

**Пояснювальна записка
до курсової роботи**

на тему: ВЕБ-СЕРВІС ДЛЯ КУПІВЛІ КВИТКІВ НА ПОТЯГ

КП. ІП-1311.045440.02.81

Київ – 2024

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	4
ВСТУП.....	5
1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	6
1.1 Загальні положення.....	6
1.2 Змістовний опис і аналіз предметної області	7
1.3 Аналіз існуючих технологій та успішних ІТ-проектів	10
1.3.1 Аналіз відомих алгоритмічних та технічних рішень.....	10
1.3.2 Аналіз допоміжних програмних засобів та засобів розробки	10
1.3.3 Аналіз відомих програмних продуктів	11
1.4 Аналіз вимог до програмного забезпечення.....	11
1.4.1 Розроблення функціональних вимог	12
1.4.2 Розроблення нефункціональних вимог	13
1.5 Постановка задачі.....	13
Висновки до розділу.....	14
2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	15
2.1 Моделювання та аналіз програмного забезпечення	15
2.2 Архітектура програмного забезпечення	16
2.3 Конструювання програмного забезпечення	16
2.4 Аналіз безпеки даних.....	19
Висновки до розділу.....	19
3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	21
3.1 Аналіз якості ПЗ.....	21
3.2 Опис процесів тестування	21
3.3 Опис контрольного прикладу	15
Висновки до розділу.....	15
4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.	26
4.1 Розгортання програмного забезпечення	26
4.2 Підтримка програмного забезпечення	27

Висновки до розділу	17
ВИСНОВКИ	28
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	29

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	–	Integrated Development Environment – інтегроване середовище розробки.
API	–	Application programming interface, прикладний програмний Інтерфейс
SDK	–	Software development kit
IT	–	Інформаційні технології
ER	–	Entity-Relation diagram
OC	–	Операційна система.
БД	–	База даних.
ORM	–	Object Relational Mapping

ВСТУП

<У вступі стисло викладають:

актуальність роботи та підстави для її виконання; світові тенденції розв'язання поставлених проблем і/або завдань; оцінку сучасного стану об'єкта розробки, розкриваючи практично розв'язані завдання провідними науковими установами та організаціями, а також провідними вченими й фахівцями певної галузі; можливі сфери застосування. 1 стр. чи більше>

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Веб-сервіси для купівлі квитків на потяги забезпечують зручний і ефективний спосіб планування подорожей залізницею. Ці сервіси дозволяють користувачам шукати, порівнювати ціни, бронювати, і купувати квитки на потяг онлайн, часто надаючи додаткові послуги, як от вибір місць та інформацію про розклади.

Залізниці з'явилися на початку 19 століття, символізуючи промислову революцію. Перші залізничні колії були побудовані в Англії для вантажних перевезень, а перший пасажирський поїзд здійснив рейс у 1825 році. Завдяки своїй ефективності та здатності перевозити великі обсяги вантажів і людей на далекі відстані, залізниці швидко розповсюдилися по всьому світу. До кінця 19 століття, залізниці стали основним видом транспорту, значно сприяючи економічному розвитку та глобалізації [1].

Предметна область "Веб-сервісу для купівлі квитків на потяг" охоплює ряд аспектів, пов'язаних з онлайн-продажем квитків, плануванням поїздок, управлінням замовленнями та інтерфейсом користувача. Основні елементи цієї предметної області включають:

1. Розклад поїздів: Інформація про час відправлення та прибуття поїздів, маршрути, частоту руху, тривалість поїздок.

2. Типи поїздів: Різні категорії поїздів, як-от швидкісні, регіональні, нічні та інші, кожен зі своїми особливостями та ціноутворенням.

3. Класи обслуговування та місця: Різні класи вагонів (економ-клас, бізнес-клас, спальні вагони тощо) із різними рівнями комфорту та вартості.

4. Ціноутворення та тарифи: Різні цінові категорії, знижки (наприклад, для студентів, пенсіонерів), акційні пропозиції.

5. Система бронювання квитків: Механізми пошуку та вибору місць, оформлення покупки квитків, опції оплати.

6. Інтерфейс користувача: Простота навігації сайту, доступність інформації, мобільна сумісність, можливості персоналізації.

7. Обслуговування клієнтів: Підтримка користувачів, вирішення проблем, зміни та скасування бронювань.

8. Інтеграція з іншими транспортними службами: Послуги трансферів, інтеграція з громадським транспортом, партнерства з авіакомпаніями для мульти-модальних поїздок.

