви кожні 45–60 хвилин тривалістю 5–10 хвилин. Протягом дня

обов'язковою є одна тривала обідня перерва, що триває 30–60 хвилин. Під час таких перерв доцільно змінювати вид діяльності, виконувати фізичні вправи або тимчасово залишати робоче місце. У короткі проміжки відпочинку корисно виконувати спеціальні вправи для очей, як наприклад фокусування погляду на предметах різної віддалі, кругові рухи очима, часте моргання чи коротке міцне змикання повік. Такі заняття сприяють зняттю напруження з очних м'язів та зволоженню слизової оболонки. Регулярні розминки тіла під час роботи запобігають застою крові та допомагають зняти м'язову напругу. До них можна віднести обертання головою, нахили корпусу, вправи для плечей, рук, зап'ясть або ніг. Важливо також займатися фізичною активністю поза межами робочого часу: прогулянками чи будь-яким видом спорту, що сприяє загальному зміцненню організму.

Психоемоційний стан має прямий вплив як на якість виконуваних завдань, так і на загальне самопочуття. Важливо планувати свій робочий день і розділяти складні проєкти на невеликі етапи, що допомагає уникнути відчуття перевантаження та підтримує мотивацію. Продуманий графік роботи зменшує рівень стресу й підвищує продуктивність. Повноцінний нічний сон, тривалістю 7–8 годин, є необхідним для відновлення нервової системи. Не слід забувати про важливість позаробочого відпочинку: захоплені хобі, час із сім'єю чи друзями допомагають тимчасово абстрагуватися від робочих завдань і поновити психоемоційні ресурси.

Дотримання елементарних правил електробезпеки є важливим чинником для запобігання травмам та надзвичайним ситуаціям. Слід користуватися лише справними електроприладами з непошкодженою ізоляцією та корпусом. Перед експлуатацією нового обладнання необхідно ознайомитися з інструкцією виробника. Варто уникати одночасного підключення великої кількості пристроїв до однієї розетки або подовжувача, аби не створювати ризику перевантаження електромережі, яке може спричинити коротке замикання чи пожежу. Рекомендується застосовувати мережеві фільтри із захистом від стрибків напруги. Увімкнені прилади не можна залишати без нагляду,

особливо ті, що не використовуються тривалий час. Після завершення роботи слід повністю вимикати комп'ютер і периферійне обладнання з електромережі.

У разі виявлення будь-яких несправностей, таких як іскріння, запах гарі, незвичні шуми чи пошкодження проводів, слід негайно від'єднати пристрій від електромережі, обережно вийнявши вилку з розетки, не торкаючись пошкоджених ділянок. Після цього необхідно повідомити про проблему відповідальній особі за електроустановки або адміністрації. Самовільний ремонт електрообладнання заборонено.

Оптимальний мікроклімат у приміщенні є ключовою умовою для забезпечення зручності та ефективності роботи. Важливо регулярно провітрювати приміщення, особливо якщо воно обладнане великою кількістю комп'ютерної техніки, яка генерує тепло. Провітрювання дозволяє відновити рівень кисню, зменшити концентрацію шкідливих речовин у повітрі та підтримувати необхідну вологість. Для створення комфортних умов слід підтримувати температуру повітря в діапазоні 20-24°C та відносну вологість на рівні 40-60%. Сухе повітря може викликати дискомфорт, зокрема подразнення слизових оболонок очей і шкіри. У разі потреби слід застосовувати зволожувачі повітря. Застосування цих комплексних заходів з охорони праці не лише сприятиме зменшенню ризиків для здоров'я розробників веб-застосунків залізничного вокзалу, а й допоможе створити безпечне та сприятливе робоче середовище.

Незважаючи на те, що процес розробки програмного забезпечення зазвичай проходить в офісному середовищі і не супроводжується підвищеним ризиком надзвичайних ситуацій, знання та вміння правильно діяти у небезпечних умовах надзвичайно важливі. Це стосується і особистої безпеки самого розробника, і забезпечення безпеки кінцевих користувачів створеного web-застосунку. Особливу актуальність це набуває тоді, коли впровадження програмного забезпечення відбувається у таких місцях, як залізничні вокзали. Зона значного скупчення людей, які є потенційно вразливими до різноманітних надзвичайних ситуацій. Знання алгоритмів дій у кризових умовах допомагає

зберегти життя і здоров'я, уникати паніки, оперативно реагувати на загрози, а також формує почуття захищеності та сприяє підвищенню загальної культури безпеки.

Серед найпоширеніших та найбільш небезпечних надзвичайних ситуацій особливе місце займає пожежа. Дотримання правил протипожежної безпеки та розуміння алгоритму дій є життєво необхідними. Кожен працівник повинен заздалегідь ознайомитися з планом евакуації, який має бути розташований на видному і доступному місці. Треба знати місцезнаходження найближчих пожежних виходів, первинних засобів пожежогасіння (таких як вогнегасники, пожежні крани або щити), а також мати базові навички їх використання. Проведення регулярних інструктажів і навчальних тренувань з пожежної безпеки є обов'язковим заходом у рамках забезпечення захисту. При виявленні навіть незначного загоряння необхідно негайно сповістити пожежну службу за телефоном 101, указати точну адресу об'єкта, місце займання та своє ім'я. Одночасно слід повідомити про інцидент адміністрацію або відповідального за протипожежну безпеку. Якщо пожежа має незначні масштаби й не загрожує життю, можна спробувати загасити її самостійно за допомогою вогнегасника, піску, води чи щільної тканини. Водночас слід пам'ятати: застосування води для гасіння електроприладів під напругою суворо заборонено. У разі значної загрози життю, сильного задимлення чи швидкого поширення вогню необхідно негайно покинути приміщення відповідно до плану евакуації. Під час евакуації потрібно допомагати тим, хто потребує допомоги: дітям, літнім людям та особам з обмеженими можливостями. Користуватися ліфтами категорично заборонено, оскільки вони можуть зупинитися через відключення електропостачання або задимлення шахти. Для евакуації слід використовувати лише сходи.

Вибух, спричинений підозрілими об'єктами, належить до найруйнівніших та непередбачуваних надзвичайних ситуацій, які можуть призвести до значних людських втрат і матеріальних пошкоджень. У разі виявлення безхазяйних предметів, таких як сумки, пакети, коробки чи будь-що

інше, що викликає підозру, необхідно негайно повідомити правоохоронні органи за номером 102 та адміністрацію відповідного об'єкта. Недопустимо торкатися, переміщати чи відкривати такі предмети. Потрібно попередити людей поруч і відійти на безпечну відстань. У випадку реальної загрози вибуху або його здійснення слід залишити небезпечну зону якомога швидше. Якщо неможливо покинути приміщення миттєво, варто знайти міцне укриття за несучими стінами, колонами чи заглибленнями. Рекомендується лягти на підлогу, захистивши голову руками, та віддалитися від вікон чи скляних поверхонь, які можуть бути розбиті ударною хвилею.

Серед техногенних катастроф варто виділити аварії на промислових об'єктах, які супроводжуються викидами хімічно небезпечних речовин або радіації. У таких випадках важливо стежити за офіційними повідомленнями органів влади, Державної служби України з надзвичайних ситуацій та місцевих ЗМІ, які надають перевірену інформацію та інструкції щодо подальших дій. Необхідно строго виконувати рекомендації рятувальних і правоохоронних служб, які можуть вимагати евакуації, укриття чи використання засобів індивідуального захисту. При оголошенні тривоги потрібно дотримуватися попередньо розробленого плану цивільного захисту, прямуючи до спеціально облаштованих укриттів або евакуюючись із небезпечної зони.

