

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ
ВІДСЛІДКОВУВАННЯ ДОСТАВКИ ЗАМОВЛЕНЬ КУР'ЄРСЬКОЮ
СЛУЖБОЮ

RESEARCH AND SOFTWARE IMPLEMENTATION OF THE SYSTEM FOR
TRACKING THE COURIER DELIVERY

Виконав: студент 2 курсу, групи КНТ-110м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Терещенко Р.М.

(прізвище та ініціали)

Керівник Степаненко О. О.

(прізвище та ініціали)

Рецензент Казурова А. Є.

(прізвище та ініціали)

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

Інститут, факультет ІРЕ, ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти магістр
 Спеціальність 121 Інженерія програмного забезпечення
 (код і найменування)

Освітня програма (спеціалізація) Інженерія програмного забезпечення
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 20 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Терещенка Руслана
Максимовича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація системи відслідковування доставки замовлень кур'єрською службою. Research and software implementation of the system for tracking the courier delivery.
 керівник проєкту (роботи) Степаненко Олександр Олексійович, к.т.н., доцент,

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

затверджені наказом закладу вищої освіти від “05” листопада 2021 року
 № 435

2. Строк подання студентом проєкту (роботи) «17» листопада 2021 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Вибір та обґрунтування системи, яка проектується. 3. Керівництво програміста. 4. Експлуатація, тестування та експериментальне дослідження програми. 5. Текст програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-6 Основна частина	Степаненко О.О., доцент		
Нормоконтроль	Дейнега Л.Ю., ст. викладач		

7. Дата видачі завдання “02” вересня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2-4 тижні	Розділ 1
3	Вибір засобів розробки програмного продукту	5 тиждень	Розділ 2
4	Розробка архітектури програми	6-9 тижні	Розділ 2
5	Розробка програмного продукту	10-14 тижні	Розділ 3
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	15 тиждень	Додатки
7	Захист роботи	16 тиждень	

Студент(ка)

(підпис)

Терещенко Р. М.

(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Степаненко О.О.

(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
121 с., 19 табл., 30 рис., 2 дод., 16 джерел.

ДОСТАВКА, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ЛОГІСТИКА,
КОНТРОЛЬ ЧАСУ, КУР'ЄР.

Об'єкт дослідження – розрахунок оптимального маршруту доставки замовлень кур'єрською службою та контроль часу та коректності місця доставки.

Предмет дослідження – програмні системи керування логістикою служб доставки.

Мета роботи – створення програмної системи для розрахунку оптимального маршруту доставки замовлень кур'єрською службою та мобільного додатку для контролю коректності часу та місцезнаходження доставки замовлення.

Матеріали, методи та технічні засоби: мова програмування PHP, мови розмітки HTML та CSS, мова структурованих запитів SQL, мова програмування Java, персональний комп'ютер з процесором Intel Core i5 під управлінням операційної системи macOS 12.

Результати. Розроблено систему запитів API для роботи з мобільними пристроями, веб-інтерфейс керування системою, а також мобільний додаток для системи Android.

Висновки. Розроблено програмну систему для пошуку оптимальних маршрутів та фіксації фактично виконаної доставки.

Галузь використання – розрахунок оптимального маршруту та контроль кур'єрів на підприємствах кур'єрських доставок.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 121 pages, 30 figures, 19 tables, 2 appendixes, 16 sources.

DELIVERY, SOFTWARE, LOGISTICS, TIME CONTROL, COURIER.

The object of research is the calculation of the optimal route of delivery of orders by courier service and control of time and correctness of the place of delivery.

The subject of research - software management systems for logistics of delivery services.

The purpose of the work is to create a software system for calculating the optimal route for delivery of orders by courier service and a mobile application to control the correctness of the time and location of order delivery.

Materials, methods and technical tools: PHP programming language, HTML and CSS markup languages, SQL structured query language, Java programming language, personal computer with Intel Core i5 processor running macOS 12 operating system.

Results. Developed an API query system for mobile devices, a web-based system management interface, as well as a mobile application for Android.

Conclusions. A software system has been developed to find the optimal routes and record the actual delivery.

Area of use - calculation of the optimal route and control of couriers at courier companies.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК	8
ВСТУП	9
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ	11
1.1 Аналіз проблеми пошуку оптимального маршруту	11
1.1.1 Методи вирішення проблеми, що використовуються на даний момент	11
1.2 Огляд аналогів	12
1.2.1 Google Maps	12
1.2.2 C2Logix.....	12
1.2.3 Badger	13
1.3 Постановка завдань дослідження	13
1.4 Висновки за розділом	14
2 ВИБІР ТА ОБҐРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ, ЯКА ПРОЕКТУЄТЬСЯ.....	15
2.1 Огляд середовища розробки	15
2.2 Архітектура БД.....	22
2.3 Архітектура програми.....	26
2.4 Логіка роботи системи.....	29
2.4 Висновки за розділом	35
3 КЕРІВНИЦТВО ПРОГРАМІСТА	36
3.1 Реалізація бази даних.....	36
3.2 Реалізація серверної частини системи	40
3.2.1 Контролери	40

3.2.2 Моделі	42
3.2.3 Представлення	46
3.2.4 Дизайн	56
3.2.5 Маршрутизація	57
3.2.6 Підключення до бази даних	59
3.3 Розробка мобільного додатку	59
3.3.1 Контролери	59
3.4 Висновки за розділом	65
4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ	66
4.1 Опис застосування	66
4.2 Методика та результати тестування	66
4.3 Експериментально отримані результати	66
4.4 Висновки за розділом	67
ВИСНОВКИ	68
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ	69
ДОДАТОК А Текст програми	71
ДОДАТОК Б Слайди презентації	118

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
FTP	– File Transfer Protocol;
GPS	– Global Positioning System
HTTP	– Hyper Text Transfer Protocol;
IDE	– Integrated Development Environment;
JSON	– JavaScript Object Notation;
MVC	– Model-View-Controller;
URL	– Uniform Resource Locator;
Wi-Fi	– Wireless Fidelity;
БД	– база даних;
СУБД	– система управління базами даних.

ВСТУП

Проблема – на даний момент на підприємствах що надають послуги з кур'єрських доставок виникає потреба у своєчасній доставці замовлень. Несвоєчасна доставка може бути спричинена тим, що неможливо точно розрахувати час прибуття кур'єра до клієнта, а також зафіксувати чи було замовлення видане своєчасно.

Програмна реалізація системи для вирішення такої проблеми має багато переваг. При наявності системи автоматичного розрахунку дистанції між точкою отримання замовлення кур'єром та точкою отримання замовлення клієнтом виникає можливість прогнозування точного часу доставки. За допомогою мобільного додатку для кур'єра з можливістю фіксації часу доставки логіст служби доставки має можливість перевірити, чи вчасно було доставлено замовлення. Також за допомогою розрахунку часу проведеного кур'єром в дорозі є можливість прогнозувати час прибуття кур'єра назад на точку отримання замовлення, для подальшого прогнозування наступних маршрутів з урахуванням часу відправлення.

Було вирішено розробити програмну реалізацію системи розрахунку оптимального маршруту та фіксації фактично виконаної доставки замовлення.

Матеріали, методи та технічні засоби – для реалізації системи було використано:

- мову програмування PHP для розробки API для обміну даними з мобільним додатком;
- мову запитів SQL для роботи з базою даних MySQL;
- мову розмітки сторінки HTML для розробки візуального інтерфейсу користувача для роботи логіста;
- мову стилів CSS для стилізації візуального інтерфейсу користувача для WEB-інтерфейсу логіста;

- мову програмування Java розробки мобільного додатку для фіксації доставки на платформі Android.

Апаратні засоби, що було використано для реалізації системи:

- Apple MacBook Air (7,2) – процесор Dual-Core Intel Core i5 1,8 GHz, пам'ять 8 ГБ, графіка Intel HD Graphics 6000 1536 МБ.

- Програмні засоби, що було використано для реалізації системи:

- PhpStorm – програмне забезпечення компанії JetBrains, середовище розробки PHP;

- Android Studio – програмне забезпечення компанії Google, інтегроване середовище розробки Android-додатків;

- Postman – програмне забезпечення для тестування запитів API.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз проблеми пошуку оптимального маршруту

У сучасному світі швидко розвиваються технології, пов'язані з кур'єрськими доставками. Кур'єрськими службами користуються люди, яким потребується доставка їжі та товарів з магазинів. Початок розвитку служб доставок можна побачити на прикладі поштових компаній-перевізників, що давно надавали послуги з кур'єрських доставок. Останнім часом багато компаній надають послуги доставки як їжі з ресторанів, так і будь-чого іншого. В реаліях пандемії коронавірусної інфекції очевидне збільшення попиту на доставки, який виникає через карантинні обмеження. Для служб доставки дуже важливо знизити час та вартість доставки, для збільшення доходу компанії.

1.1.1 Методи вирішення проблеми, що використовуються на даний момент

На підприємствах, що надають послуги з кур'єрської доставки, проблеми розрахунку оптимального порядку доставки та фіксації доставки вирішуються такими методами:

- розрахунок оптимального маршруту за допомогою сервісу Google Maps. Метод дозволяє з високою точністю розрахувати дистанцію та час доставки, але за наявності декількох точок доставки в маршруті потребує ручної перестановки точок для знаходження оптимального маршруту. Також сервіс не дозволяє додавати в один маршрут більше 7 точок;

- спеціалізовані сервіси з оптимального розрахунку маршрутів дозволяють вирішувати таку проблему. Але проблема фіксації фактично виконаної доставки залишається актуальною, так як у таких сервісах зазвичай використовується мобільний додаток для кур'єрів з відстеженням

місцезнаходження в реальному часі, що призводить до швидкого розряджання акумуляторних батарей та як наслідок - до деградації акумуляторів пристроїв [1].

Перший метод не дає достатньої достовірності та призводить до додаткових витрат часу. Системи автоматичного розрахунку надають можливість швидкого розрахунку, але при цьому не надають достатньо зручного методу контролю доставки кур'єром, при цьому такі системи автоматичного розрахунку мають фіксовану за місяць вартість, яка може перевищувати реальні потреби компанії.

Таким чином метою роботи було визначено розробку програмного засобу для автоматичного розрахунку оптимального маршруту.

1.2 Огляд аналогів

1.2.1 Google Maps

Компанія Google надає можливість розрахунку оптимального маршруту руху по заданим точкам [2]. Сервіс безкоштовний для використання. Підтримується до 7 точок в одному маршруті. Сервіс з достатньо високою точністю розраховує дистанцію шляху та час в дорозі, але не надає можливості автоматичного формування порядку точок у маршруті. Також відсутня можливість фіксації фактично виконанох доставки.

Підтримка платформ:

- WEB-інтерфейс;
- Android;
- iOS.

1.2.2 C2Logix

Веб-рішення, яке можна використовувати з будь-якого комп'ютера з підключенням до Інтернету. C2Routing дає можливість маршрутизувати один або багато польових ресурсів на наступний день або на тижні вперед. За

допомогою понад 100 різних налаштувань параметрів можна налаштувати свої маршрути відповідно до бізнес-операцій саме так, як ви хочете [3].

Оскільки немає обмежень на кількість разів, які ви можете оптимізувати, можна використовувати C2Routing як інструмент маршрутизації та інструмент планування для виконання різних бізнес-сценаріїв, щоб побачити, що найкраще працюватиме для діяльності компанії.

Ключові переваги:

- не потребує програмного забезпечення для встановлення;
- інтерфейс Планувальника маршрутів, спеціально розроблений, щоб бути інтуїтивно зрозумілим і простим у використанні;
- створено для користувачів, які не володіють комп'ютером.
- Недоліки:
- не надається можливість фіксації фастично виконаної доставки.

1.2.3 Badger

Платформи: Web, iOS, Android, Mac OS, Windows Desktop.

Надає можливість розрахунку оптимального шляху за заданими точками. Завдяки підтримці мобільних додатків пропонує зручно виконувати поставлені логісту задачі на ходу. При цьому, дане ПЗ пропонує обирати точки для маршруту, що може бути не зручно, так як логіст не завжди може оптимально розподілити точки за маршрутами [4].

Badger пропонує відслідковування кур'єрів у реальному часі, що достатньо зручно, але може впливати на стан акумуляторів пристроїв.

1.3 Постановка завдань дослідження

Завданням дослідження було обрано методи оптимізації маршрутів та аналіз методів зниження затрат для компаній, що надають послуги з кур'єрських доставок. За результатом аналізу методів оптимізації та програмної реалізації існуючих методів було розроблено модифікацію

методу пошуку оптимального маршруту, та програмні продукти для контролю роботи служби доставки, з використанням даного алгоритму.

1.4 Висновки за розділом

Було розглянуто аналоги розроблюваної системи, та визначено, що аналоги або недостатньо зручні у використанні, або мають простір до покращення. Тому вирішено розробити систему, яка буде розраховувати оптимальний маршрут та контролювати фактично виконану кур'єром доставку.

2 ВИБІР ТА ОБҐРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ, ЯКА ПРОЕКТУЄТЬСЯ

Розробку можна поділити на такі логічні частини:

- проектування та створення структури БД;
- проектування архітектури програми.

2.1 Огляд середовища розробки

Для розробки серверної частини продукту було обрано мову програмування PHP. Цей вибір обґрунтовується тим, що ця мова програмування надає всі необхідні можливості для роботи серверної частини системи, а конкретно:

- зручна взаємодія з базами даних;
- виконання коду на сервері;
- можливість звертатися до програми шляхом HTTP-запитів для обміну даними між клієнтською частиною та серверною;
- можливість розробки візуального інтерфейсу користувача на мові розмітки сторінки HTML з використанням вставок коду на PHP.

Для розробки серверної частини програмного продукту було обрано інтегроване середовище розробки PhpStorm від компанії JetBrains. На рисунку 2.1 показано як виглядає робочий простір програми.

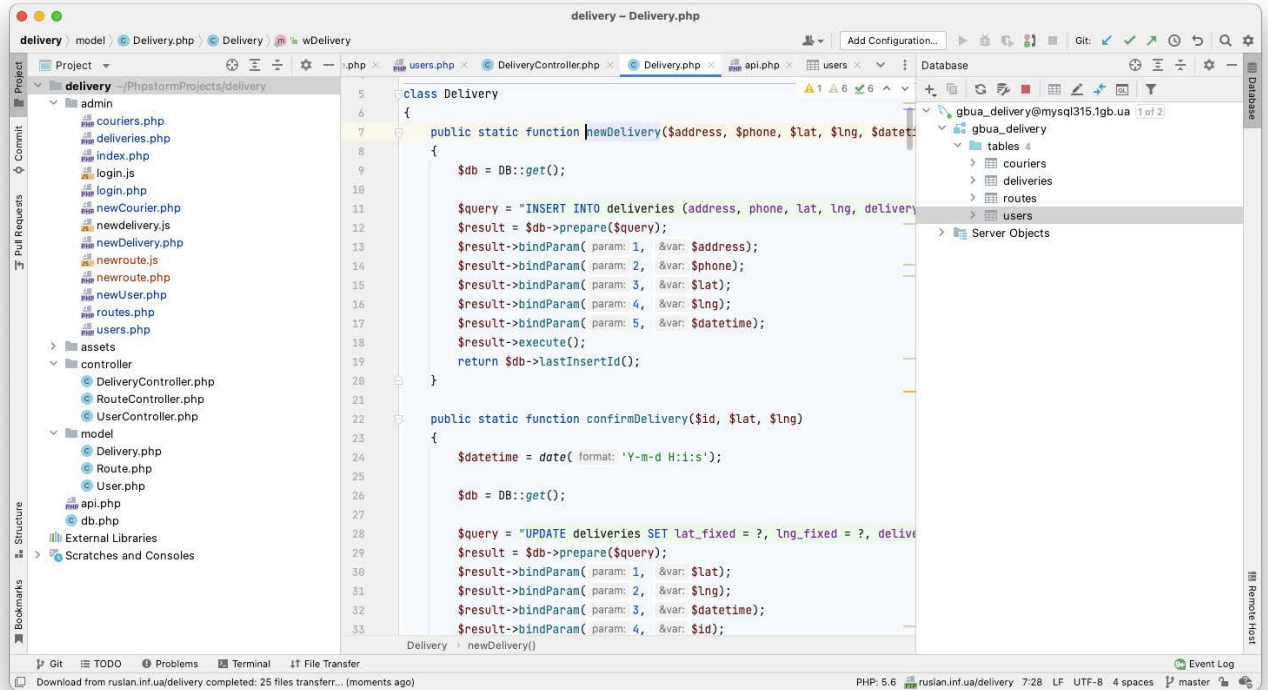


Рисунок 2.1 – Знімок екрану робочого простору програми PhpStorm

PhpStorm – програмне забезпечення від компанії JetBrains. Воно являє собою інтегровану середу розробки, що надає можливість розробки web-застосунків та інших видів програм, які можна розробляти за допомогою мови програмування PHP та з використанням таких мов програмування як HTML, JavaScript і CSS. Середа розробки PhpStorm надає можливості автоматичного завершення мовних конструкцій коду PHP, інспектування коду, різноманітні алгоритми рефакторингу та швидку навігацію по коду [5].