9. Безпека та приватність: Захист даних користувачів, безпечні методи оплати, відповідність нормативам захисту даних.

10. Аналітика та звітність: Збір та аналіз даних про продажі, попит на різні маршрути

1.2 Змістовний опис і аналіз предметної області

Залізниця — це система транспорту, яка включає в себе поїзди, що рухаються по спеціально побудованих коліях. Основні компоненти та принципи залізниці включають:

1. Колії: Паралельні рейки, зазвичай зроблені зі сталі, які укладаються на шпалах. Колії визначають напрямок руху поїздів та підтримують їх стабільність.

2. Поїзди: Транспортні засоби, що складаються з одного або декількох вагонів, які тягнуться локомотивом. Існують різні типи поїздів, включаючи пасажирські, вантажні, швидкісні та міські електропоїзди.

3. Локомотиви: Тягові агрегати, які можуть бути паровими, дизельними, або електричними. Вони забезпечують потужність для руху поїздів.

4. Вокзали та станції: Місця, де пасажирів можуть сісти на поїзд або виходити з нього, а також здійснювати покупку квитків і отримувати інформацію.

5. Сигналізація та управління рухом: Системи, що забезпечують безпечне та ефективне керування рухом поїздів, уникнення зіткнень, та регулювання швидкості.

6. Інфраструктура: Включає мости, тунелі, переїзди, залізничні вузли, обладнання для технічного обслуговування та ремонту поїздів.

7. Транспортна логістика: Організація ефективного розкладу руху, маршрутів, вантажних та пасажирських перевезень.

Залізниця відіграє ключову роль в транспортній інфраструктурі багатьох країн, забезпечуючи ефективний та екологічно сталий транспорт.

У сфері ІТ предметна область залізниць застосовується для розробки та підтримки різноманітних технологічних рішень, які покращують ефективність, безпеку та зручність залізничних перевезень. Ось деякі приклади використання:

1. Системи бронювання та продажу квитків: Онлайн-платформи для придбання квитків, мобільні додатки для планування поїздок, електронні квитки, автоматизовані системи для обробки та управління бронюваннями.

2. Інформаційні системи для пасажирів: Електронні табло розкладів, інформаційні додатки, сповіщення про зміни в розкладі, реальний час відстеження поїздів.

3. Системи управління рухом поїздів: Автоматизовані системи для контролю та керування рухом поїздів, забезпечення безпеки, моніторингу стану колій.

4. Логістичні системи: Для ефективного планування маршрутів, розкладів, управління вантажними перевезеннями та оптимізації роботи залізничного транспорту.

5. Системи безпеки та нагляду: Відеонагляд, системи контролю доступу, технології виявлення та запобігання надзвичайним ситуаціям.

6. Аналітика даних і штучний інтелект: Аналіз великих даних для прогнозування попиту, оптимізації ресурсів, покращення якості обслуговування та стратегічного планування.

7. Інтеграція з іншими транспортними системами: Розробка інтегрованих мульти-модальних транспортних рішень, що зв'язують залізницю з іншими видами транспорту.

Ці та інші ІТ-рішення дозволяють залізничному транспорту бути більш ефективним, доступним та безпечним для пасажирів та вантажовласників.

1.3 Аналіз існуючих технологій та успішних ІТ-проектів

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації Trainly. Далі будуть розглянуті допоміжні програмні засоби, засоби розробки та готові програмні рішення.

1.3.1 Аналіз відомих алгоритмічних та технічних рішень

Алгоритмічна складність полягає в підборі необхідного маршруту, при умові що застосунок покриває велику географічну область. Найважливіша частина – доступ до розкладів та даних маршрутів різних перевізників. Знову ж таки, якщо ми не розглядаємо регіонального/національного перевізника, який часто має монополію або керується з боку держави. Дані можуть завантажуватися через програмні інтерфейси (API) зі сторонніх ресурсів, або напряду працівниками компанії. Необхідно розділяти маршрути, які часто слідує визначеному розкладу, і конкретні рейси на чітко визначений час. Покупець купує квитки не на маршрути, а на обрані рейси. В Trainly було обрано використовувати принцип resource-based APIs, які фокусуються на управлінні ресурсами такі як: потяг, маршрут, сидіння і т.д.