На тлі теперішнього воєнного часу питання безпеки громадян під час бойових дій є вкрай важливим. Кожен має знати правила поведінки у разі обстрілів, бомбардувань чи інших загроз. При оголошенні сигналу "Повітряна тривога" необхідно якнайшвидше прямувати до найближчого укриття: бомбосховища, підвалу чи підземного паркінгу. Важливо попередньо ознайомитися з розташуванням таких місць поруч із домом або роботою. А також слід завжди мати підготовлену "тривожну валізу" з найважливішими речами: документи, готівку, запас води та їжі на кілька днів, аптечку з індивідуальними ліками, ліхтарик, павербанком для заряджання мобільних пристроїв, теплий одяг. Якщо немає можливості дістатися укриття, слід залишатися вдома й знаходитися у приміщенні, яке відокремлене від

зовнішнього простору двома стінами (наприклад, коридор чи ванна кімната), що забезпечує додатковий захист від уламків і ударної хвилі. Варто також слухати рекомендації ДСНС, військових адміністрацій та ЗСУ. Надзвичайно важливо уникати поширення неперевіреної інформації.

Розробка web-застосунку для залізничного вокзалу, попри свою ІТ-специфіку, має важливе відношення до безпеки пасажирів і персоналу в надзвичайних ситуаціях. Цей програмний продукт може стати ефективним засобом швидкого інформування та координації дій. Він може бути інтегрований з базами даних вокзалу для оперативного відображення змін у розкладі поїздів, інформації про затримки, скасування та зміни платформ. У критичних ситуаціях такий застосунок здатен швидко повідомляти пасажирів про необхідність евакуації, спрямування до безпечних місць або зміну маршрутів, що є життєво важливим для управління великими потоками людей і уникнення паніки. Крім того, застосунок може включати функцію швидкого зворотного зв'язку з відповідними службами вокзалу у випадку надзвичайних ситуацій або потреби у допомозі, наприклад, через чат з диспетчером. В перспективі, web-застосунок можна інтегрувати із загальнонаціональними чи місцевими системами оповіщення про надзвичайні ситуації, щоб автоматично розсилати відповідні повідомлення користувачам або відображати їх в інтерфейсі для надання актуальної і централізованої інформації про загрози та рекомендації щодо дій. Належна, своєчасна та точна інформація через застосунок знижує рівень паніки серед пасажирів, дозволяючи їм приймати обґрунтовані рішення і діяти відповідно до інструкцій.

Хоча розробка програмного забезпечення має мінімальний прямий негативний вплив на навколишнє середовище порівняно з промисловими виробництвами, ІТ-сфера не є повністю екологічно нейтральною. Існують деякі опосередковані впливи, значні в масштабах усієї індустрії, які вимагають відповідального підходу. Розуміння цих аспектів і впровадження відповідних практик є важливим елементом сучасної розробки, особливо для систем, такі

як web-застосунок для залізничного вокзалу, працюють постійно і споживають значні обсяги ресурсів.

Функціонування комп'ютерів, серверів, мережевого обладнання та систем охолодження в центрах обробки даних є одним із найзначніших факторів споживання електроенергії у сфері ІТ. Постійне зростання обсягу даних і обчислювальних потужностей призводить до збільшення енергоспоживання, що неминуче впливає на зростання вуглецевого сліду, особливо якщо електроенергія отримується шляхом спалювання викопного палива. При виборі технічного обладнання для розробки (комп'ютери, монітори, периферія) та забезпечення роботи застосунків (сервери) варто надавати перевагу моделям із високим класом енергоефективності. Сучасне обладнання проєктується таким чином, щоб виконувати свої завдання за мінімального споживання ресурсів. Наприклад, менш енергоємні процесори, SSD-накопичувачі як заміна HDD-дисків, енергоощадні монітори це дозволяє суттєво зменшити витрати енергії. Дотримання простих рекомендацій допоможе скоротити як індивідуальний, так і загальний "енергетичний слід". Серед основних правил це вимикати комп'ютер, монітор та інше обладнання після завершення роботи або на час тривалих перерв. Режими сну чи очікування також споживають електроенергію, хоч і в менших обсягах. Від'єднувати зарядні пристрої та адаптери живлення від мережі, коли вони не використовуються, оскільки навіть у бездіяльному стані можуть споживати енергію. Оптимізувати налаштування операційних систем та програмного забезпечення для використання енергозберігаючого режиму. При впровадженні веб-застосунку, наприклад, у межах залізничного вокзалу з уже існуючою складною енергосистемою, важливо оцінювати його вплив на додаткове навантаження в мережі. Оптимізація енергоспоживання як серверної частини застосунку, так і клієнтського боку сприятиме покращенню загальної енергоефективності всього об'єкта.

Електронні відходи належать до найшвидше зростаючих видів сміття у світі. Комп'ютерна техніка містить цілий спектр небезпечних речовин,

включаючи свинець, ртуть, кадмій, хром, бромовані антипірени та інші шкідливі компоненти. Неправильна утилізація таких відходів спричиняє забруднення ґрунту, води та повітря, створюючи серйозний ризик для здоров'я людей і екосистем. Електронне обладнання категорично не можна викидати разом із побутовим сміттям. Застарілу чи несправну техніку слід передавати до спеціалізованих пунктів прийому електронних відходів. Існують компанії, які мають ліцензію на переробку E-waste та забезпечують безпечне вилучення цінних матеріалів і нейтралізацію токсичних речовин. Своєчасне обслуговування, ремонт і модернізація обладнання допомагають максимально продовжити його термін служби. Це дозволяє зменшувати потребу у виробництві нових пристроїв і знижувати обсяг відходів. Робоче, але морально застаріле обладнання доцільно передавати тим, хто може його ефективно використовувати, наприклад освітнім установам або благодійним організаціям. Цифровий слід це не лише дані, які ми залишаємо в інтернеті, а й екологічний вплив зберігання, обробки та передачі цих даних. Сервери, що займаються зберіганням великих масивів інформації, потребують постійного живлення та інтенсивного охолодження, який характеризується високим енергоспоживанням. Використання ефективного, оптимізованого програмного коду, здатного виконувати завдання з мінімальною кількістю обчислювальних операцій, зменшує навантаження на процесори та оперативну пам'ять. Завдяки цьому скорочується споживання енергії серверів, що позитивно впливає як на екологію, так і на ресурсну ефективність.

Розробляючи web-застосунок для залізничного вокзалу, необхідно спрямувати зусилля на досягнення таких цілей як оптимізація роботи з базою даних, яка передбачає підвищення ефективності SQL-запитів, впровадження індексів, використання кешування задля зниження навантаження на систему зберігання даних. Зменшення обчислювального навантаження на сервер (частину обчислень варто перенести на клієнтську сторону за допомогою JavaScript, що дозволить не лише покращити продуктивність серверної частини, але й підвищити загальну швидкість роботи застосунку). Скорочення

обсягу переданих даних для використання методів стиснення даних, оптимізація графічних файлів, а також мініфікація CSS та JavaScript дозволяють знизити обсяги трафіку та завдяки цьому можна зменшити енергоспоживання, що витрачається на передання цих даних. Управління даними для збереження лише необхідної інформації та регулярне очищення від застарілих або непотрібних даних сприяють економному використанню ресурсів сховищ. Це також охоплює продуманий підхід до ведення логів і створення резервних копій. Екологічна відповідальність при розгортанні застосунку варто надавати перевагу хостинг-провайдерам і хмарним платформам, які впроваджують екологічно сталий підхід. Сучасні центри обробки даних дедалі частіше використовують відновлювану енергію, зокрема сонячні панелі чи вітрові турбіни, а також інноваційні системи охолодження наприклад, рідинне охолодження, що допомагає мінімізувати вуглецевий слід.