Реалізований в PhpStorm графічний PHP відладчик підтримує умовні точки зупину, відстеження значень і автоматизований вхід у налагодження окремих процедур. Для тестування застосунків підтримується каркас тестових модулів PHPUnit і графічний інтерфейс для запуску тестів. При редагуванні коду виділяються конструкції синтаксису, здійснюється розширене форматування конфігурації, виявлення помилок в режимі реального часу і завершення коду. PhpStorm-редактор враховує коментарі до коду при його завершенні, автоматично вибираючи оптимальне рішення

проблеми. PHP-рефакторинг і редагування шаблонів гарантує зміна проекту в найкоротші терміни. PhpStorm дозволяє візуалізувати код в ієрархічному вигляді і забезпечує швидку навігацію по всім елементам.

Завдяки застосуванню PHPUnit-тестів можна швидко переглядати результати генерації коду окремих блоків або всієї програми. Якщо тест був проведений невдало, продукт дозволяє переглядати окремі кодові рядки, в яких була виявлена помилка. PhpStorm забезпечує налагодження коду JavaScript і надає широкий діапазон можливостей: знаходження точки зупину в HTML і JavaScript, налаштування параметрів точки зупину, тестування синтаксису коду в режимі реального часу і т. д.

Сотні інспекцій піклуються про верифікації коду, аналізуючи проект цілком під час розробки. Підтримка PHPDoc, code (re)arranger, форматтера коду з конфігурацією стилю коду та інші можливості допомагають розробникам писати охайний і легко-підтримуваний код.

Підтримуються передові технології веб-розробки, включаючи HTML5, CSS, Sass, SCSS, Less, Stylus, Compass, CoffeeScript, TypeScript, ECMAScript Harmony, шаблони Jade, Zen Coding, Emmet, і, звичайно ж, JavaScript [6].

PhpStorm включає в себе всю функціональність JetBrains WebStorm (HTML/CSS редактор, редактор JavaScript) і додає повнофункціональну підтримку PHP і баз даних SQL завдяки інтеграції JetBrains DataGrip (програма для роботи з базами даних) [7].

Нова технологія JetBrains CodeWithMe, що також інтегрована в PhpStorm, дозволяє отримувати віддалений доступ до проекту іншим користувачам, наприклад існує можливість підключитись до відкритого проекту PhpStorm у режимі спостереження з підсвічуванням поточного розташування курсора ініціатора підключення, а також у режимі спільного редагування, що дозволяє одночасно редагувати файли декількома користувачами з відображенням змін у реальному часі. Плагін дозволяє ефективно співпрацювати у дистанційному режимі, що достатньо актуально в умовах дистанційної роботи багатьох підприємств [8].

Середовище розробки підтримує системи контролю версій, такі як GitHub, BitBucket, Subversion та інші. Для розміщення проектів на віддаленому сервері існує можливість підключення до сервера за допомогою FTP.

Ключові можливості PhpStorm:

- інтелектуальний редактор PHP коду з підсвічуванням синтаксису, автодоповненням коду, розширеними налаштуваннями форматування коду, запобіганням помилок;
- підтримує PHP 7.0, 5.6, 5.5, 5.4 та 5.3, генератори, співпрограми і все синтаксичні поліпшення;
- PHP рефакторинг, code (re) arranger, детектор дубльованого коду;
- підтримка Vagrant, Composer, вбудований REST клієнт, Command Line Tools, SSH консоль;
- підтримка фреймворків (MVC view для Symfony2, Yii) і спеціалізовані плагіни для провідних PHP фреймворків (Symfony, Magento, Drupal, Yii, CakePHP і багато інших);
- візуальний відладчик для PHP застосунків, валідація конфігурації відладчика, PHPUnit з покриттям коду (підтримка PHPUnit 5), а також інтеграція з профілювальником;
- HTML, CSS, JavaScript редактор. Налагодження і модульне тестування для JS. Підтримка HTML5, CSS, Sass, SCSS, Less, Stylus, Compass, CoffeeScript, TypeScript, ECMAScript Harmony, Emmet і інших передових технологій веб-розробки;
- повний набір інструментів для фронтенд-розробки;
- підтримка стилів коду, вбудовані стилі PSR1 / PSR2, Symfony2, Zend, Drupal та інші;
- інтеграція з системами управління версіями, включаючи уніфікований інтерфейс;
- віддалене розгортання застосунків і автоматична синхронізація з використанням FTP, SFTP, FTPS і ін.;

- Live Edit: зміни в коді можна миттєво переглянути в браузері без перезавантаження сторінки;
- PHP UML;
- інтеграція з баг-трекерами;
- інструменти роботи з базами даних, SQL-редактор;
- крос-платформність (PhpStorm доступний на Windows, Mac OS X, Linux) [6].

Для розробки мобільного додатку для фіксації факту доставки кур'єром було обрано платформу Android та інтегроване середовище розробки Android Studio. Операційна система Android широко розповсюджена серед користувачів мобільних пристроїв та популярна завдяки тому, що встановлюється на більшість виготовлених смартфонів. Інтегроване середовище розробки Android Studio розповсюджується компанією Google та позиціонується як офіційне інтегроване середовище розробки для платформи Android. Візуальний редактор розмітки дозволяє проектувати інтерфейс користувача простим перетягуванням елементів на полотно. Елементи можна розташовувати як це необхідно, та змінювати параметри, такі як розмір, колір, текст та інше. Для розташування елементів використовується один з декількох контейнерів Layout, які відрізняються принципом розташування елементів всередині. Також редактор розмітки підтримує редагування коду XML, що дозволяє швидко копіювати елементи розмітки з одного файлу в інший. Android Studio комплектується програмним забезпеченням Android Emulator, яке дозволяє запускати на комп'ютері копію операційної системи Android для тестування готового програмного продукту. Надається можливість обрати діагональ екрану емульованого пристрою, а також версію операційної системи, для тестування функцій розроблюваного програмного забезпечення в умовах підтримки різними версіями операційної системи. Технологія ADB (скорочено від Android Debug Bridge) дозволяє запускати розроблюване програмне забезпечення не реальному пристрої шляхом під'єднання його за допомогою кабеля USB. Також, починаючи з Android 11 існує вбудована можливість підключатися до відладчика шляхом

бездротового з'єднання за умов розташування в одній мережі Wi-Fi. Середовище розробки Android Studio орієнтоване на роботу з мовами програмування Java, Kotlin, XML. Для роботи сторонніх бібліотек в Android-додатках найчастіше використовують файл залежностей на мові Gradle, Android Studio також має підтримку цієї мови. Середовище розробки надає можливість використовувати підказки під час написання коду програми, а також пропонує можливості переходу до функцій, пошуку дубльованих фрагментів коду та інше. Під час відладки розроблюваного мобільного додатку можна в реальному часі відстежувати такі показники, як навантаження на центральний процесор, використання оперативної пам'яті, використання мережі. Одна з нових функцій Android Studio – відображення вмісту локальної бази даних SQLite, що дозволяє перевіряти коректність зберігання даних під час роботи додатку та змінювати такі дані при необхідності. Також доступний режим перегляду у реальному часі, без необхідності оновлення, у такому разі можливість редагування даних вимикається. Для Android Studio пропонується велика кількість сторонніх плагінів, що дозволяє суттєво розширити функціональність середовища розробки. Інтеграція з акаунтом Google дозволяє швидко керувати хмарними можливостями додатку, такими як push-повідомлення, сервіси Google Maps та інше.

У таблиці 2.1 можна побачити порівняння основних мов програмування які використовуються для виконання коду на сервері. Можна побачити, що PHP виграє по показникам швидкості роботи та підключення до баз даних, що і стало причиною вибору цієї мови програмування.

Таблиця 2.1 – Порівняння мов програмування для серверної частини програми

Параметри	PHP	Python
1	2	3

Легкість встановлення	+	-
Швидкість роботи	+	-
Підтримка більшої кількості бібліотек	-	+

Продовження таблиці 2.1

1	2	3
Просте підключення до баз даних	+	-
Захищеність	-	+

У таблиці 2.2 можна побачити порівняння основних мов програмування для платформи Android, таких як Java і Kotlin. Саме наявність більшої кількості бібліотек та документації вплинуло на вибір мови програмування.

Таблиця 2.2 – Порівняння мов програмування для мобільного додатку на платформу Android

Параметри	Java	Kotlin
Поширеність серед розробників	+	-
Простота коду	-	+
Підтримка більшої кількості бібліотек	+	-

2.2 Архітектура БД

База даних повинна зберігати у собі такі дані:

- дані користувачів, а саме логін та пароль для входу в систему логістів, та місцезнаходження точки відправлення для кожного конкретного логіста;
- дані кур'єрів, а саме ім'я, пін-код для авторизації в додатку для фіксації доставки, та показник витрати пального;
- дані маршрутів, а саме дата і час створення, дані про логіста, що створив маршрут, дані про кур'єра що був обраний для виконання доставки по маршруту, статус маршруту а також дистанція маршруту;
- дані про доставку, а саме час доставки, адреса, контактний номер телефону для зв'язку з отримувачем, широта та довгота точки доставки, фактичний час доставки та широта і довгота фактичного місцезнаходження точки доставки.

При виборі СУБД важливо розуміти різницю між двома фундаментально різними типами баз даних - реляційною та нереляційною. Реляційна база даних зберігає у собі дані у вигляді таблиці з стовпцями та рядками, та дозволяє звертатися до даних шляхом виконання SQL-запиту. Нереляційна база даних зберігає у собі дані у вигляді файлів, наприклад формату JSON. Ключова перевага таких баз даних порівняно з реляційними - можливість зберігання даних з різними полями та атрибутами. При цьому реляційні бази даних дозволяють бачити збережені дані у зручному для розробника форматі, а також з легкістю пов'язувати між собою дані з різних таблиць. Для реалізації бази даних для системи було обрано СУБД MySQL. MySQL на даний момент є найбільш популярною СУБД.

SQL (скорочено від Structured Query Language) — мова структурованих запитів, що використовується для управління даними в реляційних базах даних. За допомогою запитів на мові SQL можна виконати такі дії, як створення, модифікація та видалення таблиці з бази даних,

додання нового запису в таблицю, змінення вже існуючого запису, видалення записів, та отримання записів з можливістю фільтрації результатів на етапі отримання. Також оператор JOIN дозволяє отримувати дані з різних таблиць, пов'язуючи їх за певними параметрами. Наприклад, у нашому випадку, в таблиці кур'єрів поле `id` пов'язане з полем `courier_id` з таблиці доставок, що дозволяє за допомогою оператора INNER JOIN отримати дані про маршрут, включивши в результат також дані про кур'єра.

Дані, отримані шляхом виконання SQL-запиту методами мови PHP можна з легкістю конвернувати в формати JSON, XML, CSV або XSL для вивантаження даних з метою подальшого аналізу.

Для передачі даних методом API найчастіше використовується формат даних JSON.

JSON (скорочено від JavaScript Object Notation) — формату передачі даних. JSON доступний для використання на багатьох мовах, включаючи в тому числі Python, Ruby, PHP і Java.

Файли що використовують формат даних JSON мають розширення `.json`. Для об'явлення об'єкту JSON у коді програми дані зберігаються у рядку, що може бути визначений як змінна в коді, така змінна буде мати тип `string`. У такому форматі дані легко передавати шляхом HTTP-запитів між сервером та клієнтом.

Об'єкт JSON – це формат даних типу «ключ-значення», який зазвичай зберігається в фігурних дужках.

Приклад найпростішого об'єкту JSON:

```
{  
  "name" : «Name»,  
  "surname" : «Surname»,  
  "address" : «Address»,  
  "status" : 1  
}
```

Ключові параметри об'єкту JSON: початок та кінець об'єкту позначається фігурними дужками, назви ключів об'єкту беруться в лапки.

Ключ та значення розділені двокрапкою, як наприклад тут "key" : "value". Кожна пара ключ-значення розділена комою, таким чином основна частина JSON виглядає так: "key" : "value", "key" : "value", "key" : "value". У прикладі, наведеному вище, перша пара ключ-значення – це "name" : "Name".

Ключі в JSON знаходяться з лівого боку від двокрапки. У кожному об'єкті ключі мають бути унікальними. Ключі можуть містити будь-які символи, але деякі символи, наприклад лапки, зворотня коса лінія та спеціальні символи, наприклад символ переносу рядка \n мають бути екрановані для коректного парсингу даних з об'єкту JSON.

Значення в JSON знаходяться з правого боку від двокрапки. Значення можуть бути рядком, цілим числом, числом з плаваючою крапкою, булевим значенням (так/ні), null, ще одним об'єктом типу JSON або масивом з даними з вище перерахованих типів даних.

Кожен тип даних, який передається як значення в JSON буде підтримувати свій власний синтаксис, так що рядки виводяться в лапках, а цифри – ні.

Для роботи з JSON немає потреби у переносі пар ключ-значення на окремі рядки, але зазвичай об'єкти форматуються у такий вид для спрощення візуального відображення для програміста.

При роботі з JSON-рядками в коді програми переноси рядків зазвичай не використовуються.

Для коректної роботи JSON не принципіальна наявність будь-якої кількості пробілів, тому для покращення читабельності людиною зазвичай використовують пробіл після двокрапки, що розділяє ключ та значення:

```
{  
  "name": "Name",  
  "surname": "Surname",  
  "status": 1  
}
```

JSON може бути ієрархічним і складним, якщо він включає в себе вкладені об'єкти та масиви.

JSON може містити інші вкладені об'єкти в JSON, на додаток до вкладених масивів. Такі об'єкти та масиви будуть передаватися, як призначені значення ключів і будуть представляти собою зв'язку ключ-значення.

У прикладі нижче для кожного з трьох записів ("employee1", "employee2", "employee3") є вкладений JSON об'єкт, який зберігає у собі значення параметрів для кожного з об'єктів, зі своїми власними вкладеними ключами "name" і "surname". Приклад:

```
{
  "employee1" : {
    "name" : "Name1",
    "surname" : "Surname1",
    "status" : 1
  },
  "employee2" : {
    "name" : "Name2",
    "surname" : "Surname2",
    "status" : 2
  },
  "employee3" : {
    "name" : "Name3",
    "surname" : "Surname3",
    "status" : 3
  }
}
```

У прикладі вище можна побачити, що фігурні дужки використовуються для обмеження вкладеного JSON-об'єкта з асоційованими назвами працівників та даними кожного з них, такими як ім'я, прізвище та поточний статус. Як з будь-якими іншими значеннями, використовуючи об'єкти, кома використовується для розділення елементів.

Для роботи з масивами даних використовуються як і в інших мовах програмування квадратні дужки, що позначають початок та кінець масиву. Об'єкти в масиві розділені між собою комою.

Порівняння JSON з XML: XML (скорочено від Extensible Markup Language) – це ще один з популярних форматів даних. Формат даних XML також доступний для використання в багатьох мовах програмування.