1.3.2 Аналіз допоміжних програмних засобів та засобів розробки

Для розробки будь-якого веб застосунку необхідні HTML, CSS, JavaScript та обрана мова серверної частини. Також не важливо які інструменти використовуються під час розробки, головне щоб клієнтська частина в кінці мала HTML, CSS та JS. Для Trainly було обрано використовувати SvelteKit та tailwindcss для клієнтської частини. Та Fastify framework in node.js runtime. Для зберігання даних краще підійде реляційна база даних, тому що дані мають глибоку зв'язаність. MySql або PostgreSQL – чудові варіанти для цієї задачі. Це досить універсальні sql бази даних, які мають гарну документацію та користуються попитом серед розробників

1.3.3 Аналіз відомих програмних продуктів

Аналогами є Trainline, Укр Залізниця та SNCB/NMBS.

Trainline – міжнародна компанія, яка покриває міжнародні перевезення. Вони реалізують велику частину функціоналу і вважаються одними з кращих на ринку. Укр Залізниця – національна компанія, яка спеціалізується на перевезеннях по Україні. SNCB/NMBS – національна компанія, яка спеціалізується на перевезеннях по Бельгії

1.4 Аналіз вимог до програмного забезпечення

<У підрозділі викладають:

опис головного(-х) функціоналу розробки, діаграму варіантів використання та опис варіантів використання. Обсяг 5 стр., чи більше >

Головною функцією програмного забезпечення є знаходження регіонального маршруту.

В таблицях 1.2 - 1.17 наведені варіанти використання програмного забезпечення.

Таблиця 1.2 - Варіант використання UC-1

Use case name	Пошук необхідного маршруту
Use case ID	UC-01
Goals	Список доступних рейсів
Actors	Користувач
Trigger	Користувач бажає знайти рейс
Pre-conditions	-
Flow of Events	Користувач переходить на головну сторінку. В полях вводяться відповідні дані: місто відбуття, місто прибуття, дата. Після заповнення даних користувача натискає кнопку пошуку. Після цього перенаправляється на сторінку рейсів.
Extension	В випадку недоступних рейсів користувач побачить пусту сторінку рейсів

Post-Condition	Користувач бачить рейси
----------------	-------------------------

Таблиця 1.3 - Варіант використання UC-2

Use case name	Обрання необхідного маршруту
Use case ID	UC-02
Goals	Деталі маршруту
Actors	Користувач
Trigger	Користувач бажає обрати рейс
Pre-conditions	Користувач на сторінці з доступними рейсами
Flow of Events	Користувач обирає необхідний рейс, клікає на нього та опиняється на сторінці з деталями
Extension	В випадку недоступних рейсів користувач побачить пусту сторінку рейсів
Post-Condition	Користувач бачить рейси

1.4.1 Розроблення функціональних вимог

<У пункті викладають:

модель вимог, їх опис та матрицю трасування.>

Програмне забезпечення розділене на модулі. Кожен модуль має свій певний набір функцій. На рисунку 1.4 наведено загальну модель вимог, а в таблицях 1.18 – 1.27 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог можна побачити на рисунку 1.5.

Рисунок 1.4 – Модель вимог у загальному вигляді

Таблиця 1.16 – Функціональна вимога FR-1

Назва	Реєстрація користувача
-------	------------------------

Опис	Система повинна надавати можливість реєстрації користувачеві шляхом введення пошти, паролю, підтвердження паролю.
------	---

	FR-1	FR-2	FR-3	FR-4	FR-5	FR-6	FR-7	FR-8	FR-9	FR-10	FR-11	FR-12	FR-13	FR-14	FR-15	FR-16
UC-1	+	+														
UC-2					+											
UC-3											+					
UC-4				+				+								
UC-5					+	+										
UC-6												+				
UC-7							+									
UC-8						+	+	+	+	+	+					
UC-9													+			
UC-10														+		
UC-11															+	
UC-12			+													

Рисунок 1.11 – Матриця трасування вимог

1.4.2 Розроблення нефункціональних вимог

<У пункті викладають:

опис нефункціональних вимог.>

1.5 Постановка задачі

<У підрозділі викладають:

що саме ви збираєтесь реалізувати. Це не зовсім вимоги і варіанти використання. Це достатньо узагальнене поняття з метою, цілями та задачами, що підлягають розв'язанню у результаті розробки програмного забезпечення на курсову роботу. Обсяг 1 сторінка, чи більше. >

Висновки до розділу

< Необхідно стисло описати усе, що було виконано у даному розділі. Обсяг 0,75-1 сторінка>