Висновки до розділу 4

Отже, робимо висновок що охорона праці та безпека життя є невід'ємною частиною процесу розробки та функціонування програмного забезпечення. Метою полягає в формуванні підходу до забезпечення безпечних умов праці розробників та безпеки клієнтів в умовах підвищеної небезпеки на залізничному вокзалі. Основними теоретичними та практичними результатами є знаходження підходу до безпеки на всіх етапах життєвого циклу програмного забезпечення. Також, визначення ергономічних вимог та умов праці для розробника, а саме важливість дотримання балансу між працею та відпочинком, обізнаність електробезпеки та ергономічні умови роботи. Ще проведено аналіз правил поведінки в критичних ситуаціях, де було зосереджено увагу на правила поведінки під час надзвичайних подій. Та розглянули акцент на екологію в ІТ-сфері, а саме відповідальність проекту перед навколишнім середовищем. Для подальшого дослідження може бути

сфокусовано на розробці систем моніторингу для умов праці розробника та створенню тренажерів для відпрацюванню дій у надзвичайних ситуаціях.

ВИСНОВКИ

В кваліфікаційній роботі було успішно вирішено завдання розробка web-застосунка для залізничного вокзалу, яка має мету оптимізацію інформаційного процесу та покращення взаємозв'язок клієнтів з сервісами залізничного вокзалу. При виконання роботи були досягнуті теоретичні та практичні результати, які реалізують поставлені мети та завдань. Мета кваліфікаційної роботи була створення ефективної та функціональної web-застосунка для залізничного вокзалу, що включає в собі аналіз предметної області, проектування та реалізація проекту що була досягнута. Дослідивши всі об'єкти дослідження можна зробити до кожного висновок. Був проведено опис функціонування залізничного вокзалу як складної системи з інформаційним потоком та взаємодії клієнтів з залізничними службами. Також, було проведено аналіз предметної області, та було виявлено недоліки існуючої системи таких як обмеженість каналів доступу, що підкреслює актуальність та необхідність розробки рішення. Проаналізовано існуючі підходи та рішення для інформування клієнтів, які дозволять уникнути помилок в системі. Також, на основі проведеного аналізу було сформульовано функціональні вимоги до web-застосунку, які забезпечать потреби клієнтів та будуть відповідають стандартам. Розроблено концепцію інформаційної системи, що показує загальне бачення, архітектурні рішення та основні принципи функціонування та сформульовано постановку задачі, яка є основою для технічного проектування. Було розроблено діаграму варіантів використання, яка визначає основних акторів системи та описує взаємозв'язок. Та було розроблено діаграми послідовності для кожного актора, вона детально показує потік повідомлень акторами та основними компонентами системи на різних етапах виконання операцій. Також, було розроблено структуру інформаційного забезпечення для web-застосунку та моделі даних для створення структурної бази даних, а саме створення концептуальної моделі даних, що відображає основні сутності і атрибути та зв'язки між ними, що забезпечать розуміння предметної області. В свою чергу на основі концептуальної моделі було

розроблено логічну модель даних, які включають таблиці для бази даних з типами даних та первинними і зовнішніми ключами. Та розроблено фізична модель даних, що надає структуру бази даних для зберігання та доступу до даних. Також, здійснено технічну реалізацію що є практичною частиною дослідження. Розроблена діаграма розгортання для розподілу програмних компонентів на апаратних вузлах, яка демонструє структуру системи. Сформульовано програмне забезпечення для створення web-застосунку, для було обрано такі параметри як HTML, CSS, JavaScript для створення інтерфейсу. Для бізнес-логіки та ефективну обробку запитів розроблено на основі Spring Boot. Для роботи з базою даних використовується Spring Data JPA, а Spring Security надає безпеку даним. Для середовища розробки було взято IntelliJ IDEA, для контролю версій Git, для управління залежностями Maven. Також було сформульовано технічні вимоги до серверної та клієнтської частини. Та проведено аналіз охорони праці, приділено уваги до ергономічних вимог до робочого місця розробника та правилам поведінки в критичних ситуаціях та інше. В практичному цінності web-застосунок може бути впроваджений на залізничних вокзалах для підвищення інформування пасажирів та модернізація сервісів. Отриманий досвід в ході дослідження може слугуватиме основою для розробки інших інформаційних систем. При подальшому розвитку web-застосунка можна додати розробка мобільного застосунку, доступ та навігація для забезпечення рішення безпеки, розробка механізмів захисту від кібератак та забезпечення міжнародними стандартами захисту даних. Отже, кваліфікаційна робота успішно реалізована з поставленою метою та демонструє розуміння предметної області для залізничного вокзалу, сучасні підходи до розробки програмного забезпечення та технічного забезпечення ї важливість питання охорони праці.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Економіка залізничного транспорту: Навчальний посібник / В. П. Яновська, М. В. Макаренко, В. І. Творонович, О. Р. Приймук та ін.; кер. кол. авт. і наук. ред. В. П. Яновська. 2-ге вид., перероб. Київ: ПрофКнига, 2019. 364 с. (дата звернення: [18.03.2025]);
2. Укрзалізниця: стаття у Вікіпедії. URL: <https://uk.wikipedia.org/wiki/%D0%A3%D0%BA%D1%80%D0%B7%D0%B0%D0%BB%D1%96%D0%B7%D0%BD%D0%B8%D1%86%D1%8F> (дата звернення: [18.03.2025]);
3. HTML: стаття у Вікіпедії. URL: <https://uk.wikipedia.org/wiki/HTML> (дата звернення: [15.04.2025]);
4. CSS: стаття у Вікіпедії. URL: <https://uk.wikipedia.org/wiki/CSS> (дата звернення: [15.04.2025]);
5. JavaScript: стаття у Вікіпедії. URL: <https://uk.wikipedia.org/wiki/JavaScript> (дата звернення: [15.04.2025]);
6. Java: стаття у Вікіпедії. URL: <https://uk.wikipedia.org/wiki/Java> (дата звернення: [15.04.2025]);
7. Spring Framework: стаття у Вікіпедії. URL: https://uk.wikipedia.org/wiki/Spring_Framework (дата звернення: [15.04.2025]);
8. MySQL: стаття у Вікіпедії. URL: <https://uk.wikipedia.org/wiki/MySQL> (дата звернення: [15.04.2025]);
9. Apache Kafka: стаття у Вікіпедії. URL: https://uk.wikipedia.org/wiki/Apache_Kafka (дата звернення: [15.04.2025]);
10. IntelliJ IDEA: стаття у Вікіпедії. URL: https://uk.wikipedia.org/wiki/IntelliJ_IDEA (дата звернення: [15.04.2025]);
11. Ізмайлова О. В. Проектування інформаційних систем: Навчальний посібник. Черкаси: ЧНУ ім. Б. Хмельницького, 2019. 186 с. (дата звернення: 15.05.2025);

12. Коробйова Р.Г., Германюк Ю.М. Організація роботи залізничних вокзальних комплексів: навчальний посібник для ВНЗ. Львів: СПОЛОМ, 2023. 218 с. (дата звернення: [18.03.2025]);

13. Литвин В. В., Пасічник В. В., Шаховська Н. Б. Проектування інформаційних систем: Навчальний посібник (затв. МОН України). Львів: Магнолія, 2024. 360 с. (дата звернення: 15.05.2025);