Фундаментально принцип зберігання даних у форматі XML дуже схожий на принцип JSON, але JSON наприклад у мові JavaScript не потребує додатково парсингу, на відміну від XML. Також на відміну від JSON, XML може зберігати у собі дані лише типу «рядок».

Так як XML є мовою розмітки, то можна знайти багато схожого з мовою розмітки сторінки HTML, наприклад такі самі відкриваючі та закриваючі теги, що позначаються символом трикутної дужки.

2.3 Архітектура програми

При розробці серверної частини коду програми було використано архітектурний патерн MVC (рис. 2.2). Концепція полягає у розділенні коду програми на три основні логічні компоненти: модель, представлення та контролер [10],[11]. MVC відокремлює бізнес-логіку та рівень відображення одне від одного. Цей патерн розробки традиційно використовувався для настільних графічних інтерфейсів користувача. Нині архітектура MVC у веб-технологіях стала популярною для розробки веб-додатків, а також мобільних додатків.

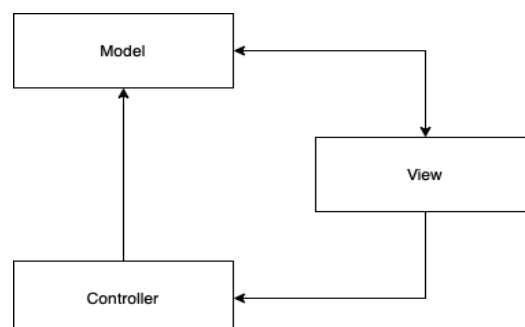
Модель – компонент, що зберігає дані та пов'язану з ними логіку. Він представляє дані, які передаються між компонентами контролера або будь-якою іншою пов'язаною бізнес-логікою. Модель буде отримувати інформацію про клієнта з бази даних. Вона маніпулює даними та надсилає назад до бази даних або використовує її для відтворення тих самих даних. Модель відповідає на запит від представлень, а також відповідає на

інструкції контролера щодо оновлення. Це також найнижчий рівень шаблону, який відповідає за збереження даних.

Контролер – це та частина програми, яка обробляє взаємодію користувача. Контролер інтерпретує введення миші та клавіатури від користувача, інформуючи про те, щоб модель і вигляд змінювалися відповідно до необхідності. Контролер надсилає команди моделі для оновлення її стану (наприклад, збереження певного документа). Контролер також надсилає команди до пов'язаного з ним представлення, щоб змінити вигляд представлення (наприклад, прокручування певного документа).

Представлення – це частина програмного коду продукту, яка представляє подання даних. Представлення створюються на основі даних, зібраних з даних моделі. Представлення запитує модель надати інформацію, щоб вона повторно надіслала вихідну презентацію користувачеві. Представлення також представляє дані з діаграм, графіків і таблиць. Наприклад, будь-яке представлення клієнта включатиме всі компоненти інтерфейсу користувача, як-от текстові поля, випадаючі меню тощо.

Для розробки мобільного додатку на платформі Android було використано схожу концепцію (рис. 2.3). Існує багато різноманітних інтерпретацій розуміння архітектурного патерну MVC саме для платформи Android. Найбільш коректна інтерпретація – файли розмітки XML, що використовуються для розробки візуального інтерфейсу користувача, представляють в архітектурі MVC модуль представлення. Класи-спадкоємці Activity відповідають одночасно контролеру та моделі, так як відповідають за



взаємодію з представленням, та включають в себе бізнес-логіку програми.

Рисунок 2.2 – Архітектура серверної частини програми

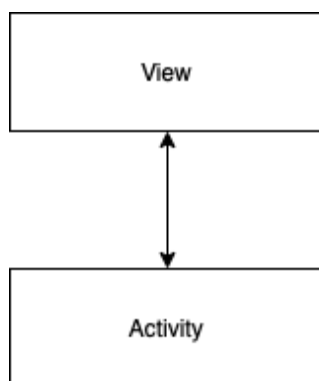


Рисунок 2.3 – Архітектура мобільного додатку

2.4 Логіка роботи системи

Першочергова задача при розробці системи – отримання у результаті розробки систему, що дозволяє додавати в базу даних замовлення, що мають бути доставлені клієнтам, з подальшим автоматичним формуванням маршрутів доставки, зважаючи на кількість вільних кур'єрів та на їх характеристики, знаходячи оптимальний, як з логістичної, так і з економічної точки зору, маршрут. Задача зі знаходження найменшої довжини шляху для декількох заданих точок відома як «задача комівояжера». В реалізації системи для служби доставки задаємо вимогу, що кур'єр служби доставки може в одному маршруті доставляти як одне, так і декілька замовлень, а також те, що після доставки всіх наявних замовлень кур'єр має повернутися до точки видачі, для продовження роботи з наступними доставками у наступному маршруті. Для початку розглянемо основні алгоритми пошуку оптимального маршруту, що існують на даний момент

Найбільшу точність має метод повного перебору. Для кожних двох точок розраховується відстань між ними, і потім послідовно, починаючи з точки отримання замовлення, будується маршрут за принципом того, що наступна точка буде та, відстань до якої є найменшою. Такий алгоритм є найбільш надійним та у кінцевому результаті знайде маршрут, максимально

наближений до оптимального з точки зору дистанції, що проїде кур'єр, але недолік цього алгоритму полягає у надзвичайно великій кількості ітерацій циклу, що буде перебирати точки. Цей алгоритм є найменш оптимальним з точки зору часу виконання, тому при рішенні практичних задач він не використовується.

Інший метод що використовується для вирішення задачі – метод мінімального остовного дерева. Він полягає в об'єднанні всіх точок між собою таким чином, щоб між будь-якими двома точками був зв'язок напрямую або через інші точки. Для коректної роботи методу необхідно поєднати кожен точку з щонайменше трьома іншими точками. Далі шляхом перебору визначається оптимальний шлях між будь-якими двома точками шляхом прокладання маршруту між точками з найменшим шляхом між ними. Але для роботи системи служби кур'єрської доставки даний метод не підходить через те, що шлях між всіма точками повинен бути заздалегідь відомий для створення зв'язку між точками. Для отримання таких даних, якщо їх немає, потрібно використати той самий метод повного перебору, що не задовільнює критерію швидкості виконання розрахунку.

Наступний метод – метод найближчого сусіда. Він полягає у тому, щоб знаходити найближчу наступну точку на кожній ітерації. Тобто для початкової точки перебираються всі точки доставки та знаходиться точка, шлях до якої є найкоротшим, та помічається як використана. Далі від знайденої точки перебираються всі невикористані точки, що залишились, та знаходиться наступна точка, шлях до якої є найкоротшим, і так до тих пір, поки всі точки не будуть використані. Цей алгоритм не є оптимальним з точки зору коректності знаходження вірного рішення, але статистика використання такого алгоритму стверджує, що коректність знаходження правильного рішення складає біля 75-80% в середньому, цей показник залежить від розташування точок. Такий метод, на відміну від методу повного перебору, не потребує таких ресурсів під час розрахунку. Тому, зважаючи на достатню для початкового етапу точність, було вирішено

використати метод найближчого сусіда як базу в розробці системи (табл. 2.3).

Таблиця 2.3 – Порівняння існуючих методів пошуку оптимального маршруту

Метод	Переваги	Недоліки
Повний перебор	Висока точність	Надзвичайно вимогливий до ресурсів
Мінімальне остовне дерево	Достатньо висока точність	Необхідні заздалегідь наявні дані про довжину шляху між всіма точками
Найближчий сусід	Достатньо висока точність	Може мати неоптимальний результат пошуку

В класичній версії пошук точок з найменшим шляхом починається від точки відправлення. Це підходить для пошуку одного оптимального маршруту, але при пошуку декількох оптимальних маршрутів алгоритм зтикається з тим, що не можна достовірно розрахувати першу точку для кожного з кур'єрів. Тому було вирішено за першу точку взяти найбільш віддалену від початкової точки. Таким чином буде знайдено такий самий маршрут, але в зворотньому порядку. Така модифікація методу дозволяє використати таку умову: якщо відстань від поточної точки до початкової менше, ніж до наступної найближчої, то виходить що додати цю точку до поточного маршруту буде рівноцінно або довше, ніж повернутись на початкову точку та знайти наступний оптимальний маршрут з новими доставками.

З економічної точки зору оптимізація роботи служби доставки полягає у розрахунку вартості доставки для кожного маршруту. Після отримання

точок кожного маршруту для отримання реальної дистанції пройденого шляху кур'єром буде робитися уточнення шляхом запиту до сервісів Google.

Google – американська компанія, основні продукти якої: пошукова система в мережі Інтернет, відеохостинг YouTube, картографічний сервіс Google Maps, поштовий сервіс Gmail, хмарне сховище Google Drive, рішення для реклами AdSense. Компанія надає найбільш точні дані, пов'язані з картами місцевості, так як дані постійно уточнюються. Google Cloud Platform – портал, на якому можна отримати доступ до публічних API, що надаються компанією. Саме тут надається доступ до сервісів, пов'язаних з картами, серед яких Maps API (відображення карти Google на сторонньому ресурсі), Places API (пошук інформації про об'єкт по його адресі), Geocoding API (отримання інформації про об'єкт по координатам його місцезнаходження), Directions API (прокладання маршруту по дорогам між точками на карті). Для реалізації проекту використовується два з них, а саме Places API та Directions API.

Places API використовується для отримання координати точки за географічною адресою. Пропонуються такі можливості інтеграції, як POST-запит з передачею в якості параметрів рядка адреси, або інтеграція віджету Autocomplete (доступний на платформах Android, iOS, JavaScript), який передбачає підказки під час вводу адреси. Обидва способи в результаті виконання повернуть масив даних, що включає в себе географічні координати точки.

Directions API використовується для розрахунку маршрутної дистанції між точками. При використанні передається параметр типу пересування mode, який дозволяє обрати метод пересування серед варіантів: автомобіль, велосипед, піший, транспорт суспільного користування. При розрахунку використовується саме дистанція по дорозі, якій буде слідувати транспортний засіб.

Завдяки цим двом API система отримує точні координати точки за адресою при створення нового замовлення, та передає ці точки для подальшого розрахунку оптимального порядку та дистанції маршруту.

Покрокове виконання алгоритму роботи системи пошуку оптимальних маршрутів виглядає так:

- знаходження найбільш віддаленої точки;
- знаходження наступної точки за принципом найкоротшого шляху, до тих пір поки всі точки не будуть використані, або шлях до початкової точки буде менше ніж до наступної;
- фіксація маршруту при виконанні попереднього кроку;
- формування наступного маршруту, за наявності невикористаних точок та вільних кур'єрів, починаючи з пошуку найбільш віддаленої точки;
- після формування всіх маршрутів за кількістю вільних кур'єрів при наявності невикористаних точок вони розподіляються за готовими маршрутами шляхом знаходження найближчої точки;
- коли всі маршрути сформовано, запит до Google Directions API уточнює реальну дистанцію маршруту між точками, включаючи шлях до початкової точки;
- після отримання всіх сформованих маршрутів з дистанцією на них призначаються кур'єри за наступним принципом: на довший маршрут призначається кур'єр з найменшим показником витрати пального (рис. 2.4).

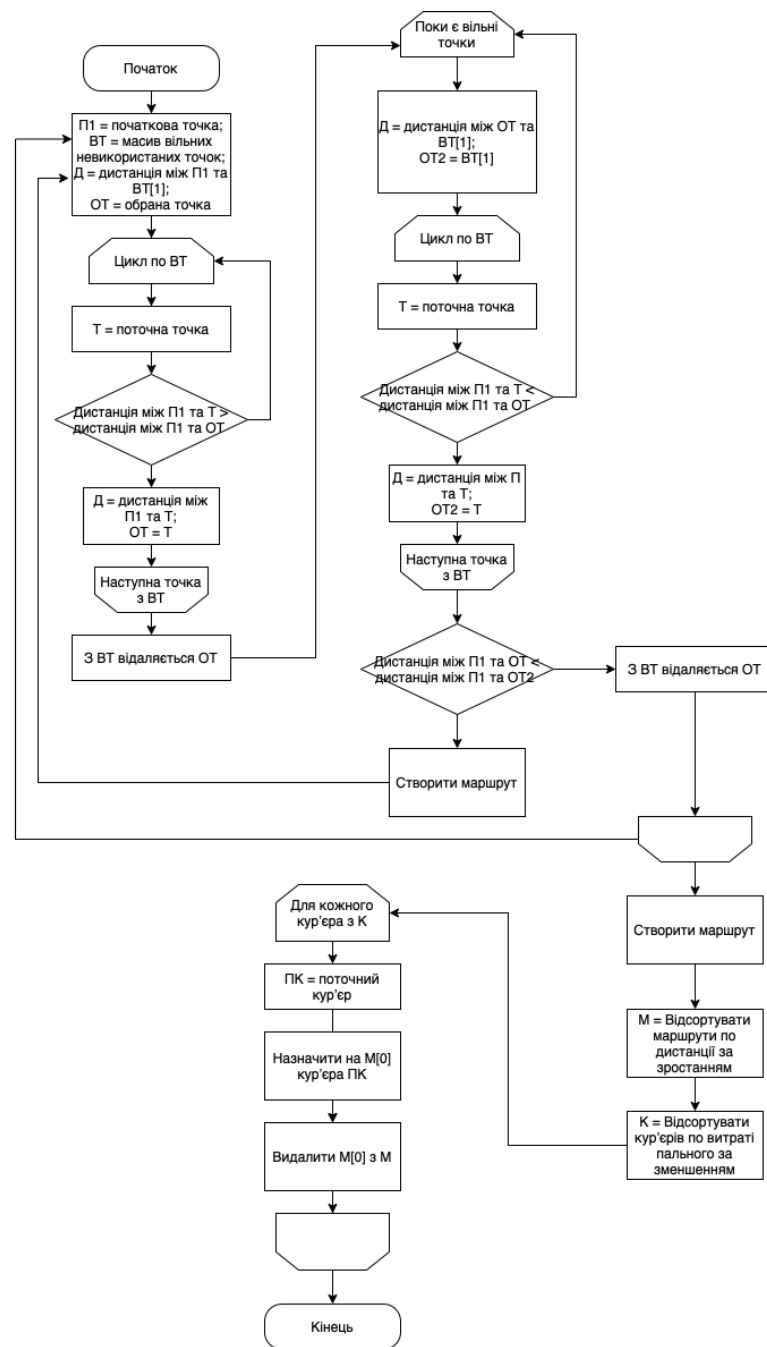


Рисунок 2.4 – Блок-схема алгоритму пошуку оптимального маршруту

Для коректного контролю виконання кур'єром своїх обов'язків система фіксує час та місцезнаходження кур'єра на момент доставки замовлення. Аналіз цих даних у подальшому може виявити проблеми з трудовою дисципліною працівників служби доставки, а також ці дані можуть бути корисними при вирішенні конфліктних ситуацій з клієнтом, якщо маються претензії з приводу часу доставки (рис. 2.5).



Рисунок 2.5 – Функціональна схема роботи системи

2.4 Висновки за розділом

На основі аналізу мов програмування та середовищ для серверної частини системи було обрано мову програмування PHP та середовище розробки JetBrains PhpStorm. Для клієнтської частини було обрано платформу Android, мову програмування Java та середовище розробки Android Studio. Було розроблено структуру бази даних та обрано формат обміну даних, у вигляді JSON, також було розроблено архітектуру програми. Для реалізації програми було вирішено використовувати концепцію MVC. Для реалізації алгоритму пошуку оптимального маршруту було обрано модифікований метод найближчого сусіда.

3 КЕРІВНИЦТВО ПРОГРАМІСТА

3.1 Реалізація бази даних

Для реалізації бази даних було обрано MySQL.