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

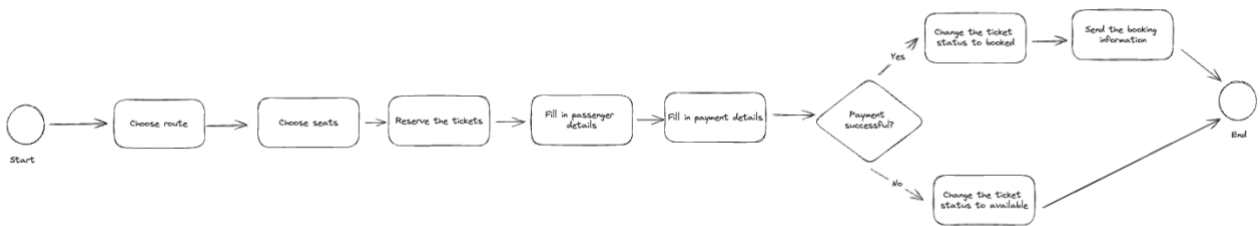


Рисунок 2.1 – Бізнес процес покупки квитків

Опис послідовності покупки квитків

- Користувач обирає необхідний маршрут
- Користувач обирає місця та клас квитків
- Обрані місця резервуються на певний проміжок часу
- Користувач заповнює персональні дані пасажирів
- Користувач заповнює платіжні дані
- Користувач робить оплату
- Якщо оплата успішна
 - Обрані місця стають зайнятими
 - Користувач отримує підтвердження покупки
- Якщо оплата не успішна
 - Обрані місця знову стають незайнятими

2.2 Архітектура програмного забезпечення

Застосунок використовує web клієнт-серверну архітектуру. Для покращення SEO веб застосунку проект будує сторінки на тонкому сервері, або, як його часто називають BFF (Back-end for front-end), який робить запити до API де відбувається уся логіка. Trainly API – незалежна система, яка немає ніякого представлення про користувача, його права, токени, дані і так далі. Всі користувачі – customers. BFF піклується про персональну інформацію, сесії та права. Таким чином ми можемо в майбутньому використовувати trainly api з декількома клієнтами та генерувати sdk на основі open api специфікації. Front-end використовує стандартні технології як HTML, CSS, JavaScript. Для нього немає значної різниці який framework використовувати, ключова ідея – server side rendering для оптимізації для пошукових ботів. Для бази даних було обрано використовувати PostgreSQL. Так як дані мають реляційну структуру, postgres – чудово підходить під цю задачу. Також це одна з найпопулярніших баз даних з широким функціоналом, яка є чудовим універсальним інструментом для багатьох задач. У разі коли її буде не вистачати, ми можемо використовувати якусь іншу спеціалізовану БД. У разі додавання аутентифікації та функціоналу пов'язаного зі зберіганням користувацьких даних, сесій, проект може використати окрему базу даних по типу firebase, dynamo db, sqlite і т.д.

2.3 Конструювання програмного забезпечення

Проект використовує базу даних PostgreSQL та drizzle ORM для data access layer.

Базове представлення структури бази даних (Рисунок 2.3). Тільки основні таблиці

Resend – сервіс для відправки електронних листів

Google Cloud Secret Manager – GCP сервіс, який зберігає application secrets.

Trainly API використовує REST-like архітектуру.

Для завантаження конфігурацій api має 3 джерела. Згідно з <https://github.com/doichev-kostia/config-loader> ми маємо env variables, json config, application secrets.

Env variables – змінні середовища що часто змінюються

JSON config – конфігурація застосунку з не секретною інформацією

Application secrets – секретна інформація, що завантажується з GCP Secret Manager під час запуску

Системні вимоги до trainly api:

- Підтримка docker containers
- 4GB RAM
- 2 vCPU

Системні вимоги до SvelteKit клієнтського застосунку

- 4GB RAM
- 2 vCPU
- Google Chrome вище v100
- Safari вище v15
- Firefox вище v95

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 2.22.

Таблиця 2.22 – Опис утиліт

№ п/п	Назва утиліти	Опис застосування
1	IntelliJ IDEA	Головне середовище розробки програмного забезпечення серверної частини курсової роботи.
2	Data Grip	Середовище доступу до БД

2.4 Аналіз безпеки даних

Застосунок не зберігає великої кількості користувацьких даних. В основному ім'я та прізвище пасажирів та email покупця квитків.

Для безпеки БД проект використовує supabase, ознайомитися з документами можна за посиланням <https://supabase.com/legal/dpa>.