14. Schildt H. Java: A Beginner's Guide. 8th Edition. McGraw-Hill Education, 2018. 684 p. (дата звернення: 17.05.2025);

15. Spring Boot Reference Documentation. URL: <https://docs.spring.io/spring-boot/docs/current/reference/html/> (дата звернення: 17.05.2025);

16. Кужель В.О. Безпека життєдіяльності та охорона праці: навч. посіб. Дніпро: НМетАУ, 2019. -188с. Режим доступу: <https://ir.nmu.org.ua/server/api/core/bitstreams/15e3ad9e-83ea-44a6-8c8b-ad8c2ab69f85/content> (дата звернення: 28.05.2025);

17. Ситнік Б. Т. Основи інформаційних систем і технологій: навчальний посібник. Харків: УкрДУЗТ, 2019. 175 с. URL: <http://lib.kart.edu.ua/bitstream/123456789/2174/1/%D0%9D%D0%B0%D0%B2%D1%87%D0%B0%D0%BB%D1%8C%D0%BD%D0%B8%D0%B9%20%D0%BF%D0%BE%D1%81%D1%96%D0%B1%D0%BD%D0%B8%D0%BA.pdf> (дата звернення: 15.04.2025);

18. MySQL 8.0 Reference Manual. URL: <https://dev.mysql.com/doc/refman/8.0/en/> (дата звернення: 17.04.2025);

19. Укрзалізниця. Річний звіт 2023. URL: https://uz.gov.ua/files/file/about/activity/annual_reports/UR_Annual_Report_2023.pdf (дата звернення: 20.05.2025).

ДОДАТОК А – ІНСТРУКЦІЯ КОРИСТУВАЧА

В цьому додатку представлено вказівки використання web-застосунку. Ця інструкція призначена для клієнтів, які хочуть отримати інформацію про розклад та скористатися функціями застосунку та для адміністраторів які управляють даними для залізничного вокзалу.

Для доступу до web-застосунку потрібно відкрити web-браузер на пристрої, ввести назву або ввести посилання на web-застосунку, далі потрібно натиснути на Enter та після завантаження потрапляємо на сторінку застосунку де потрібно обрати роль Клієнт/Адміністратор. Після обраної ролі потрапляємо до сторінки з Входом/Реєстрацією для клієнта, або сторінка з входом для Адміністратора. На рис. Додаток А. 1 та рис. Додаток А. 2 зображено як виглядають ці сторінки.

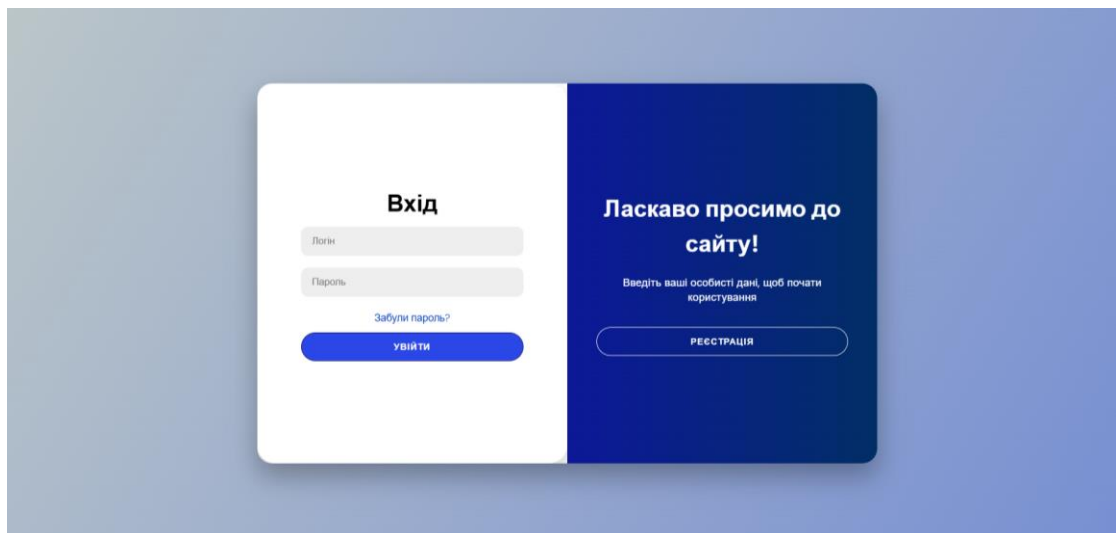


Рис. Додаток А.1- Сторінка для Входу/Реєстрації для Клієнта

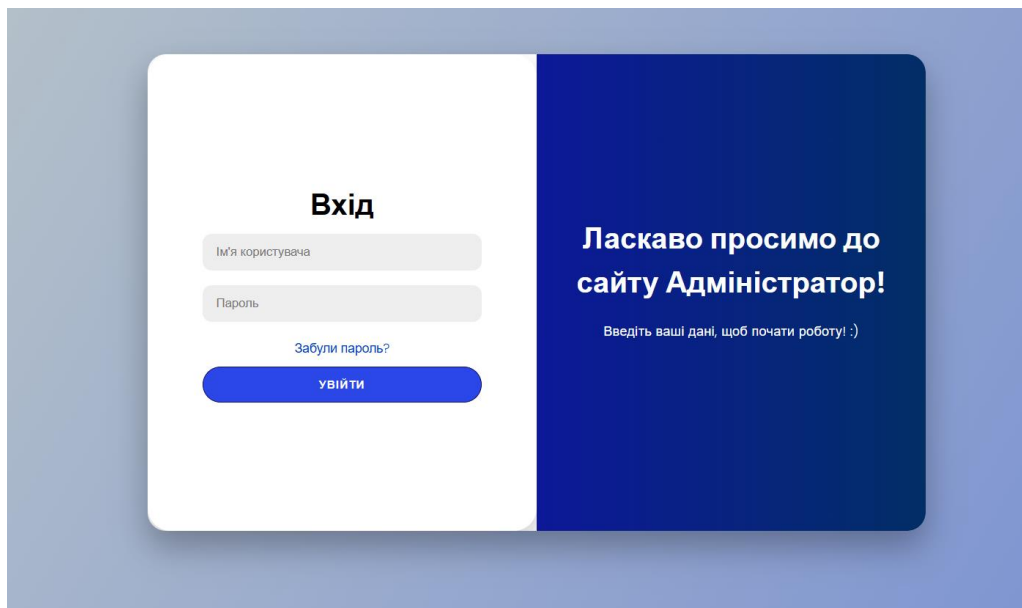


Рис. Додаток А.2- Сторінка для Входу для Адміністратора

Для реєстрації клієнта потрібно натиснути на кнопку Реєстрація та переходить до форми реєстрації. В цій формі потрібно заповнити форму, а саме ПІБ(Прізвище, Ім'я, По-батькові), Тип документа для подальшої купівлі квитка та отримання знижок, Номер документа для підтвердження особи, Номер телефона, Електронна пошта, Логін та Пароль. Після заповнення форми потрібно натиснути зареєструватися та після чого застосунок автоматично переходить до форми Вхід. На рис. Додаток А. 3 зображено як виглядають ця сторінка.