В процесі роботи було реалізовано базу даних, яка містить в собі такі таблиці (табл. 3.1):

Таблиця 3.1 – Таблиці БД

Назва таблиці	Короткий опис таблиці
couriers	Дані кур'єрів
deliveries	Дані про доставки
routes	Дані про маршрути
users	Дані про логістів

У наступних таблицях наведено типи даних, що містяться в таблицях бази даних (табл. 3.2 – табл. 3.8):

Таблиця 3.2 – Типи даних у таблиці couriers

Назва стовпця	Тип даних
id	int(11)
name	varchar(99)
pin	int(11)
consumption	int(11)

Таблиця 3.3 – Типи даних у таблиці deliveries

Назва стовпця	Тип даних
id	int(11)
route_id	int(11)
delivery_time	datetime
address	varchar(255)
phone	varchar(20)
lat	double
lng	double
delivery_time_fixed	datetime
lat_fixed	double
lng_fixed	double

Таблиця 3.4 – Типи даних у таблиці routes

Назва стовпця	Тип даних
id	int(11)
date_added	datetime
logistician_id	int(11)
courier_id	int(11)
status	int(11)
distance	double

Таблиця 3.5 – Типи даних у таблиці users

Назва стовпця	Тип даних
id	int(11)
login	varchar(99)
password	varchar(255)
lat	double
lng	double

Нижче наведено концептуальну схему бази даних (рис. 3.1).

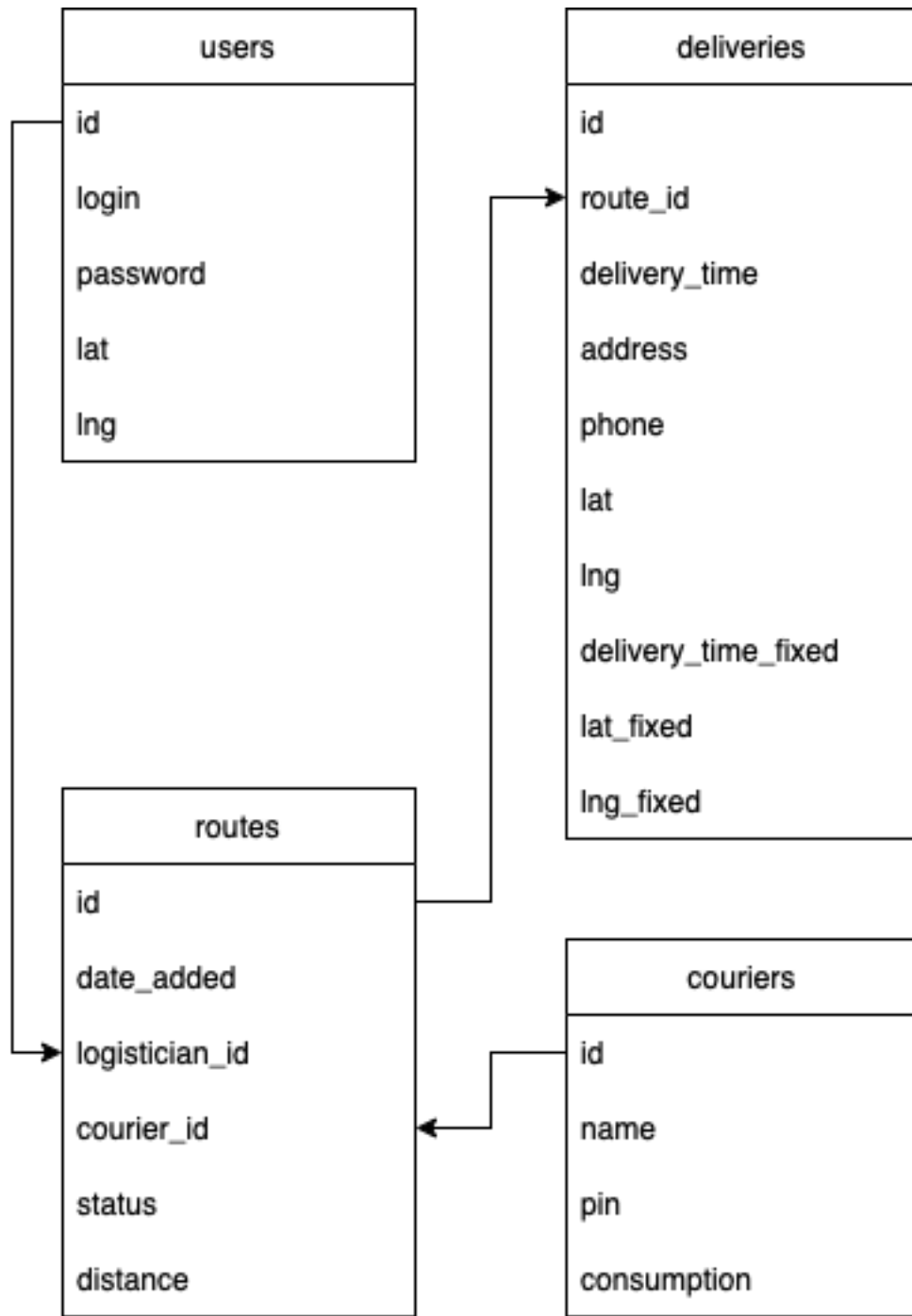


Рисунок 3.1 – Концептуальна модель бази даних

3.2 Реалізація серверної частини системи

3.2.1 Контролери

Під час розробки серверної частини програми було створено файли контролерів `DeliveryController`, `RouteController`, `UserController` (табл. 3.6 – табл. 3.8):

Таблиця 3.6– Функції класу `DeliveryController`

Назва функції	Метод моделі, що викликається
<code>newDelivery()</code>	<code>Delivery::newDelivery();</code>
<code>confirmDelivery()</code>	<code>Delivery::confirmDelivery();</code>
<code>getDelivery()</code>	<code>Delivery::getDelivery();</code>
<code>assignRoute()</code>	<code>Delivery::assignRoute();</code>
<code>getAllPendingDeliveries()</code>	<code>Delivery::getAllPendingDeliveries();</code>
<code>startRouteCalculation()</code>	<code>Delivery::startRouteCalculation();</code>

Таблиця 3.7 – Функції класу `RouteController`

Назва функції	Метод моделі, що викликається
<code>getAllNewRoutes()</code>	<code>Route::getAllNewRoutes();</code>
<code>getAllActiveRoutes()</code>	<code>Route::getAllActiveRoutes();</code>
<code>getAllRoutesByDate()</code>	<code>Route::getAllRoutesByDate();</code>
<code>newRoute()</code>	<code>Route::newRoute();</code>
<code>getActiveRoute()</code>	<code>Route::getActiveRoute();</code>

Таблиця 3.8 – Функції класу UserController

Назва функції	Метод моделі, що викликається
login()	User::login();
authorize()	User::authorize();
registerUser()	User::registerUser();
registerCourier()	User::registerCourier();
getCouriers()	User::getCouriers();
getUsers()	User::getUsers();

Таким чином кожен запит до API викликає певний метод контроллера, у якому відбувається перевірка наявності необхідних даних, що були передані POST-параметром HTTP запиту.

В HTTP-запитах існує поняття статусу запиту. За статусом можна визначити, чи виникли помилки при виконання, та якщо помилка виникла, то яка саме.

Нижче наведені деякі найбільш відомі статуси (табл. 3.9):

Таблиця 3.9 – Статуси запитів HTTP

Код статусу	Короткий опис
1	2
200	Успішно
400	Невірний синтаксис запиту
401	Неавторизовано

403	Доступ не надано
-----	------------------

Продовження таблиці 3.9

1	2
404	Ресурс не знайдено
500	Внутрішня помилка сервера
502	Сервер отримав недопустиму відповідь від стороннього ресурсу

При відсутності необхідних даних контролер повертає у відповідь код помилки 404 з описом помилки.

3.2.2 Моделі

Під час розробки було створено файли моделей Delivery, Route, User (табл. 3.10 – табл. 3.12):

Таблиця 3.10 – Функції класу Delivery

Назва функції	Короткий опис функції
1	2
newDelivery(\$address, \$phone, \$lat, \$lng, \$datetime)	Функція створює новий запис в таблиці deliveries, використовуючи першочергово необхідні дані, такі як адреса, номер телефону клієнта, місцезнаходження точки, дата та час доставки
confirmDelivery(\$id, \$lat, \$lng)	Функція оновлює дані в таблиці deliveries, фіксуючи фактичний час доставки замовлення та місцезнаходження кур'єра на момент доставки

getDelivery(\$id)	Функція повертає всю інформацію про конкретну доставку
-------------------	--

Продовження таблиці 3.10

1	2
getAllPendingDeliveries()	Функція повертає масив всіх доставок, що ще не були доставлені
startRouteCalculation(\$lat, \$lng, \$couriers)	Функція ініціює початок роботи алгоритму з пошуку оптимальних маршрутів
checkIfNewRouteNeeded(\$point1Lat, \$point1Lng, \$point2Lat, \$point2Lng, \$originLat, \$originLng)	Функція перевіряє, чи є необхідність у створенні нового маршруту, чи можна додати точку у поточний маршрут
getDistance(\$lat1, \$lng1, \$lat2, \$lng2)	Функція повертає відстань між двома точками
getNearestPoint(\$pendingDeliveries, \$origin)	Функція повертає точку, найближчу до поточної
getActualDistance(\$lat, \$lng, \$routes, \$couriers)	Функція звертається до Google Directions API та отримує маршрутну відстань між точками
assignCourier(\$routes, \$couriers)	Функція назначає кур'єрів на маршрут за принципом: на довший маршрут призначається кур'єр з найменшим показником витрати пального

saveCourier(\$courier_id, \$distance)	Функція зберігає утверджений маршрут до таблиці routes
---------------------------------------	--

Таблиця 3.11 – Функції класу Route

Назва функції	Короткий опис функції
getAllNewRoutes()	Функція повертає масив всіх пустих маршрутів, які ще не було підтверджено
getAllActiveRoutes()	Функція повертає масив всіх маршрутів, які було підтверджено
getAllRoutesByDate(\$from, \$to)	Функція повертає масив всіх маршрутів, які потрапляють в діапазон дат від-до
getDeliveries(\$id)	Функція повертає масив всіх доставок в маршруті
newRoute(\$logistician_id, \$courier_id)	Функція створює новий запис в таблиці routes, використовуючи першочергово необхідні дані, такі як id логіста та кур'єра
getActiveRoute(\$courier_id)	Функція повертає активний маршрут для кур'єра

Таблиця 3.12 – Функції класу User

Назва функції	Короткий опис функції
1	2
login(\$login, \$password)	Функція авторизує логіста в системі використовуючи його логін та пароль, зашифрований за алгоритмом MD5
authorize(\$id, \$pin)	Функція авторизує кур'єра в додатку використовуючи його id та PIN-код

Продовження таблиці 3.12

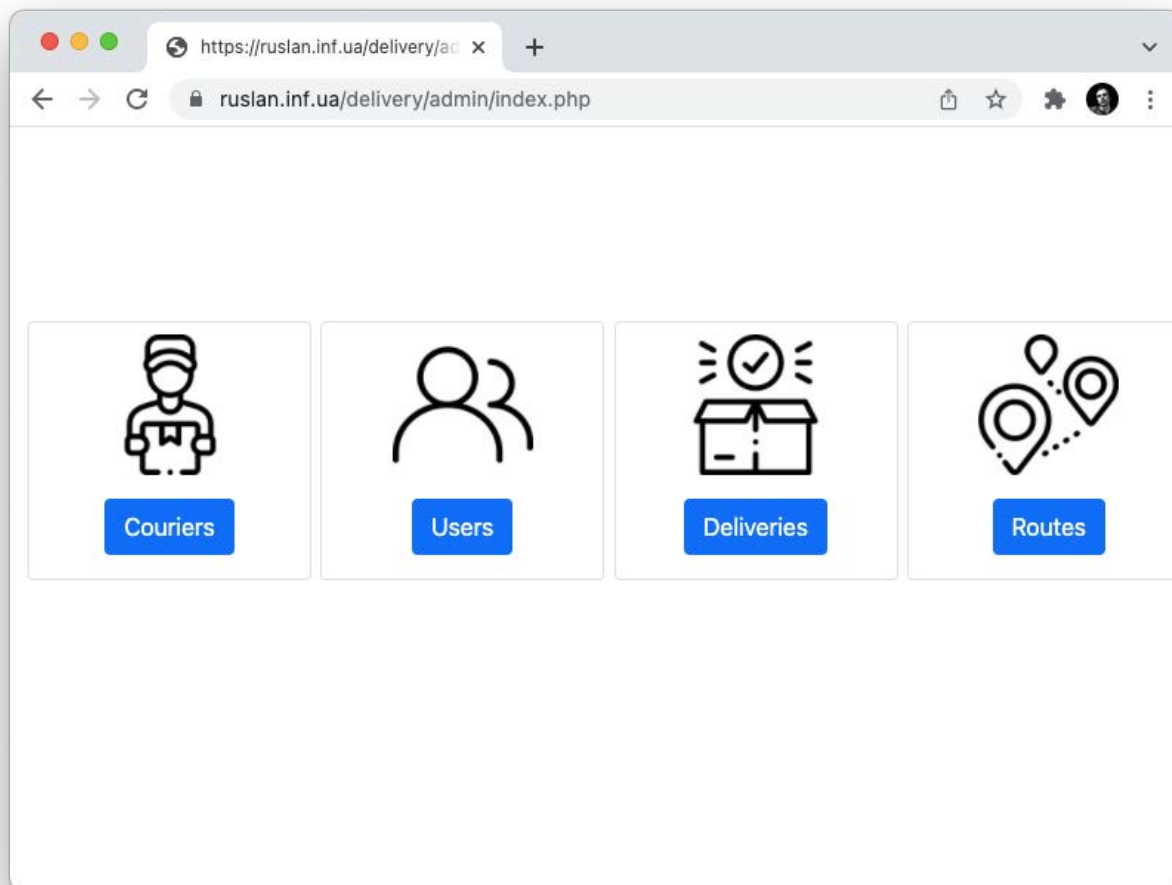
1	2
<code>registerUser(\$login, \$password)</code>	Функція реєструє логіста в системі використовуючи логін та пароль, зашифрований за алгоритмом MD5
<code>registerCourier(\$name, \$consumption)</code>	Функція реєструє кур'єра в системі, генеруючи його секретний PIN-код, та зберігає дані з витрати пального
<code>getCouriers()</code>	Функція повертає масив всіх кур'єрів

3.2.3 Представлення

Для веб-інтерфейсу адміністратора було використано мову розмітки сторінки HTML, таблиці стилів CSS та мову програмування JavaScript [12]-[16]. Було розроблено такі сторінки:

- головна сторінка;
- перелік логістів;
- перелік кур'єрів;
- перелік доставок;
- перелік маршрутів;
- реєстрація логіста;
- реєстрація кур'єра;
- створення нової доставки.