Всі вхідні дані валідуються через валідаційні схеми. ORM піклується про санітизацію sql запитів для уникнення SQL injection.

Всі secrets зберігаються в Google Cloud Secret Manager і встановлюються під час запуску застосунку

Висновки до розділу

Застосунок використовує web клієнт – серверну архітектуру. Trainly API використовує ідеї resource-based architecture. Для клієнту і серверу використовується JavaScript і node.js runtime. Front-end написаний з використанням SvelteKit, back-end використовує fastify. Для бази даних використовується PostgreSQL. Trainly має інтеграцію з декількома third-party сервісами, такими як Stripe, Resend, Supabase, Google Cloud Platform. На основі openapi spec можна згенерувати api client sdks, які різні клієнти можуть використовувати для запитів.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Для аналізу якості пз використовується ручне тестування, лінтери та code review. Аби додати зміни в проект при команді в 2 людини і більше розробник має відкрити pull request, що автоматично, при необхідності, запустить GitHub actions pipelines для перевірки коду. Це можуть бути тести, лінтери та інші інструменти. Далі, розробник має зробити запит від колеги для перевірки змін і після погодження ці зміни можуть заливатися в master гілку

3.2 Опис процесів тестування

Було виконане мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 3.3 – 3.30.

Таблиця 3.3 – Тест 1.1

Тест	Пошук маршруту
Модуль	Пошук маршруту
Номер тесту	1.1
Початковий стан системи	Користувач на головній сторінці
Вхідні данні	Місто відправлення, місто прибуття, дата відправлення
Опис проведення тесту	У select компоненті обирається місто відправлення, у наступному select елементі обирається місто прибуття. В input з датою вводиться дата відбуття
Очікуваний результат	Користувача переносить на сторінку з доступними маршрутами
Фактичний результат	Користувач бачить маршрути

Таблиця 3.4 – Тест 1.2

Тест	Обрання маршруту
Модуль	Обрання маршруту
Номер тесту	1.2
Початковий стан системи	Користувач на сторінці з маршрутами
Вхідні данні	
Опис проведення тесту	Серед списку маршрутів користувач обирає потрібний
Очікуваний результат	Користувач бачить деталі маршруту
Фактичний результат	Користувач бачить деталі маршруту

Таблиця 3.5 – Тест 1.3

Тест	Обрання місця
Модуль	Обрання місця
Номер тесту	1.3
Початковий стан системи	Користувач на сторінці маршруту
Вхідні данні	
Опис проведення тесту	Серед списку сидінь користувач обирає необхідне доступне місце і натискає кнопку continue
Очікуваний результат	Користувач переходить на сторінку з заповненням деталей пасажирів
Фактичний результат	Користувач переходить на сторінку з заповненням деталей пасажирів

Таблиця 3.6 – Тест 1.4

Тест	Заповнення деталей пасажирів
Модуль	Заповнення деталей пасажирів
Номер тесту	1.4
Початковий стан системи	Користувач на сторінці деталей пасажирів
Вхідні данні	Email отримувача квитків, ім'я пасажирів, прізвище пасажирів
Опис проведення тесту	Користувач вводить персональні дані пасажирів та email отримувача квитків. Натискає кнопку checkout
Очікуваний результат	Користувач переходить на сторінку з заповненням платіжної інформації
Фактичний результат	Користувач переходить на сторінку з заповненням платіжної інформації

Таблиця 3.6 – Тест 1.5

Тест	Заповнення платіжної інформації
Модуль	Заповнення платіжної інформації
Номер тесту	1.5
Початковий стан системи	Користувач на сторінці платежу
Вхідні данні	Неправильний номер платіжної карти, card expiry date, cvc, ім'я власника карти
Опис проведення тесту	Користувач вводить неправильні дані карти
Очікуваний результат	Користувач отримує помилку

Фактичний результат	Користувач отримує помилку
---------------------	----------------------------

Таблиця 3.7 – Тест 1.6

Тест	Заповнення платіжної інформації
Модуль	Заповнення платіжної інформації
Номер тесту	1.6
Початковий стан системи	Користувач на сторінці платежу
Вхідні данні	номер платіжної карти, card expiry date, cvc, ім'я власника карти
Опис проведення тесту	Користувач вводить дані карти
Очікуваний результат	Користувач отримує підтвердження оплати та email з інформацією про квитки
Фактичний результат	Користувач отримує підтвердження оплати та email з інформацією про квитки