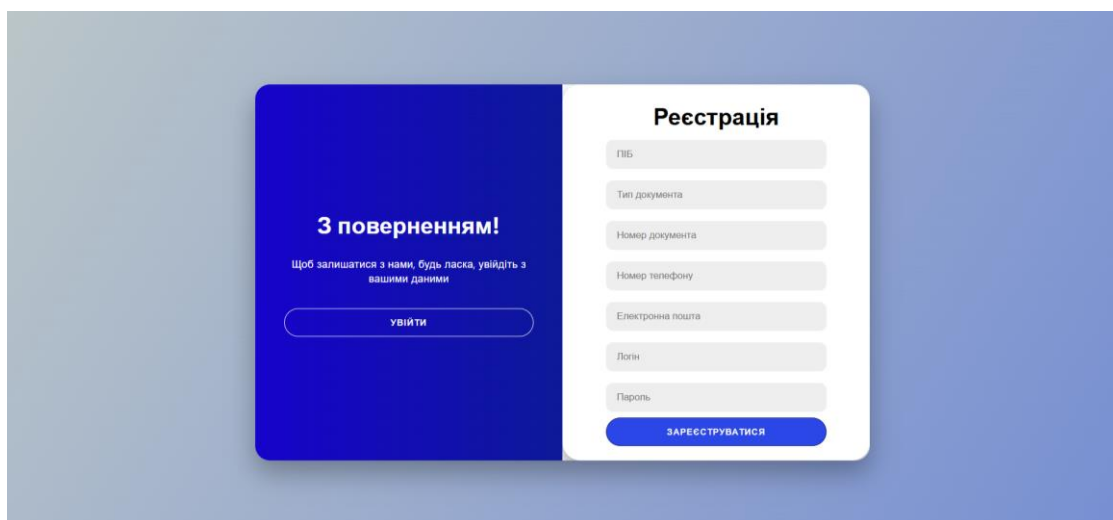


Рис. Додаток А.3- Сторінка для Реєстрації для Клієнта

Для входу до головного меню застосунку потрібно ввести Логін та Пароль на який зареєстрований Клієнт або Адміністратор. Якщо Логін та

Пароль дійсний та правильний то переходить до сторінки головного меню. Якщо ні, то з'являється повідомлення про неправильний Пароль чи Логін. Якщо клієнт/Адміністратор забув пароль, то є посилання де відкривається віконце з пропозицією змінити пароль.

Після входу в систему, Адміністратору дають доступ до панелі управління та основних функцій. Редактор для потяга надає дозвіл для перегляду, додавання, редагування, та видалення інформації про потяг. На цій сторінці відображається таблиця з даними, біля цих даних знаходиться кнопка Редагувати та Видалити, а також зверху над таблицею є кнопка Додати. Для додавання потрібно натиснути кнопку Додати, далі з'являється форма для заповнення даними (номер потяга, назва маршруту, тип потяга, загальна кількість вагонів, статус потяга), після заповнення натиснути Додати та дані про потяг додається до таблиці. Для редагування потрібно натиснути на кнопку Редагувати до рядку даних якій хочемо змінити, з'являється форма для зміни даними (назва маршруту, загальну кількість вагонів, статус вагона), після редагування натиснути на кнопку Редагувати та змінюються рядок з даними який змінили. Для видалення даних потрібно натиснути кнопку Видалити до рядку з даними який хочемо видалити, з'являється вікно з підтвердженням Що дійсно хочемо видалити?, якщо натискаємо Ні то все залишається без змін, якщо Так до видалення дані якій обрали. Таким чином працює Редактор для Вагонів, Редактор для Місць у вагоні, Редактор для Станцій, Редактор Розкладу, Редактор між потягом та вагоном, Редактор між потягом та станцією. Надсилання відповідей та сповіщень знаходиться список повідомлень від клієнтів. Для надсилання відповідей потрібно обрати запит зі списку та натиснути кнопку Відповісти, далі відкривається форма для відповідей в форматі чату та написати відповідь вручну або натиснути кнопку шаблон відповіді та обрати підходящу відповідь до запитання, після чого натиснути на кнопку відправити та повідомлення відправляється до клієнта. А сповіщення надсилається клієнту в тому випадку коли адміністратор змінює розклад. Для перегляду аналітики потрібно перейти до сторінки Аналітика даних, там

можна обрати що саме можна зробити аналіз (наприклад, по продажі квитків на обраний маршрут чи потяг, кількість людей на обраний потяг) у вигляді графіків.

Після входу в систему, Клієнту дають функції такі як Розклад поїздів, Купити квиток, Пошук поїзда, Особистий кабінет, Повернути квиток, Звернутися до підтримки. Для того щоб знайти потяг потрібно зайти на сторінку пошук поїзда, в якій є поля для написання назви станції відправлення та назви станції прибуття, також поле для обирання дати та час відправлення, та натиснути на кнопку пошук, після чого з'являється таблиця з поїздами які підходять під ці параметри, та біля даних поїзда є кнопка купити квиток. Після натискання цієї кнопки переходим на сторінку Купити квиток де вже заповнено дані про потяг (дата та час відправлення, обрана станція відправлення, обрана станція прибуття, дата та час прибуття), далі потрібно обрати вагон, для вибір місця у вагоні з'являється карта вагону за його типом (плацкарт, купе, люкс) та обирається вільне місце, для вибору додаткових послуг з'являється блок з послугами, а також автоматично заповнюється дані клієнта у полі ПІБ, після полів вибору знаходиться ціна яка змінюється за послугами та місцем у вагоні та кнопка купити квиток, після натискання Купити квиток переходить до зовнішньої платіжної системи, де вводиться банківські дані та після підтвердження оплати переходить на сторінку з повідомлення про успішну покупку, та на електронну пошту приходить повідомлення з електронним квитком. Для знаходження розклад потрібно зайти на сторінку Розклад поїздів, там знаходиться поля для написання назви станції відправлення та станції прибуття, дата та час та кнопка Пошук. Після натискання на кнопку Пошук з'являється таблиця з підходящими потягами та розкладом їхнього руху. Для повернення квитка потрібно перейти до сторінки Повернути квиток, на цій сторінці знаходиться докладні правила повернення квитка та в якому випадку можна їх повертати, знаходяться поля для вводу даних, а саме номер квитка який можна повернути, та кнопка Повернути квиток, після чого дані з квитка який повернули змінюють на доступний, та приходить повідомлення що цей

квиток тепер не дійсний та кількість діб для повернення коштів кошти. В особистому кабінеті можна змінити дані (ПІБ, номер документа якщо змінився, номер телефону), а також можна звернутися до служби підтримки з питанням. Для звернення до підтримки потрібно перейти до сторінки Звернутися до підтримки в форматі чату де буде знаходитися поле для написання питання та кнопка відправити повідомлення. Та після отримання повідомлення приходить сповіщення на електронну пошту про відповідь з підтримки.

ДОДАТОК Б – ТЕХНІЧНЕ ЗАВДАННЯ

Назва роботи: Розроблення web-застосунку для залізничного вокзалу.

Підстава для розроблення: Кваліфікаційна робота на здобуття ступеня вищої освіти "бакалавр" за спеціальністю 122 "Комп'ютерні науки", освітня програма "Комп'ютерні науки".

Виконавець: Студент групи 4141 Яременко Д.О.

Термін виконання: Згідно з графіком виконання кваліфікаційної роботи, 20.06.2025 рік.

Призначення web-застосунка:

Автоматизація процесів відображення та оновлення розкладу потягів;

Надання клієнтам швидкого та зручного доступу до актуальної інформації про відправлення та прибуття, розклад руху;

Надання адміністрації вокзалу інструменти для управління даними про поїзда, вагони, місця у вагоні, розклад, станції та інше;

Спрощення процесу отримання додаткової інформації для клієнтів.