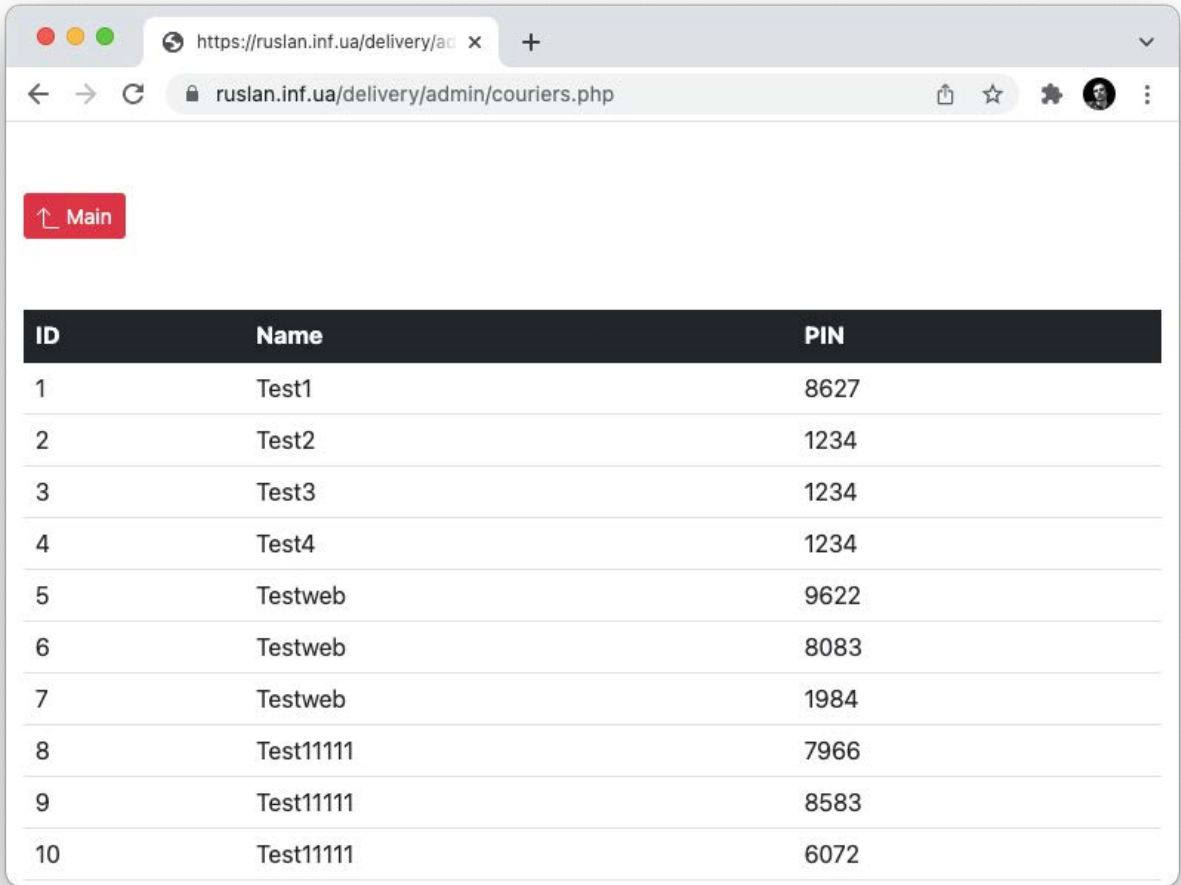
На головній сторінці доступне меню переходу до основних розділів



системи (рис. 3.2).

Рисунок 3.2 - Головне меню

В розділі переліку кур'єрів можна побачити список всіх кур'єрів та

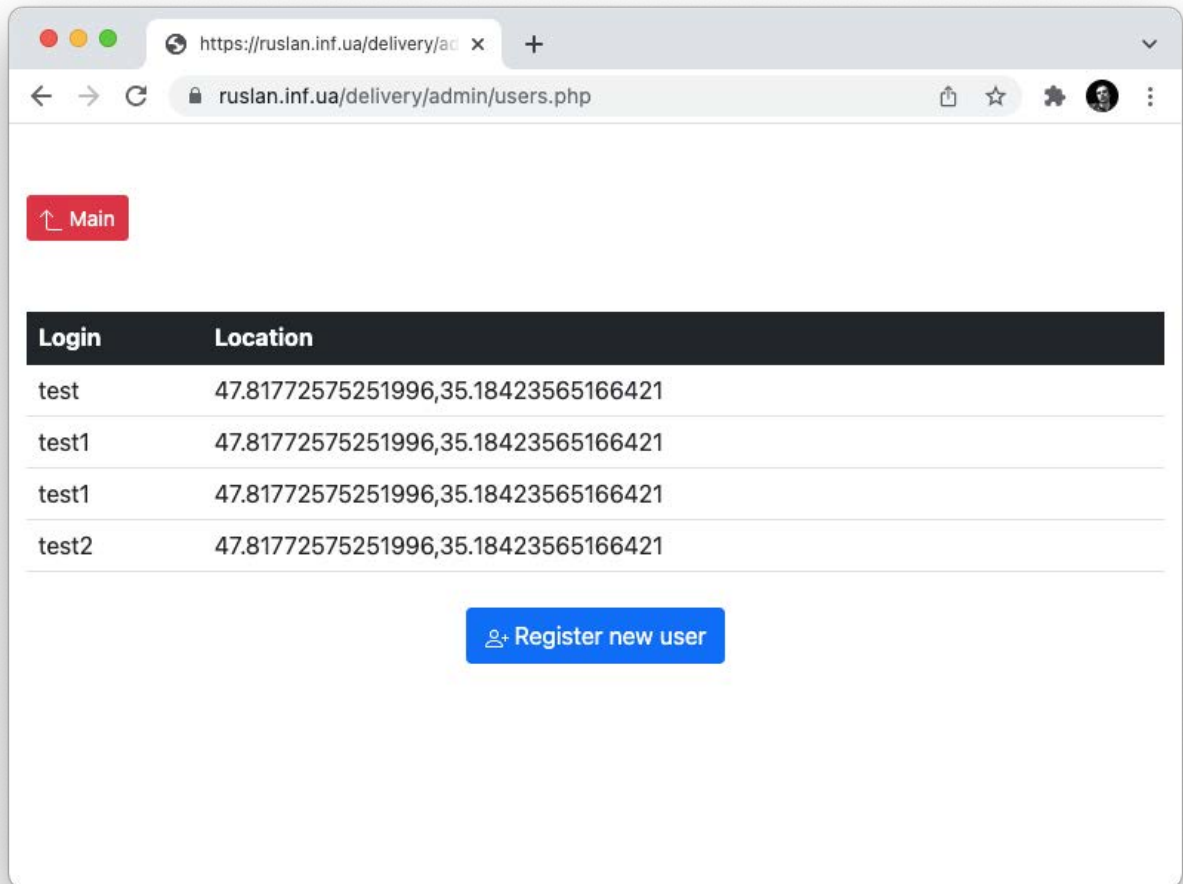


ID	Name	PIN
1	Test1	8627
2	Test2	1234
3	Test3	1234
4	Test4	1234
5	Testweb	9622
6	Testweb	8083
7	Testweb	1984
8	Test11111	7966
9	Test11111	8583
10	Test11111	6072

кнопку переходу на сторінку створення нового кур'єра (рис. 3.3).

Рисунок 3.3 - Перелік кур'єрів

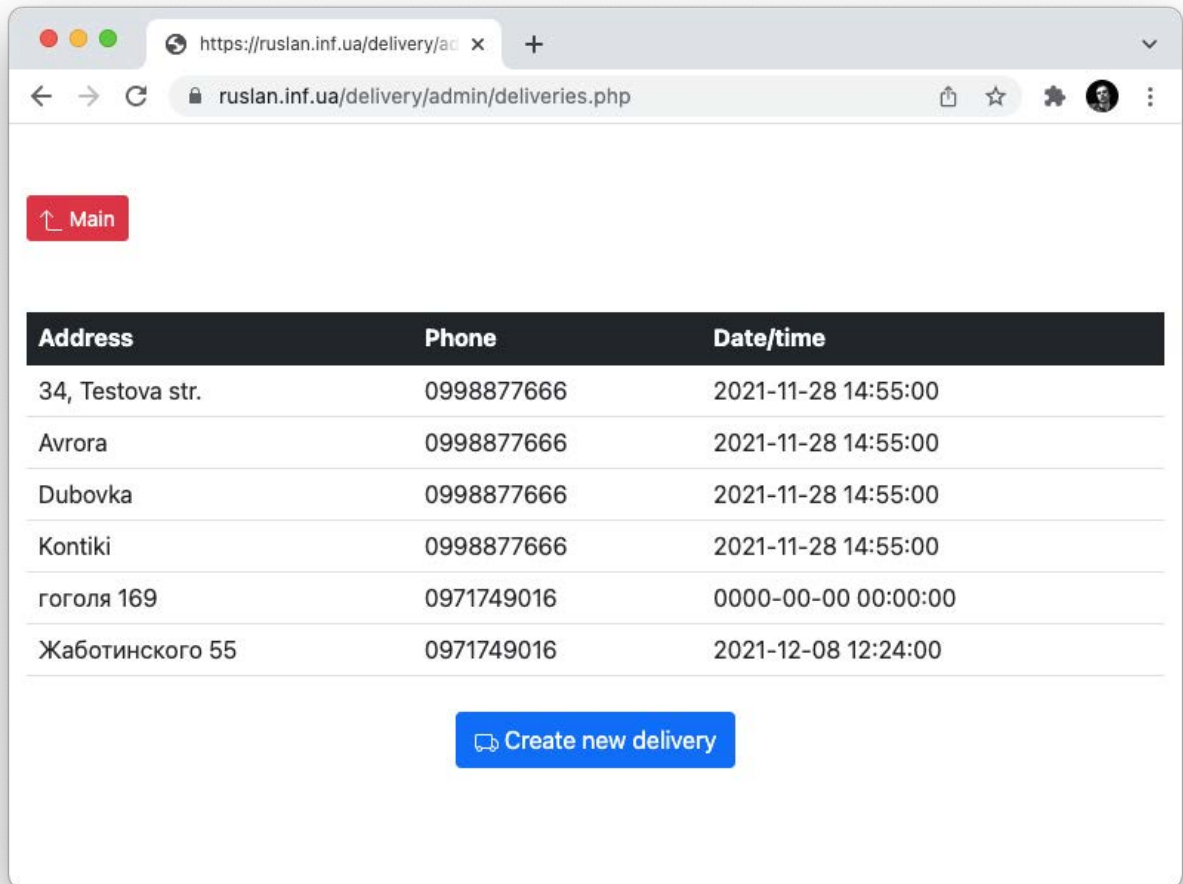
В розділі переліку користувачів можна побачити список всіх логістів



та кнопку переходу на сторінку створення нового логіста (рис. 3.4).

Рисунок 3.4 - Перелік користувачів

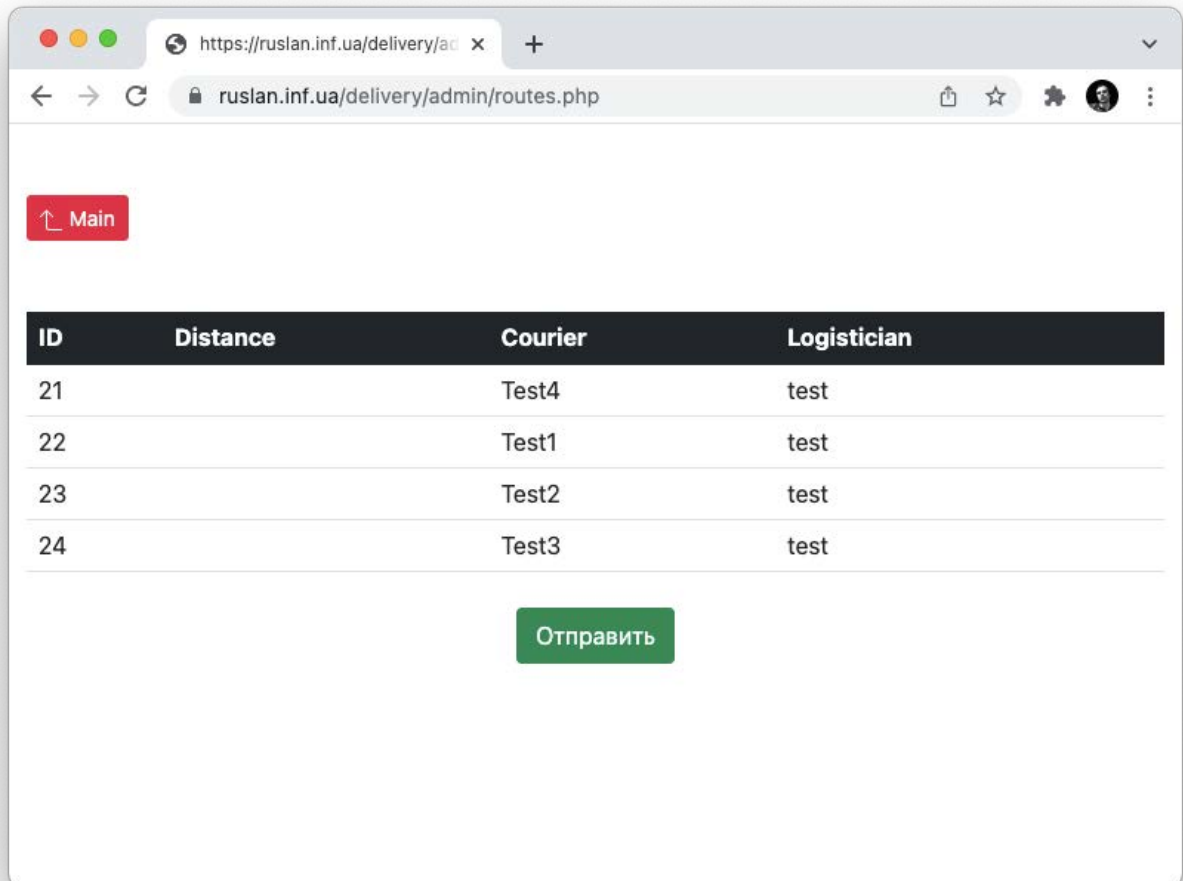
В розділі переліку доставок можна побачити список всіх доставок та



кнопку переходу на сторінку створення нової доставки (рис. 3.5).

Рисунок 3.5 - Перелік доставок

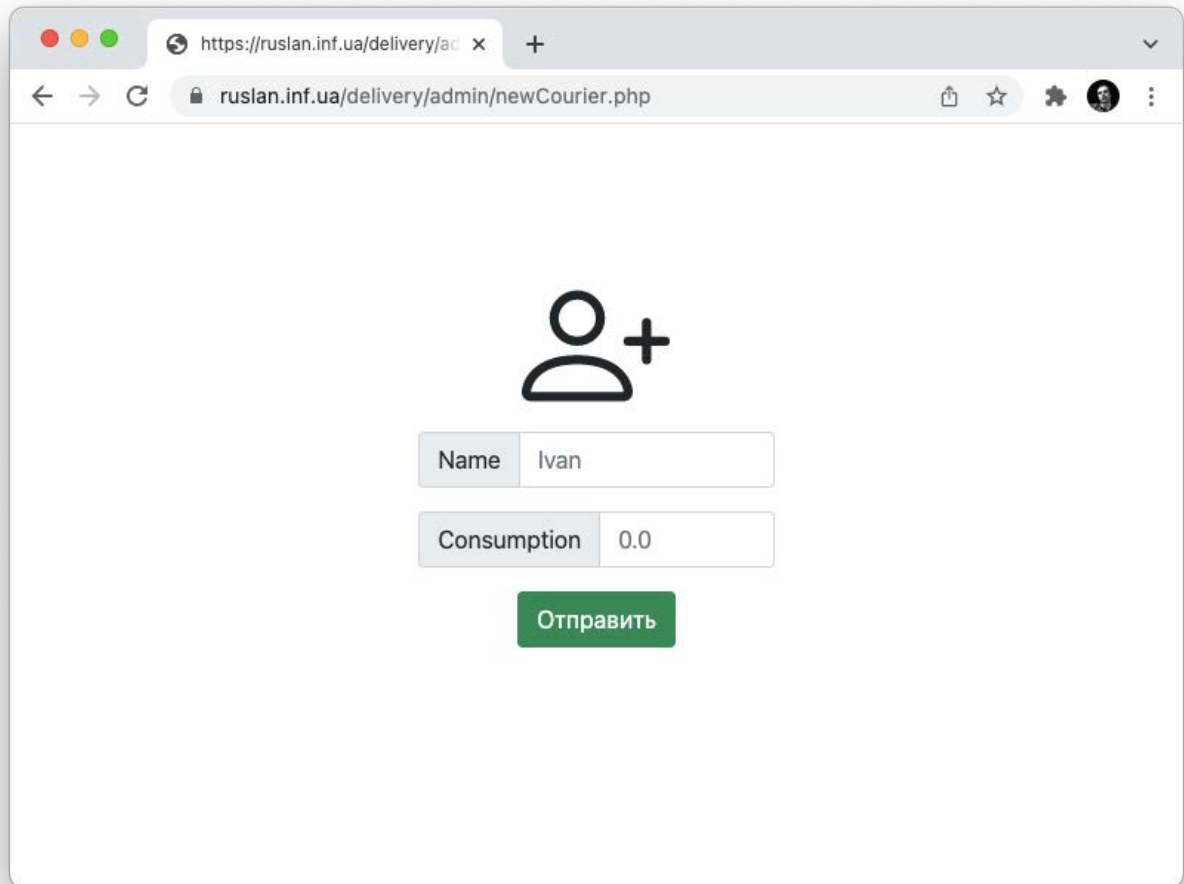
В розділі переліку маршрутів можна побачити список всіх маршрутів та кнопку автоматичного розрахунку нових маршрутів з доступних доставок



(рис. 3.6).