Таблиця 3.8 – Тест 1.7

Тест	Отримання інформації про квитки
Модуль	Отримання інформації про квитки
Номер тесту	1.7
Початковий стан системи	Користувач отримує email з посиланням на деталі квитків
Вхідні данні	
Опис проведення тесту	Користувач переходить за посиланням

Очікуваний результат	Користувач бачить інформацію про квитки
Фактичний результат	Користувач бачить інформацію про квитки

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Клієнська частина розгортається на платформі Vercel. Серверна частина розгортається на платформі Railway. Весь код зберігається у форматі монорепозіторію у GitHub. Vercel надає можливість встановити GitHub application, яке використовує webhooks для прослуховування подій. При доставлянні нової версії застосунку в GitHub репозиторій, GitHub відправляє подію і Vercel починає деплой коду. Вони запускають процес зборки і при успішному закінченні оновлюють версію, яка на хостингу. Щодо back-end частини, то за допомогою GitHub actions збирається docker image і деплоїться на railway. Тригером є новий реліз застосунку в GitHub.

```

name: deploy

on:
  workflow_dispatch:
  release:
    types: [ created ]

permissions:
  id-token: write
  contents: read

jobs:
  deploy:
    runs-on: ubuntu-latest
    timeout-minutes: 20
    steps:
      - name: Checkout
        uses: actions/checkout@v4

      - uses: pnpm/action-setup@v2

      - name: Use Node.js
        uses: actions/setup-node@v3
        with:
          node-version-file: '.nvmrc'
          cache: 'pnpm'

      - run: pnpm add -g @railway/cli

      - id: auth
        name: Authenticate to Google Cloud
        uses: google-github-actions/auth@v1
        with:
          workload_identity_provider: ${ secrets.IDENTITY_PROVIDER }
          service_account: ${ secrets.SERVICE_ACCOUNT }

      - name: 'Set up Cloud SDK'
        uses: 'google-github-actions/setup-gcloud@v1'

      - name: Set credentials
        run: gcloud auth login --cred-file=${ steps.auth.outputs.credentials_file_path }

      - name: Pull service account key
        run: |
          gcloud secrets versions access latest --project=${ secrets.GCP_PROJECT_ID } --secret=${ secrets.SERVICE_ACCOUNT_SECRET_NAME } > ./apps/api/service-account.json

      - name: Remove the service account key from the gitignore file to allow railway to access it
        run: |
          sed -i '/service-account.json/d' ./gitignore

      - name: Deploy
        run: railway up --service trainly --environment production -d
        env:
          RAILWAY_TOKEN: ${ secrets.RAILWAY_TOKEN }

```

Рисунок 4.1 – Інструкція для деплою back-end частини

4.2 Підтримка програмного забезпечення

Для підтримки програмного забезпечення розробнику потрібно зклонувати GitHub репозиторій проекту, встановити весію node.js яка вказана в .nvmrc та використати пакетний менеджер (package manager) вказаний в package.json. Для внесення змін розробник має зробити нову git гілку та після завершення завдання відкрити pull request для пропозиції змін в головну гілку master. При злитті vercel bot задеплойть клієнтську частину застосунку. При release GitHub actions зберуть docker image та відправлять його на Railway для подальшого розгортання.

ВИСНОВКИ

У висновках викладають найважливіші наукові й практичні результати роботи та наводять:

- оцінку одержаних результатів і їх відповідність сучасному рівню наукових і технічних знань;
- ступінь впровадження та можливі галузі або сфери використання результатів роботи;
- наукову, науково-технічну, соціально-економічну значущість роботи;
- доцільність продовження досліджень за відповідною тематикою тощо.

Також у висновках необхідно відобразити стан вирішення усіх поставлених в курсовій роботі задач.

В результаті виконання курсової роботи було спроектовано ...

В якості середовища розробки обрано ...

У якості БД використано ...

Після реалізації застосунку він був протестований на пристроях з різними версіями Android, з різними розмірами екранів щоб переконатися, що додаток акуратно відображається на різних пристроях.

Наукова новизна роботи (якщо вона є) полягає в наступному (достатньо вказати щось одне).

Вперше:

- реалізовано можливість запитів від пацієнта до лікаря;
- використано те-то, що дозволило те-то.

Модифіковано:

- те-то, що дозволило те-то.

Набуло подальший розвиток:

- те-то, що дозволило те-то.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- 1) Залічнийний транспорт [Інтернет ресурс] – [посилання](#)