Мета розробки є:

Створення зрозумілого та функціонального web-застосунок для клієнтів, який дозволить краще знаходити інформацію;

Створення адміністративної панелі в форматі web-застосунка для управління даними, що забезпечить актуальністю інформації;

Оптимізувати взаємодію між Адміністратором та Клієнтом;

Підвищити ефективність роботи залізничного вокзалу через автоматизацію операцій з оновлення даних;

Отримати практичні навички в розробці web-застосунку з використання технологій, як Java, Spring Boot, MySQL, HTML, CSS, JavaScript.

Користувачі системи:

Адміністратор - це користувач з повними правами доступу до адміністративної панелі системи. Його функції: керування даними про потяги, вагонів, місця у вагоні, станціями, зупинки потягів на станціях, розкладом. Та перегляд звітності;

Клієнт – це користувач, який отримує доступ до публічної частини web-застосунку. Його функція: перегляд розкладу потягів, пошук потягів за станцією відправлення/прибуття, датою та час, купівлі квитків на потяг, повернення квитків.

Функціональні вимоги до web-застосунку:

Для адміністраторів: Вхід (надання форми для входу, перевірка введених даних, перенаправлення адміністраторів до головної панелі), адміністративна панель(зручна навігація по розділах управління), управління потягами(реалізація додавання, редагування та видалення даних), управління вагонами(реалізація додавання, редагування та видалення даних), управління місця у вагоні(реалізація додавання, редагування, видалення, відновлення даних), управління станціями (реалізація додавання, редагування та видалення даних), управління зупинки потягів (реалізація додавання, редагування та видалення даних), управління розкладом (реалізація додавання, редагування та видалення даних), перегляд звітності(аналіз продажу та інше).

Для клієнтів: Вхід та Реєстрація (надання форми для входу та реєстрації, успішного входу до web-застосунку),перегляд розкладу(відображення актуальної інформації розкладу, надання функції пошуку за різними критеріями), купівля квитків (надання послуг купівлі квитків за обраним потягом, станцію відправлення/прибуття, датою та час, надання додаткових послуг, надання місце у вагоні), зворотній зв'язок(надає для питання до адміністратора), повернення квитків(надання послуг повернення за номером квитка).

Нефункціональні вимоги: надійність (забезпечення доступності web-застосунку, стійкість до системних помилок, збереження даних у базі даних), продуктивність(оптимізація запитів до бази даних, витримувати одночасну роботу), безпека (захист від небажаного доступу до адміністративної панелі, надійне зберігання паролів), зручність (зрозумілий інтерфейс), масштабованість (розширення функціоналу), сумісність (робота web-застосунку в основних web-браузерах).

Вимоги до програмного середовища:

Мова програмування:Java

Каркас для побудови: Spring Boot

База даних:MySQL

Презентаційна частина: HTML, CSS, JavaScript

Середовище розробки: IntelliJ IDEA Ultimate

Web-сервер: Tomcat

Система контролю версій: Git

Операційна система для розробки: Windows 10

Система управління збирання проекту: Maven.

Архітектура застосунку: Рівень представлення(відповідає за взаємодію з клієнтом/адміністраторів до системи), Рівень бізнес-логіки (містить основну логіку обробки запитів, взаємодія з базою даних через сервіси та репозиторії), Рівень бази даних (відповідає за зберігання, управління та доступ до даних).

Модулі системи: Модуль "Вхід та Реєстрація", Модуль "Перегляд розкладу" (для клієнтів), Модуль "Управління потягами" (для адміністратора), Модуль "Управління вагонами" (для адміністратора), Модуль "Управління місцями у вагоні" (для адміністратора), Модуль "Управління станціями" (для адміністратора), Модуль "Управління зупинками потягів" (для адміністратора), Модуль "Управління розкладом" (для адміністратора), Модуль "Купівля та Повернення квитків" (для клієнтів), Модуль "Зворотній зв'язок" (для клієнтів), Модуль "Звітність" (для адміністратора), Модуль "Робота з базою даних".

Взаємодія з модулями: Клієнтська частина ініціює HTTP-запити до серверної частини, в свою чергу серверна частина приймає запити за допомогою контролера, контролери виконують запити відповідним сервісам, сервіси взаємодіють з репозиторіями для доступу бази даних та після обробки сервіси формують та відправляють відповідь до клієнтської частини для відображення.

Етапи розробки: Збір та аналіз функціональних та нефункціональних вимог до web-застосунку, розробка основного функціоналу, розгортання web-застосунку на сервіси, підготовка документації, підготовка до захисту.

ДОДАТОК В – ФРАГМЕНТ ПРОГРАМНОГО КОДУ

Реалізація функціоналу реєстрації та авторизації клієнтів

У цьому додатку представлено ключові фрагменти програмного коду, що відповідають за реалізацію функціоналу реєстрації та авторизації користувачів в інформаційній системі залізничного вокзалу. Надані фрагменти демонструють взаємодію між клієнтською частиною та серверною частиною, а також логіку обробки введених користувачем даних, їх валідації та забезпечення безпеки.

На лістингу 1 написано код для контролера який обробляє HTTP-запити, пов'язані з реєстрацією та авторизацією клієнтів. Він надає сторінки для входу та реєстрації, а також методи для обробки даних форм реєстрації та логіну.

Лістинг-1. Код до класу ClientsController.

```
@Controller
public class ClientsController {
    @Autowired
    private ClientsService clientService;
    @GetMapping("/login-register")
    public String loginRegisterPage(Model model) {
        model.addAttribute("client", new Clients());
        return "login_registerClients";
    }
    @PostMapping("/register-client")
    public String registerClient(@ModelAttribute("client")
Clients client, Model model) {
        System.out.println("Реєстрація клієнта: " +
client.getFullName());
        try {
            clientService.registerClient(client);
            return "redirect:/login_registerClients";
        } catch (Exception e) {
            model.addAttribute("error", "Помилка реєстрації: " +
e.getMessage());
            return "login_registerClients";
        }
    }
}
```

```

    }
}

@GetMapping("/loginClients")
public String loginPage(Model model) {
    model.addAttribute("client", new Clients());
    return "login_registerClients";
}

@PostMapping("/loginCl")
public String loginClient(@ModelAttribute("client") Clients
client, Model model, HttpSession session) {
    try {
        Clients authenticatedClient =
clientService.authenticateClient(client.getUsername(),
client.getPasswordUser());
        session.setAttribute("client", authenticatedClient);
        return "redirect:/client-home";
    } catch (Exception e) {
        model.addAttribute("errorLogin", "Невірний логін або
пароль");
        return "login_registerClients";
    }
}

@GetMapping("/client-home")
public String clientHome(HttpSession session, Model model) {
    Clients client = (Clients)
session.getAttribute("client");
    if (client == null) {
        return "redirect:/login-register";
    }
    model.addAttribute("client", client);
    return "client_home";
}
}

```

На лістингу 2 написано код для контролера який обробляє HTTP-запити, пов'язані з авторизацією адміністратора. Він надає сторінки для входу, а також методи для обробки даних форм логіну.