Рисунок 3.6 - Перелік маршрутів

В розділі реєстрації нового кур'єра необхідно ввести необхідні дані,



The screenshot shows a web browser window with the address bar displaying `https://ruslan.inf.ua/delivery/admin/newCourier.php`. The page content includes a user icon with a plus sign, a form with two input fields, and a green button labeled "Отправить".

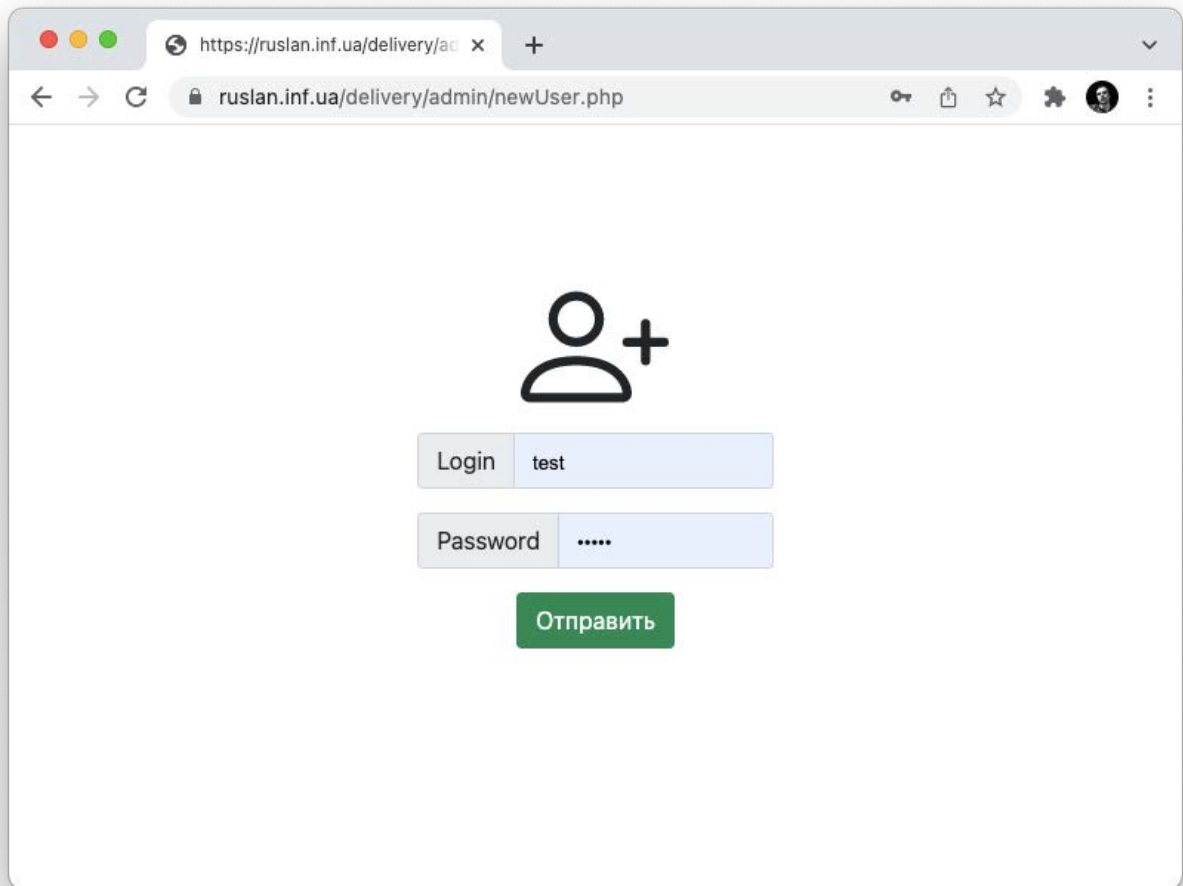
Field	Value
Name	Ivan
Consumption	0.0

Отправить

такі як ім'я та показник витрати пального (рис. 3.7).

Рисунок 3.7 - Реєстрація кур'єра

В розділі реєстрації нового користувача необхідно ввести необхідні

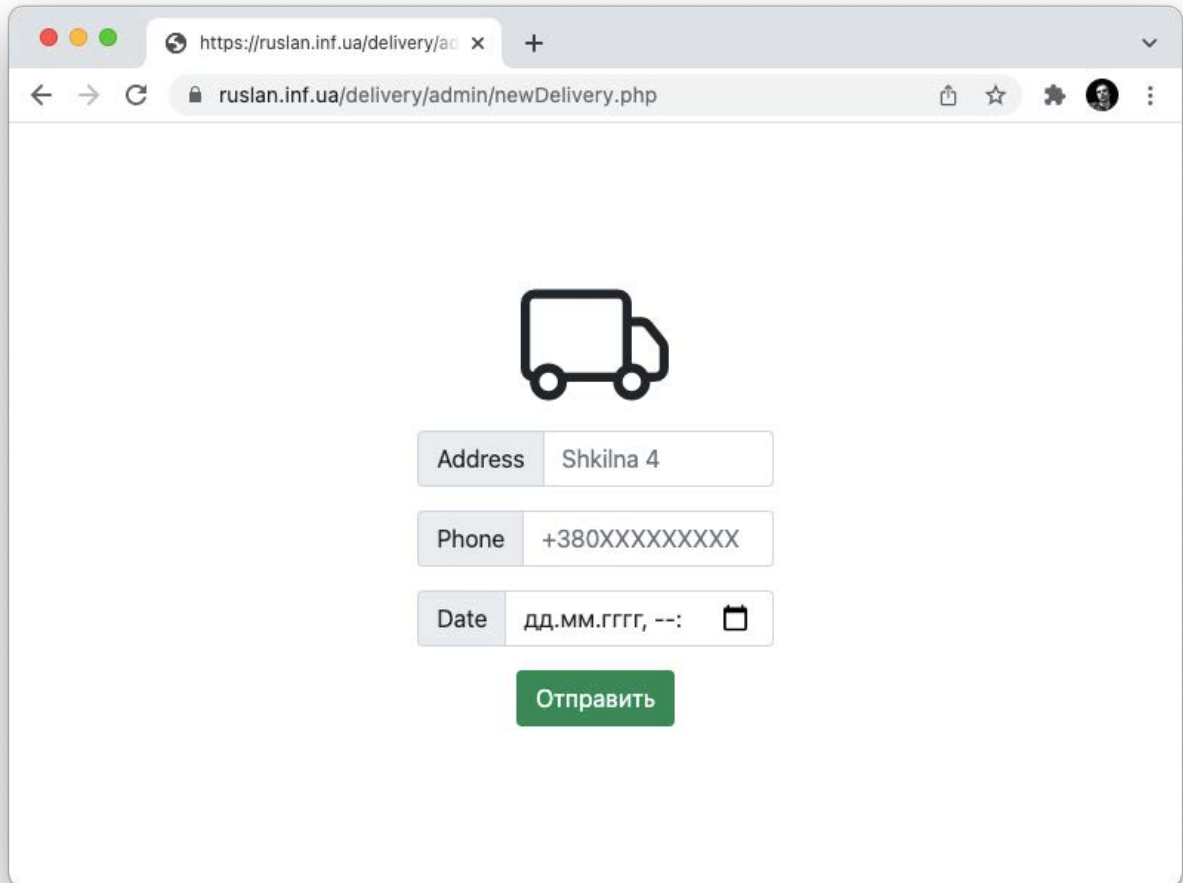


The screenshot shows a web browser window with the address bar displaying `https://ruslan.inf.ua/delivery/admin/newUser.php`. The page content features a large icon of a person with a plus sign, indicating a new user registration. Below this icon are two input fields: the first is labeled "Login" and contains the text "test"; the second is labeled "Password" and contains masked characters ".....". A green button labeled "Отправить" (Submit) is positioned below the password field.

дані, такі як логін та пароль (рис. 3.8).

Рисунок 3.8 - Реєстрація користувача

В розділі створення нової доставки необхідно ввести необхідні дані,



The screenshot shows a web browser window with the address bar displaying `https://ruslan.inf.ua/delivery/admin/newDelivery.php`. The page content features a central truck icon. Below the icon are three input fields: 'Address' with the value 'Shkilna 4', 'Phone' with the value '+380XXXXXXXXXX', and 'Date' with the placeholder 'дд.мм.гггг, --:' and a calendar icon. A green button labeled 'Отправить' is positioned below the date field.

такі як адреса, номер телефону та дата доставки (рис. 3.9).

Рисунок 3.9 - Створення доставки

Для відображення візуального інтерфейсу форми було обрано тег `<form>`, використовуваний для відображення елементів вводу даних та для відправки даних з цих елементів на вказаний URL. Всередині тегу `<form>` розміщують теги `<input>`, які містять в собі характеристики полів для заповнення.

Існують такі елементи форми:

- `text` – звичайний текст;
- `number` – тільки цифри;
- `password` – текст, що не відображається в полі;
- `radio` – вибір одного з варіантів;
- `checkbox` – вибір декількох варіантів;
- `date` – вибір дати за допомогою календаря (рис. 3.10);

- `submit` – кнопка.

Рисунок 3.10 – Елемент `date` форми HTML

Для відправки запитів використовується POST-запит.

У розділі «Реєстрація кур'єра» відображається HTML-форма з полями:

- name (text);
- consumption (number).

Дані за допомогою POST-запиту відправляються до couriers.php, звідки викликається User::registerCourier().

У розділі «Реєстрація користувача» відображається HTML-форма з полями:

- login (text);
- password (password).

Дані за допомогою POST-запиту відправляються до users.php, звідки викликається User::registerUser().

3.2.4 Дизайн

Для створення дизайну були використані каскадні таблиці стилів CSS. CSS описує, як HTML-елементи повинні відображатися на екрані, папері чи інших носіях.

CSS можна додати до елементів HTML 3-ма способами:

- Вбудований - за допомогою атрибуту стилю в елементах HTML;
- Внутрішній - за допомогою елемента <style> у розділі <head>;
- Зовнішній - за допомогою зовнішнього файлу CSS.

Найпоширеніший спосіб додати CSS - це зберігати стилі в окремих CSS-файлах, який і було використано.

Зовнішній аркуш стилів використовується для визначення стилю для багатьох сторінок HTML.

За допомогою зовнішнього аркуша стилів можна змінити вигляд цілого веб-сайту, змінивши один файл.

Щоб використовувати зовнішній аркуш стилів, додайте посилання на нього в розділі `<head>` сторінки HTML.

Зовнішній аркуш стилів можна записати в будь-який текстовий редактор. Файл не повинен містити HTML-код і повинен бути збережений з розширенням `.css`.

3.2.5 Маршрутизація

Для роботи запитів API за неявними посиланнями було розроблено систему маршрутизації, яка приймає параметр «action» та зветрається до необхідного контролера. На початку файлу маршрутизації підключаються всі необхідні класи, після чого конструкцією «switch» викликаються необхідні функції. Нижче наведено значення параметру «action» та викликані функції (табл. 3.13):

Таблиця 3.13 – Маршрутизація запитів

Параметр «action»	Функція
login	UserController::login()
authorize	UserController::authorize()
registeruser	UserController::registerUser()
registercourier	UserController::registerCourier()
newdelivery	DeliveryController::newDelivery()
getdeliveries	RouteController::getDeliveries()
startroutecalculatation	DeliveryController::startRouteCalculation()
getactiveroute	RouteController::getActiveRoute()
confirmdelivery	DeliveryController::confirmDelivery()

3.2.6 Підключення до бази даних

Для підключення до бази даних було використано клас DB з методом get(). Цей метод ініціює підключення до бази даних в використанні таких даних, як хост, назва бази, логін та пароль.

3.3 Розробка мобільного додатку

3.3.1 Контролери

Під час розробки мобільного додатку для платформи Android було створено класи-нащадки Activity: LoginActivity, MainActivity (табл. 3.14 – табл. 3.16):

Таблиця 3.14 – Функції класу LoginActivity

Назва функції	Короткий опис функції
onCreate(Bundle savedInstanceState)	Вбудована функція, що викликається при створенні об'єкту класу
authorize()	Функція робить запит action=authorize для підтвердження дійсності даних входу кур'єру

Таблиця 3.15 – Функції класу MainActivity

Назва функції	Короткий опис функції
1	2
onCreate(Bundle savedInstanceState)	Вбудована функція, що викликається при створенні об'єкту класу

Продовження таблиці 3.15

1	2
onBackPressed()	Функція переавантажує стандартну функцію, з метою закриття додатку при натиску кнопки «назад», щоб запобігти повторного відкриття вікна авторизації
getCurrentRoute()	Функція робить запит для отримання поточного маршруту кур'єра
getDeliveriesInRoute(String route_id)	Функція робить запит для отримання всіх доставок у маршруті
showDelivery()	Функція відображає дані про доточну доставку та налаштовує дії кнопок дзвінка та навігації
callAction(String phone)	Функція ініціює відкриття стандартного додатку для дзвінків та передає номер телефону клієнта
navigateAction(String lat, String lng)	Функція ініціює відкриття додатку Google Maps та передає координати точки доставки для відображення шляху в навігаторі
doneAction()	Функція відправляє дані поточного місцезнаходження пристрою для фіксації факту доставки

Продовження таблиці 3.15

Назва функції	Короткий опис функції
<code>navigateAction(String lat, String lng)</code>	Функція ініціює відкриття додатку Google Maps та передає координати точки доставки для відображення шляху в навігаторі
<code>doneAction()</code>	Функція відправляє дані поточного місцезнаходження пристрою для фіксації факту доставки
<code>getLocation()</code>	Функція ініціює відслідковування поточного місцезнаходження пристрою, або при скасуванні дозволу на роботу з місцезнаходженням пристрою, створює запит на надання такого доступу
<code>cancelLocation()</code>	Функція скасовує відслідковування поточного місцезнаходження пристрою
<code>onRequestPermissionsResult(int requestCode, String permissions[], int[] grantResults)</code>	Функція обробляє відповідь на запит на надання доступу до місцезнаходження пристрою

Для реалізації HTTP-запитів у мобільному додатку було використано бібліотеку Volley. Volley — це бібліотека HTTP, яка робить роботу в мережі для додатків Android простішою і, головне, швидшою. Volley доступний на GitHub.

Volley пропонує наступні переваги:

- автоматичне планування мережових запитів;
- кілька одночасних мережових підключень;

- прозоре кешування відповідей на диск і пам'ять із стандартною узгодженістю кешу HTTP;
- підтримка визначення пріоритетів запитів;
- API запиту на скасування. Ви можете скасувати один запит або встановити блоки чи діапазони запитів для скасування;
- простота налаштування, наприклад, для повторної спроби та повернення;
- надійне впорядкування, яке полегшує правильне заповнення інтерфейсу користувача даними, які асинхронно завантажуються з мережі;
- інструменти налагодження та трасування.

Volley відмінно виконує операції типу RPC, які використовуються для заповнення інтерфейсу користувача, наприклад отримання сторінки результатів пошуку як структурованих даних. Він легко інтегрується з будь-яким протоколом і поставляється з підтримкою необроблених рядків, зображень і JSON. Надаючи вбудовану підтримку необхідних функцій, Volley звільняє розробника від написання шаблонного коду і дозволяє зосередитися на логіці, характерній для програми.