Лістинг-2. Код до класу AdminController.

```
@Controller
public class AdminController {
    @Autowired
    private AdminService adminService;
    @GetMapping("/loginAdmin")
    public String loginPage(Model model) {
        model.addAttribute("admin", new AdminUsers());
        return "loginAdmins";
    }
    @PostMapping("/loginAdm")
    public String loginAdmin(@ModelAttribute("admin") AdminUsers
adminUsers, Model model, HttpSession session) {
        try {
            AdminUsers authenticatedAdmin =
adminService.authenticateAdmin(adminUsers.getUsername(),
adminUsers.getPasswordAdmin());
            session.setAttribute("admin", authenticatedAdmin);
            return "redirect:/admin-home";
        } catch (Exception e) {
            model.addAttribute("errorLogin", "Невірний логін або
пароль");
            return "loginAdmins";
        }
    }
    @GetMapping("/admin-home")
    public String adminHome(HttpSession session, Model model) {
        AdminUsers adminUsers = (AdminUsers)
session.getAttribute("admin");
        if (adminUsers == null) {
            return "redirect:/loginAdmin";
        }
    }
}
```



```

    }
    model.addAttribute("admin", adminUsers);
    return "admin_home";
}
}

```

На лістинингу 3 написано код до інтерфейсу що розширює JpaRepository, надаючи базові операції CRUD для сутності Clients та дозволяючи знаходити клієнта за іменем користувача.

Лістинг-2. Код до класу ClientsRepository.

```

@Repository
public interface ClientsRepository extends JpaRepository<Clients,
Long> {
    Optional<Clients> findByUsername(String username);
}

```

На лістинингу 4 написано код до інтерфейсу що розширює JpaRepository, надаючи базові операції CRUD для сутності Admin та дозволяючи знаходити адміністратор за логіном.

Лістинг-4. Код до класу AdminRepository.

```

@Repository
public interface AdminRepository extends
JpaRepository<AdminUsers, Integer> {
    Optional<AdminUsers> findByUsername(String username);
}

```

На лістингу 5 написано код для сутності, яка представляє клієнта в базі даних. Він визначає структуру даних для збереження інформації про клієнтів, включаючи особисті дані, облікові дані для входу та дату реєстрації.

Лістинг-5. Код до класу Clients.

```

@Entity
@Table(name = "clients")
public class Clients {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")

```

```

private Long id;
@Column(name = "fullName", length = 100, nullable = false)
private String fullName;
@Enumerated(EnumType.STRING)
@Column(name = "typeDoc", nullable = false)
private TypeDoc typeDoc;
@Column(name = "numberDoc", length = 50, nullable = false)
private String numberDoc;
@Column(name = "username", length = 100, unique = true,
nullable = false)
private String username;
@Column(name = "passwordUser", length = 100, nullable =
false)
private String passwordUser;
@Column(name = "email", length = 100, unique = true,
nullable = false)
private String email;
@Column(name = "numberPhone", length = 13, nullable =
false)
private String numberPhone;
@Column(name = "dataRegistr", nullable = false)
private Timestamp dataRegistration;
public Clients() {}
public Clients(String fullName, String numberDoc, String
username, String passwordUser, String email, String numberPhone,
Timestamp dataRegistration) {
    this.fullName = fullName;
    this.numberDoc = numberDoc;
    this.username = username;
    this.passwordUser = passwordUser;
    this.email = email;
    this.numberPhone = numberPhone;
    this.dataRegistration = dataRegistration;
}
public Long getId() { return id; }

```

```

        public void setId(Long id) { this.id = id; }
        public String getFullName() { return fullName; }
        public void setFullName(String fullName) { this.fullName
= fullName; }
        public String getNumberDoc() { return numberDoc; }
        public void setNumberDoc(String numberDoc) {
this.numberDoc = numberDoc; }
        public String getUsername() { return username; }
        public void setUsername(String username) { this.username
= username; }
        public String getPasswordUser() { return passwordUser; }
        public void setPasswordUser(String passwordUser) {
this.passwordUser = passwordUser; }
        public String getEmail() { return email; }
        public void setEmail(String email) { this.email = email;
}

        public String getNumberPhone() { return numberPhone; }
        public void setNumberPhone(String numberPhone) {
this.numberPhone = numberPhone; }
        public Timestamp getDataRegistration() { return
dataRegistration; }
        public void setDataRegistration(Timestamp
dataRegistration) { this.dataRegistration = dataRegistration; }
        public TypeDoc getTypeDoc() { return typeDoc; }
        public void setTypeDoc(TypeDoc typeDoc) { this.typeDoc =
typeDoc; }
        public enum TypeDoc {
            Паспорт, Дитячий, Студентський, Пільговий
        }
    }
}

```

На лістингу 6 написано код для сутності, яка представляє адміністратора в базі даних. Він визначає структуру даних для збереження інформації про адміністратора, включаючи особисті дані, облікові дані для входу, статус та дату реєстрації.

Лістинг-6. Код до класу AdminUsers.

```

@Entity
@Table(name = "admins_users")
public class AdminUsers {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "adminID")
    private int admin_id;
    @Column(name = "fullName", length = 100, nullable = false)
    private String fullName;
    @Column(name = "position", length = 100, nullable = false)
    private String position;
    @Enumerated(EnumType.STRING)
    @Column(name = "statusUser", nullable = false)
    private StatusUser statusUser;
    @Column(name = "username", length = 50, unique = true, nullable
= false)
    private String username;
    @Column(name = "passwordAdmin", length = 100, nullable =
false)
    private String passwordAdmin;
    @Column(name = "email", length = 100, unique = true, nullable
= false)
    private String email;
    @Column(name = "numberPhone", length = 13, nullable = false)
    private String numberPhone;
    @Column(name = "dataRegistration", nullable = false)
    private LocalDateTime dataRegistration;
    public AdminUsers() {}
    public AdminUsers(String fullName, String position, String
username, String passwordAdmin, String email, String numberPhone,
LocalDateTime dataRegistration) {
        this.fullName = fullName;
        this.position = position;

```

```

        this.username = username;
        this.passwordAdmin = passwordAdmin;
        this.email = email;
        this.numberPhone = numberPhone;
        this.dataRegistration = dataRegistration;
    }
    public int getAdmin_id() { return admin_id; }
    public void setAdmin_id(int admin_id) { this.admin_id =
admin_id; }
    public String getFullName() { return fullName; }
    public void setFullName(String fullName) { this.fullName =
fullName; }
    public String getPosition() { return position; }
    public void setPosition(String position) { this.position =
position; }
    public StatusUser getStatusUser() { return statusUser; }
    public void setStatusUser(StatusUser statusUser) {
this.statusUser = statusUser; }
    public String getUsername() { return username; }
    public void setUsername(String username) { this.username =
username; }
    public String getPasswordAdmin() { return passwordAdmin; }
    public void setPasswordAdmin(String passwordAdmin) {
this.passwordAdmin = passwordAdmin; }
    public String getEmail() { return email; }
    public void setEmail(String email) { this.email = email; }
    public String getNumberPhone() { return numberPhone; }
    public void setNumberPhone(String numberPhone) {
this.numberPhone = numberPhone; }
    public LocalDateTime getDataRegistration() { return
dataRegistration; }
    public void setDataRegistration(LocalDateTime
dataRegistration) { this.dataRegistration = dataRegistration; }
    public enum StatusUser {
        Звільнений, У_відпустці, Працює

```

```

    }
}

```

На лістингу 7 написано код для сервісу який містить бізнес-логіку для реєстрації та автентифікації клієнтів. Він взаємодіє з репозиторієм для доступу до бази даних, виконує перевірку існування користувача та хешування паролів.

Лістинг-7. Код до класу ClientsService.

```

@Service
public class ClientsService {
    private ClientsRepository clientRepository;
    private BCryptPasswordEncoder passwordEncoder = new
BCryptPasswordEncoder();
    public ClientsService(ClientsRepository
clientRepository) {
        this.clientRepository = clientRepository;
    }
    public void registerClient(Clients client) throws
Exception {
        Optional<Clients> existingClient =
clientRepository.findByUsername(client.getUsername());
        if (existingClient.isPresent()) {
            throw new Exception("Користувач з таким іменем
вже існує");
        }
        String encryptedPassword =
passwordEncoder.encode(client.getPasswordUser());
        client.setPasswordUser(encryptedPassword);
        client.setDataRegistration(new
Timestamp(System.currentTimeMillis()));
        clientRepository.save(client);
        System.out.println("Клієнт збережений в базі: " +
client.getUsername());
    }
}

```

```

        public Clients authenticateClient(String username, String
password) throws Exception {
            Optional<Clients> clientOptional =
clientRepository.findByUsername(username);
            if (clientOptional.isPresent()) {
                Clients client = clientOptional.get();
                if (passwordEncoder.matches(password,
client.getPasswordUser())) {
                    return client;
                }
            }
            throw new Exception("Невірні дані для входу");
        }
    }
}

```

На лістингу 8 написано код для сервісу який містить бізнес-логіку для авторизації адміністратора. Він взаємодіє з репозиторієм для доступу до бази даних, виконує перевірку існування користувача та хешування паролів.

Лістинг-8. Код до класу AdminService.

```

@Service
public class AdminService {
    private AdminRepository adminRepository;
    private BCryptPasswordEncoder passwordEncoder = new
BCryptPasswordEncoder();
    public AdminService(AdminRepository adminRepository) {
        this.adminRepository = adminRepository;
    }
    public AdminUsers authenticateAdmin(String username, String
passwordAdmin) throws Exception{
        Optional<AdminUsers> adminOptional =
adminRepository.findByUsername(username);
        if (adminOptional.isPresent()) {
            AdminUsers adminUsers = adminOptional.get();
            if
(passwordAdmin.equals(adminUsers.getPasswordAdmin())) {

```

```

        return adminUsers;
    }
}
throw new Exception("Невірні дані для входу");
}
}

```

На лістингу 9 написано код для автентифікації та авторизації. Він визначає доступ до URL-адрес, форми логіну, обробку успішної авторизації, хешування паролів та управління автентифікацією.

Лістинг-9. Код до класу Security.

```

@Configuration
public class Security {
    @Autowired
    private CustomAuthenticationSuccessHandler successHandler;
    private PasswordEncoder passwordEncoder;

    @Bean
    @Order(1)
    public SecurityFilterChain
adminSecurityFilterChain(HttpSecurity http) throws Exception {
        http
            .securityMatcher("/admin/**", "/loginAdmin",
"/loginAdm")
            .csrf(csrf -> csrf.disable())
            .authorizeHttpRequests(auth -> auth
                .requestMatchers("/loginAdmin",
"/loginAdm").permitAll()

            .requestMatchers("/admin/**").hasRole("ADMIN")
                .anyRequest().authenticated()
            );
        return http.build();
    }
}
@Bean

```



```

@Order(2)

public SecurityFilterChain
clientSecurityFilterChain(HttpSecurity http) throws Exception {
    http
        .securityMatcher("/client/**", "/loginClients",
"/login")
        .csrf(csrf -> csrf.disable())
        .authorizeHttpRequests(auth -> auth
            .requestMatchers("/", "/loginClients",
"/register-client", "/style1.css",
"/script1.js", "/styleMenuClient.css").permitAll()
            .requestMatchers("/client/**").hasRole("CLIENT")
            .anyRequest().authenticated()
        )
        .formLogin(form -> form
            .loginPage("/loginClients")
            .loginProcessingUrl("/login")
            .successHandler(successHandler)
            .failureUrl("/loginClients?error=true")
            .permitAll()
        )
        .logout(logout -> logout
            .logoutUrl("/logout")

.logoutSuccessUrl("/loginClients?logout=true")
            .permitAll()
        );
    return http.build();
}

@Bean
public PasswordEncoder passwordEncoder() {
    return NoOpPasswordEncoder.getInstance();
}
}

```

На лістингу 10 написано код для HTML-шаблону якій відображає інтерфейс для реєстрації нового клієнта та входу для існуючих користувачів. Він використовує JavaScript для перемикання між формами.

Лістинг-10. Код до класу HTML.

```
<!DOCTYPE html>
<html lang="uk" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Сторінка входу та реєстрації</title>
    <link rel="stylesheet" href="/style1.css">
</head>
<body>
<div class="container" id="container">
    <div class="form-container sign-up-container">
        <form
                                th:action="@{/register-client}"
th:object="$${client}" method="post">
            <h1>Реєстрація</h1>
            <input
                type="text"
                th:field="*{fullName}"
placeholder="ПІБ" required />
            <input
                list="doc-types"
                th:field="*{typeDoc}"
placeholder="Тип документа" required>
            <datalist id="doc-types">
                <option value="Паспорт" >
                <option value="Дитячий" >
                <option value="Студентський" >
                <option value="Пільговий" >
            </datalist>
            <input
                type="text"
                th:field="*{numberDoc}"
placeholder="Номер документа" required>
            <input
                type="text"
                th:field="*{numberPhone}"
placeholder="Номер телефону" required>
            <input
                type="email"
                th:field="*{email}"
placeholder="Електронна пошта" required />
```

```

        <input          type="text"          th:field="*{username}"
placeholder="Логін" required />
        <input          type="password"      th:field="*{passwordUser}"
placeholder="Пароль" required />
        <button type="submit">Зареєструватися</button>
    </form>
</div>
<div class="form-container sign-in-container">
    <form          th:action="@{/loginCl}"      th:object="${client}"
method="post">
        <h1>Вхід</h1>
        <input          type="text"          th:field="*{username}"
placeholder="Логін" required />
        <input          type="password"      th:field="*{passwordUser}"
placeholder="Пароль" required />
        <a href="#">Забули пароль?</a>
        <button type="submit">Увійти</button>
    </form>
</div>
<div class="overlay-container">
    <div class="overlay">
        <div class="overlay-panel overlay-left">
            <h1>З поверненням!</h1>
            <p>Щоб залишатися з нами, будь ласка, увійдіть з
вашими даними</p>
            <button class="ghost" id="signIn">Увійти</button>
        </div>
        <div class="overlay-panel overlay-right">
            <h1>Ласкаво просимо до сайту!</h1>
            <p>Введіть ваші особисті дані, щоб почати
користування</p>
            <button
                                class="ghost"
id="signUp">Реєстрація</button>
        </div>
    </div>
</div>

```

```

        </div>
    </div>
</script>
</body>
</html>

```

На лістингу 11 написано код для HTML-шаблону якій відображає інтерфейс входу для адміністратора. Він використовує JavaScript для перемикання між формами.

Лістинг-11. Код до класу HTML.

```

<!DOCTYPE html>
<html lang="uk" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Стопінка входу</title>
    <link rel="stylesheet" href="/style1.css">
</head>
<body>
<div class="container" id="container">
    <div class="form-container sign-in-container">
        <form th:action="@{/loginAdm}" method="post"
th:object="${admin}">
            <h1>Вхід</h1>
            <input type="text" th:field="*{username}"
placeholder="Ім'я користувача" required />
            <input type="password" th:field="*{passwordAdmin}"
placeholder="Пароль" required />
            <a href="#">Забули пароль?</a>
            <button type="submit">Увійти</button>
            <div th:if="${errorLogin}" th:text="${errorLogin}"
style="color:red"></div>
        </form>
    </div>
<div class="overlay-container">
    <div class="overlay">

```

```
<div class="overlay-panel overlay-right">
    <h1>Ласкаво просимо до сайту Адміністратор!</h1>
    <p>Введіть ваші дані, щоб почати роботу! :)</p>
</div>
</div>
</div>
<script src="/script1.js"></script>
</body>
</html>
```