Volley не підходить для великих операцій завантаження або потокової передачі, оскільки Volley зберігає всі відповіді в пам'яті під час аналізу. Для великих операцій завантаження є сенс задуматись про використання альтернативи, наприклад DownloadManager.

Основна бібліотека Volley розроблена на GitHub і містить основний конвеєр відправлення запитів, а також набір загальнозастосовних утиліт, доступних у наборі інструментів Volley. Найпростіший спосіб інтеграції бібліотеки у проект – за допомогою Gradle.

Gradle — це інструмент збірки, зосереджений на автоматизації збірки та підтримці багатомовної розробки. Якщо ви створюєте, тестуєте, публікуєте та розгортаєте програмне забезпечення на будь-якій платформі, Gradle пропонує гнучку модель, яка може підтримувати весь життєвий цикл розробки від компіляції та пакування коду до публікації веб-сайтів. Gradle був розроблений для підтримки автоматизації збірки на кількох мовах і

платформах, включаючи Java, Scala, Android, Kotlin, C/C++ і Groovy, і тісно інтегрований з інструментами розробки та серверами безперервної інтеграції, включаючи Eclipse, IntelliJ і Jenkins.

Мінімальною версією операційної системи Android, яку потребує додаток для запуску, є версія Android 5.0 Lollipop, що відповідає рівню API 21. Згідно статистики, що надає компанія Google, відсоток пристроїв з версією 5.0 і вище складає 98% [9] (рис. 3.11).

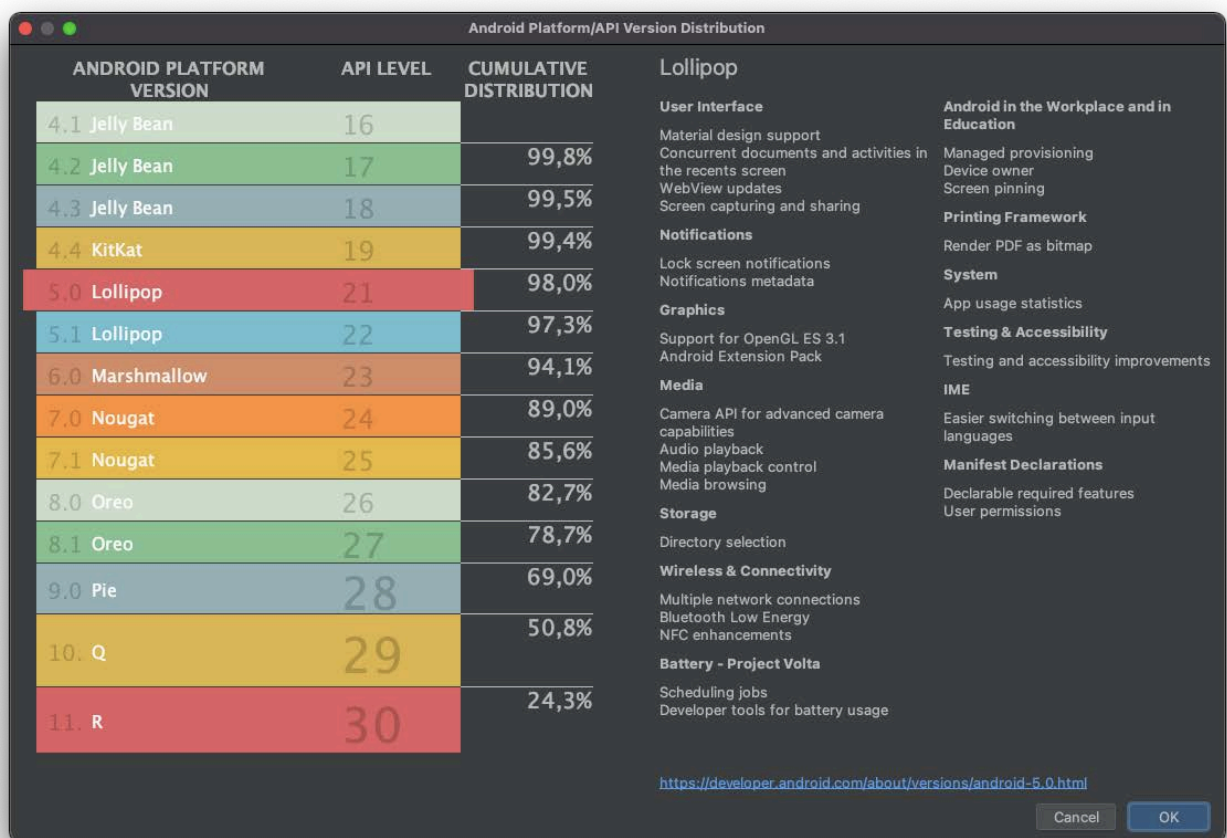


Рисунок 3.11 – Відсоткове співвідношення версій Android на активних пристроях [9]

Нижче наведено знімки екрану розділів авторизації (рис. 3.12) та



головного меню додатку (рис. 3.13).

Рисунок 3.12 – Екран авторизації



Рисунок 3.13 – Головне меню

3.4 Висновки за розділом

Було розроблено серверну частину програми на мові PHP, яка використовує бази даних MySQL. За допомогою запитів API було розроблено принцип взаємодії мобільного застосунку користувача з системою. Також було розроблено мобільний додаток для взаємодії з API.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Опис застосування

Призначення програми – автоматичний пошук оптимального маршруту доставки та контроль йактично виконаної доставки.

Областями застосування можуть бути служби доставки, поштові служби.

Мінімальна конфігурація технічних засобів:

- комп'ютер з доступом до мережі Інтернет;
- смартфони з Android 5.0 і вище у кожного кур'єра.

4.2 Методика та результати тестування

Для тестування програмного продукту було створено чек-лист тестування, який включає в себе:

- перевірку реєстрації логістів;
- перевірку реєстрації кур'єрів;
- перевірку створення нової доставки;
- перевірку автоматичного роозахунку маршруту доставки;
- аутентифікацію в системі логіста;
- аутентифікацію в системі кур'єра;
- перевірку отримання та передачі поточного місцезнаходження з мобільного додатку.

Було проведено тестування коректності роботи розробником системи.

4.3 Експериментально отримані результати

Під час тестування було виявлено ряд проблем, які фіксувалися у електронних таблицях та перевірялися експериментально. Було виявлено такі проблеми, як некоректна робота функції отримання поточного місцезнаходження в мобільному додатку, некоректна робота віджету підстановки адреси за допомогою Google API, проблема з фіксацією результатів запиту при нестабільному інтернет-з'єднанні. За результатами тестування ці проблеми було виправлено.

4.4 Висновки за розділом

За результатами тестування було виправлено ряд помилок, які не відтворилися після повторного тестування. Таким чином результати тестування показують систему як працездатну.

ВИСНОВКИ

В результаті роботи було розроблено систему для автоматичного пошуку оптимального маршруту доставки та контроль фактично виконаної доставки. Систему було протестовано в умовах, близьких до реальних. Отримано такі результати, як зменшення середньої дистанції маршруту, зменшення затрат пального кур'єром.

При реалізації проекту було досягнуто мети, яка полягала в створенні програмної системи для розрахунку оптимального маршруту доставки замовлень кур'єрською службою та мобільного додатку для контролю коректності часу та місцезнаходження доставки замовлення.

Система може бути використана на будь-яких підприємствах, що займаються наданням послуг з кур'єрської доставки.

Систему було розроблено на мобільну платформу Android з використанням API. Також було розроблено панель керування для створення даних для передачі в додаток.

Систему було протестовано в умовах, наближених до реальних, та було підтверджено доцільність використання системи у реальних умовах.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Should you always have your phone's GPS Location turned on? [Electronic resource] – Access mode: <https://www.dignited.com/32554/leaving-gps-or-location-on-phone-battery/>
2. Google Maps [Electronic resource] – Access mode: <https://maps.google.com>
3. C2Logix [Electronic resource] – Access mode: <https://c2logix.com/c2routing/>
4. Badger [Electronic resource] – Access mode: <https://www.badgermapping.com/>
5. JetBrains PhpStorm [Electronic resource] – Access mode: <https://www.jetbrains.com/phpstorm/>
6. JetBrains PhpStorm [Електронний ресурс] – Режим доступу: <https://itpro.ua/product/jetbrains-phpstorm/?tab=description>
7. JetBrains DataGrip [Electronic resource] – Access mode: <https://www.jetbrains.com/datagrip/>
8. JetBrains CodeWithMe [Electronic resource] – Access mode: <https://www.jetbrains.com/code-with-me/>
9. Top Android OS versions [Electronic resource] – Access mode: <https://www.appbrain.com/stats/top-android-sdk-versions>
10. Що таке MVC [Електронний ресурс] – Режим доступу: <https://ru.wikipedia.org/wiki/Model-View-Controller>
11. Основні принципи MVC [Електронний ресурс] – Режим доступу: <https://habr.com/ru/post/181772/>
12. Ташков, Петро Веб-мастеринг HTML, CSS, JavaScript, PHP, CMS, AJAX, розкрутка / Петро Ташков. - М .: Книга по Требованию, 2014. - 512 с.
13. Дакетт, Джон HTML і CSS. Розробка і дизайн веб-сайтів / Джон Дакетт. - М .: Ексмо, 2013. - 480 с.

14. Дакетт, Джон Основи веб-програмування з використанням HTML, XHTML і CSS / Джон Дакетт. - М.: Ексмо, 2013. - 768 с.
15. Дебольт HTML і CSS. Спільне використання / Дебольт, Вірджинія. - М.: НТ Пресс, 2013. - 512 с.
16. Чебикін, Ростислав Самовчитель HTML і CSS. Сучасні технології / Ростислав Чебикін. - Москва: Вогні, 2012. - 624 с.

ДОДАТОК А
Текст програми

A1. Текст файлу api.php

```

<?php

require_once $_SERVER['DOCUMENT_ROOT'].'/delivery/db.php';
require_once
$_SERVER['DOCUMENT_ROOT'].'/delivery/controller/UserController.php';
require_once
$_SERVER['DOCUMENT_ROOT'].'/delivery/controller/DeliveryController.php';
require_once
$_SERVER['DOCUMENT_ROOT'].'/delivery/controller/RouteController.php';

switch($_GET['action'])
{
    case "login":
        echo UserController::login();
        break;
    case "authorize":
        echo UserController::authorize();
        break;
    case "registeruser":
        echo UserController::registerUser();
        break;
    case "registercourier":
        echo UserController::registerCourier();
        break;
    case "newdelivery":
        echo DeliveryController::newDelivery();
        break;
    case "getdeliveries":
        echo RouteController::getDeliveries();
        break;
    case "startroutecalculation":
        echo DeliveryController::startRouteCalculation();
        break;
    case "getactiveroute":
        echo RouteController::getActiveRoute();
        break;
    case "confirmdelivery":
        echo DeliveryController::confirmDelivery();
        break;
    default:
        echo 'no route';
        break;
}

```

```
}
```

A2. Текст файла db.php

```
<?php

class DB
{
    public static function get()
    {
        $host = host';
        $db_name = db_name ' ;
        $user = user ' ;
        $password = 'password;
        $db = new
PDO("mysql:host=$host;dbname=$db_name;charset=utf8", $user, $password);

        return $db;
    }
}
```

A3. Текст файла DeliveryController.php

```
<?php

require_once
$_SERVER['DOCUMENT_ROOT'].'/delivery/model/Delivery.php';

class DeliveryController
{
    public static function newDelivery()
    {
        if(isset($_POST['address']) && isset($_POST['phone']) &&
isset($_POST['lat']) && isset($_POST['lng']) && isset($_POST['datetime']))
        {
            return Delivery::newDelivery($_POST['address'], $_POST['phone'],
$_POST['lat'], $_POST['lng'], $_POST['datetime']);
        }
        else
        {
            http_response_code(404);
            $error = array('status'=>'error', 'message'=>'missing params');
            return json_encode($error);
        }
    }
}
```

```

    public static function confirmDelivery()
    {
        if(isset($_POST['id'])      &&      isset($_POST['lat'])      &&
isset($_POST['lng']))
        {
            return Delivery::confirmDelivery($_POST['id'], $_POST['lat'],
$_POST['lng']);
        }
        else
        {
            http_response_code(404);
            $error = array('status'=>'error', 'message'=>'missing params');
            return json_encode($error);
        }
    }

```

```

    public static function getDelivery()
    {
        if(isset($_POST['id']))
        {
            return json_encode(Delivery::getDelivery($_POST['id']));
        }
        else
        {
            http_response_code(404);
            $error = array('status'=>'error', 'message'=>'missing params');
            return json_encode($error);
        }
    }

```

```

    public static function assignRoute()
    {
        if(isset($_POST['id']) && isset($_POST['route_id']))
        {
            return Delivery::assignRoute($_POST['id'], $_POST['route_id']);
        }
        else
        {
            http_response_code(404);
            $error = array('status'=>'error', 'message'=>'missing params');
            return json_encode($error);
        }
    }

```

```

    public static function getAllPendingDeliveries()
    {

```

```

        return json_encode(Delivery::getAllPendingDeliveries());
    }

    public static function startRouteCalculation()
    {
        $coursiers =json_decode(User::getCouriers());

        return      Delivery::startRouteCalculation(47.81772575251996,
35.18423565166421, $coursiers);
    }
}

?>

```

A4. Текст файла RouteController.php

```

<?php

require_once
$_SERVER['DOCUMENT_ROOT'].'/delivery/model/Route.php';

class RouteController
{
    public static function getAllNewRoutes()
    {
        return Route::getAllNewRoutes();
    }

    public static function getAllActiveRoutes()
    {
        return Route::getAllActiveRoutes();
    }

    public static function getAllRoutesByDate()
    {
        if(isset($_POST['from']) && isset($_POST['to']))
        {
            return Route::getAllRoutesByDate($_POST['from'], $_POST['to']);
        }
        else
        {
            http_response_code(404);
            $error = array('status'=>'error', 'message'=>'missing params');
            return json_encode($error);
        }
    }
}

```

```

public static function getDeliveries()
{
    if(isset($_POST['id']))
    {
        return Route::getDeliveries($_POST['id']);
    }
    else
    {
        http_response_code(404);
        $error = array('status'=>'error', 'message'=>'missing params');
        return json_encode($error);
    }
}

public static function newRoute()
{
    if(isset($_POST['logistician_id']) && isset($_POST['courier_id']))
    {
        return Route::newRoute($_POST['logistician_id'],
$_POST['courier_id']);
    }
    else
    {
        http_response_code(404);
        $error = array('status'=>'error', 'message'=>'missing params');
        return json_encode($error);
    }
}

public static function getActiveRoute()
{
    if(isset($_POST['courier_id']))
    {
        return Route::getActiveRoute($_POST['courier_id']);
    }
    else
    {
        http_response_code(404);
        $error = array('status'=>'error', 'message'=>'missing params');
        return json_encode($error);
    }
}
}
?>

```

A5. Текст файла UserController.php

```

<?php

require_once
$_SERVER['DOCUMENT_ROOT'].'/delivery/model/User.php';

class UserController
{
    public static function login()
    {
        if(isset($_POST['login']) && isset($_POST['password']))
        {
            return User::login($_POST['login'], $_POST['password']);
        }
        else
        {
            http_response_code(404);
            $error = array('status'=>'error', 'message'=>'missing login or
password');
            return json_encode($error);
        }
    }

    public static function authorize()
    {
        if(isset($_POST['id']) && isset($_POST['pin']))
        {
            return User::authorize($_POST['id'], $_POST['pin']);
        }
        else
        {
            http_response_code(404);
            $error = array('status'=>'error', 'message'=>'missing login or
password');
            return json_encode($error);
        }
    }

    public static function registerUser()
    {
        if(isset($_POST['login']) && isset($_POST['password']) &&
isset($_POST['status']))
        {
            return User::registerUser($_POST['login'], $_POST['password']);
        }
    }
}

```

```

    }
    else
    {
        return false;
    }
}

public static function registerCourier()
{
    if(isset($_POST['name']) && isset($_POST['consumption']))
    {
        return User::registerCourier($_POST['name'],
$_POST['consumption']);
    }
    else
    {
        return false;
    }
}

public static function getCouriers()
{
    return User::getCouriers();
}

public static function getUsers()
{
    return User::getUsers();
}
}

?>

```

A6. Текст файла Delivery.php

```

<?php

require_once $_SERVER['DOCUMENT_ROOT'] . '/delivery/db.php';

class Delivery
{
    public static function newDelivery($address, $phone, $lat, $lng,
$datetime)
    {
        $db = DB::get();
    }
}

```

```

        $query = "INSERT INTO deliveries (address, phone, lat, lng,
delivery_time) VALUES (?, ?, ?, ?, ?)";
        $result = $db->prepare($query);
        $result->bindParam(1, $address);
        $result->bindParam(2, $phone);
        $result->bindParam(3, $lat);
        $result->bindParam(4, $lng);
        $result->bindParam(5, $datetime);
        $result->execute();
        return $db->lastInsertId();
    }

```

```

public static function confirmDelivery($id, $lat, $lng)
{
    $datetime = date('Y-m-d H:i:s');

```

```

    $db = DB::get();

```

```

        $query = "UPDATE deliveries SET lat_fixed = ?, lng_fixed = ?,
delivery_time_fixed = ? WHERE id = ?";
        $result = $db->prepare($query);
        $result->bindParam(1, $lat);
        $result->bindParam(2, $lng);
        $result->bindParam(3, $datetime);
        $result->bindParam(4, $id);
        $result->execute();
        return $db->lastInsertId();
    }

```

```

public static function getDelivery($id)
{
    $db = DB::get();

```

```

        $query = "SELECT * FROM deliveries WHERE id = ?";
        $result = $db->prepare($query);
        $result->bindParam(1, $id);
        $result->execute();
        return $result->fetch();
    }

```

```

public static function assignRoute($id, $route_id)
{
    $db = DB::get();

```

```

        $query = "UPDATE deliveries SET route_id = ? WHERE id = ?";
        $result = $db->prepare($query);

```

```

        $result->bindParam(1, $route_id);
        $result->bindParam(2, $id);
        $result->execute();
        return $db->lastInsertId();
    }

    public static function getAllPendingDeliveries()
    {
        $db = DB::get();

        $query = "SELECT * FROM deliveries WHERE delivery_time_fixed
IS NULL";
        $result = $db->prepare($query);
        $result->execute();
        return $result->fetchAll(PDO::FETCH_ASSOC);
    }

    //

    public static function startRouteCalculation($lat, $lng, $couriers)
    {
        $pendingDeliveries = self::getAllPendingDeliveries();

        $routes = array();

        foreach ($couriers as $courier)
        {
            $deliveries = array();

            $distance = 0;
            $selectedIndex = 0;

            for ($i = 0; $i < count($pendingDeliveries); $i++)
            {
                $pendingDelivery = $pendingDeliveries[$i];

                $pendingDistance = self::getDistance($lat, $lng,
                $pendingDelivery['lat'], $pendingDelivery['lng']);

                if($pendingDistance > $distance)
                {
                    $distance = $pendingDistance;
                    $selectedIndex = $i;
                }
            }
        }
    }

```

```

array_push($deliveries, $pendingDeliveries[$selectedIndex]);

$pendingDelivery = $pendingDeliveries[$selectedIndex];

array_splice($pendingDeliveries, $selectedIndex, 1);

array_push($routes, array('deliveries' => $deliveries)); //создать
маршрут по количеству курьеров

for($i = 0; $i < count($pendingDeliveries); $i++)
{
    $candidateRoute = $routes[0];
    $candidateRouteIndex = 0;
    $distance = 360;

    $pendingDeliveryArray =
self::getNearestPoint($pendingDeliveries, $pendingDelivery);
    $pendingDelivery = $pendingDeliveryArray['point'];
    $pendingDeliveryIndex = $pendingDeliveryArray['index'];

    for($j = 0; $j < count($routes); $j++)
    {
        $route = $routes[$j];

        $routeLastDelivery =
$route['deliveries'][count($route['deliveries'])-1];

        $pendingDistance =
self::getDistance($routeLastDelivery['lat'], $routeLastDelivery['lng'],
$pendingDelivery['lat'], $pendingDelivery['lng']);

        if($pendingDistance < $distance)
        {
            $distance = $pendingDistance;
            $candidateRoute = $route;
            $candidateRouteIndex = $j;
        }
    }

    $routeLastDelivery =
$candidateRoute['deliveries'][count($candidateRoute['deliveries'])-1];

    if(!self::checkIfNewRouteNeeded($routeLastDelivery['lat'],
$routeLastDelivery['lng'], $pendingDelivery['lat'], $pendingDelivery['lng'], $lat,
$lng))

```

```

        {
            array_push($routes[$candidateRouteIndex]['deliveries'],
$pendingDelivery);

            array_splice($pendingDeliveries, $pendingDeliveryIndex,
1);
        }
        else
        {
            //echo 'near from origin<br>';
        }
    }
}

self::getActualDistance($lat, $lng, $routes, $couriers);
}

public static function checkIfNewRouteNeeded($point1Lat, $point1Lng,
$point2Lat, $point2Lng, $originLat, $originLng)
{
    $currentDistance = sqrt(pow($point1Lat-$point2Lat, 2) +
pow($point1Lng-$point2Lng, 2));
    $newDistance = sqrt(pow($point1Lat-$originLat, 2) +
pow($point1Lng-$originLng, 2));

    return ($currentDistance > $newDistance);
}

public static function getDistance($lat1, $lng1, $lat2, $lng2)
{
    return sqrt(pow($lat1-$lat2, 2) + pow($lng1-$lng2, 2));
}

public static function getNearestPoint($pendingDeliveries, $origin)
{
    $distance = 360;
    $selectedIndex = 0;

    for ($i = 0; $i < count($pendingDeliveries); $i++)
    {
        $pendingDelivery = $pendingDeliveries[$i];

        $pendingDistance = self::getDistance($origin['lat'], $origin['lng'],
$pendingDelivery['lat'], $pendingDelivery['lng']);

        if($pendingDistance < $distance)

```

```

        {
            $distance = $pendingDistance;
            $selectedIndex = $i;
        }
    }

    return array('index'=>$selectedIndex,
'point'=>$pendingDeliveries[$selectedIndex]);
}

public static function getActualDistance($lat, $lng, $routes, $couriers)
{
    foreach ($routes as $route)
    {
        if(count($route['deliveries']) > 0)
        {
            $requestParams = 'origin=' . $lat . ',' . $lng;

            $requestParams .= '&waypoints=optimize:true';

            foreach ($route['deliveries'] as $delivery)
            {
                $requestParams .= '|' . $delivery['lat'] . ',' . $delivery['lng'];
            }

            $requestParams .= '&destination=' . $lat . ',' . $lng;
            $requestParams .= '&key=API_KEY_HERE';

            $ch = curl_init('https://maps.googleapis.com/maps/api/directions/json?'
            $requestParams);

            curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
            curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
            curl_setopt($ch, CURLOPT_HEADER, false);
            $result = curl_exec($ch);
            curl_close($ch);

            $totalDistance = 0;//rand(10, 20);

            $resultObject = json_decode($result);

            if(count($resultObject->routes) > 0)
            {
                $groutes = $resultObject->routes[0];

                $glegs = $groutes->legs;

```

```

        echo count($glegs);

        foreach ($glegs as $leg)
        {
            $totalDistance += $leg->distance->value;
        }
    }

    $route['totalDistance'] = $totalDistance/1000;
}
}

self::assignCourier($routes, $couriers);
}

public static function assignCourier($routes, $couriers)
{
    usort($routes, function ($a, $b) {
        return $a['totalDistance'] > $b['totalDistance'];
    });

    usort($couriers, function ($a, $b) {
        return $a->consumption < $b->consumption;
    });

    for ($i = 0; $i < count($routes); $i++)
    {
        self::saveCourier($couriers[$i], $routes[$i]['totalDistance']);
    }
}

public static function saveCourier($courier_id, $distance)
{
    $logistician_id = 1;
    $status = 1;

    $db = DB::get();

    $query = "INSERT INTO routes (logistician_id, courier_id, status,
distance) VALUES (?, ?, ?, ?)";
    $result = $db->prepare($query);
    $result->bindParam(1, $logistician_id);
    $result->bindParam(2, $courier_id->id);
    $result->bindParam(3, $status);
    $result->bindParam(4, $distance);

```

```

        $result->execute();
        return true;
    }
}

?>

```

A6. Текст файла Delivery.php

```

<?php

require_once $_SERVER['DOCUMENT_ROOT'].'/delivery/db.php';

class Route
{
    public static function getAllNewRoutes()
    {
        $db = DB::get();

        $query = "SELECT * FROM routes WHERE status = 0";
        $result = $db->prepare($query);
        $result->execute();
        $data = $result->fetchAll(PDO::FETCH_ASSOC);
        return json_encode($data);
    }

    public static function getAllActiveRoutes()
    {
        $db = DB::get();

        $query = "SELECT r.*, c.name, u.login FROM routes r JOIN couriers
c on c.id = r.courier_id JOIN users u ON u.id = r.logistician_id WHERE r.status =
1";

        $result = $db->prepare($query);
        $result->execute();
        $data = $result->fetchAll(PDO::FETCH_ASSOC);
        return json_encode($data);
    }

    public static function getAllRoutesByDate($from, $to)
    {
        $db = DB::get();

        $query = "SELECT * FROM routes WHERE date_added >= $from
AND date_added <= $to";
        $result = $db->prepare($query);
    }
}

```

```

        $result->execute();
        $data = $result->fetchAll(PDO::FETCH_ASSOC);
        return json_encode($data);
    }

    public static function getDeliveries($id)
    {
        $db = DB::get();

        $query = "SELECT * FROM deliveries WHERE route_id = ?";
        $result = $db->prepare($query);
        $result->bindParam(1, $id);
        $result->execute();
        $data = $result->fetchAll(PDO::FETCH_ASSOC);

        return json_encode($data);
    }

    public static function newRoute($logistician_id, $courier_id)
    {
        $db = DB::get();

        $query = "INSERT INTO routes (logistician_id, courier_id) VALUES
(?, ?)";
        $result = $db->prepare($query);
        $result->bindParam(1, $logistician_id);
        $result->bindParam(2, $courier_id);
        $result->execute();
        return $db->lastInsertId();
    }

    public static function getActiveRoute($courier_id)
    {
        $db = DB::get();

        $query = "SELECT * FROM routes WHERE courier_id = ? LIMIT
1";
        $result = $db->prepare($query);
        $result->bindParam(1, $courier_id);
        $result->execute();
        $data = $result->fetch();
        return json_encode($data);
    }
}

?>

```

A7. Текст файла User.php

```

<?php

require_once $_SERVER['DOCUMENT_ROOT'].'/delivery/db.php';

class User
{
    public static function login($login, $password)
    {
        $password = md5($password);

        $db = DB::get();

        $query = "SELECT * FROM users WHERE login = ? AND password
= ?";

        $result = $db->prepare($query);
        $result->bindParam(1, $login);
        $result->bindParam(2, $password);
        $result->execute();
        if($result->rowCount() > 0)
        {
            $row = $result->fetch();
            return $row['id'];
        }
        else
        {
            http_response_code(404);
            $error = array('status'=>'error', 'message'=>'not found');
            return json_encode($error);
        }
    }

    public static function authorize($id, $pin)
    {
        $db = DB::get();

        $query = "SELECT * FROM couriers WHERE id = ? AND pin = ?";
        $result = $db->prepare($query);
        $result->bindParam(1, $id);
        $result->bindParam(2, $pin);
        $result->execute();
        if($result->rowCount() > 0)
        {
            $row = $result->fetch();

```

```

        return $row['id'];
    }
    else
    {
        http_response_code(404);
        $error = array('status'=>'error', 'message'=>'not found');
        return json_encode($error);
    }
}

public static function registerUser($login, $password)
{
    $password = md5($password);

    $db = DB::get();

    $query = "SELECT * FROM users WHERE login = ?";
    $result = $db->prepare($query);
    $result->bindParam(1, $login);
    $result->execute();
    if($result->fetchAll(PDO::FETCH_ASSOC) == false)
    {
        $query = "INSERT INTO users (login, password) VALUES (?, ?)";
        $result = $db->prepare($query);
        $result->bindParam(1, $login);
        $result->bindParam(2, $password);
        $result->execute();
        return true;
    }
    else
    {
        return false;
    }
}

public static function registerCourier($name, $consumption)
{
    $pin = rand(1000, 9999);

    $db = DB::get();

    $query = "INSERT INTO couriers (name, pin, consumption)
VALUES (?, ?, ?)";
    $result = $db->prepare($query);
    $result->bindParam(1, $name);
    $result->bindParam(2, $pin);

```

```

        $result->bindParam(3, $consumption);
        $result->execute();
        return true;
    }

    public static function getCouriers()
    {
        $db = DB::get();

        $query = "SELECT * FROM couriers";
        $result = $db->prepare($query);
        $result->execute();
        $data = $result->fetchAll(PDO::FETCH_ASSOC);
        return json_encode($data);
    }

    public static function getUsers()
    {
        $db = DB::get();

        $query = "SELECT * FROM users";
        $result = $db->prepare($query);
        $result->execute();
        $data = $result->fetchAll(PDO::FETCH_ASSOC);
        return json_encode($data);
    }
}

?>

```

A8. Текст файла index.php

```

<html lang="">
<head>
    <title></title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
1BmE4kWBq78iYhFtdvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jI
W3" crossorigin="anonymous">
    </head>
<?php

if(!$_COOKIE['login'])
{
    header('Location: login.php');
}

```

```

        die();
    }

    require_once
$_SERVER['DOCUMENT_ROOT'].'/delivery/controller/RouteController.php';

?>
<div class="container-fluid">
    <div style="margin-top: 17%" class="row">
        <div class="col-3">
            <div class="card" style="width: 18rem;">
                
                <div class="card-body text-center">
                    <a class="btn btn-primary" href="couriers.php">Couriers</a>
                </div>
            </div>
        </div>
        <div class="col-3">
            <div class="card" style="width: 18rem;">
                
                <div class="card-body text-center">
                    <a class="btn btn-primary" href="users.php">Users</a>
                </div>
            </div>
        </div>
        <div class="col-3">
            <div class="card" style="width: 18rem;">
                
                <div class="card-body text-center">
                    <a
                        class="btn
                        btn-primary"
href="deliveries.php">Deliveries</a>
                </div>
            </div>
        </div>
        <div class="col-3">
            <div class="card" style="width: 18rem;">
                
                <div class="card-body text-center">
                    <a class="btn btn-primary" href="routes.php">Routes</a>
                </div>
            </div>
        </div>
    </div>

```

```

    </div>
</div>
</html>

```

A9. Текст файла couriers.php

```

<html lang="">
<head>
    <title></title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet"                                integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jI
W3" crossorigin="anonymous">
    </head>
<?php

if(!$_COOKIE['login'])
{
    header('Location: login.php');
    die();
}

require_once
$_SERVER['DOCUMENT_ROOT'].'/delivery/controller/UserController.php';
require_once
$_SERVER['DOCUMENT_ROOT'].'/delivery/model/User.php';

if(isset($_POST['name']))
{
    User::registerCourier($_POST['name'], $_POST['consumption']);
    header('Location: couriers.php');
}

$couriers = json_decode(UserController::getCouriers());

?>
<div class="container-fluid">
    <a class="btn-sm bi bi-arrow-90deg-up my-5 btn btn-danger"
href="index.php">
        <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
fill="currentColor" class="bi bi-arrow-90deg-up" viewBox="0 0 16 16">
            <path fill-rule="evenodd" d="M4.854 1.146a.5.5 0 0 0-.708 0l-4
4a.5.5 0 1 0 .708.708L4.27 12.5A2.5 2.5 0 0 0 6.5 15h8a.5.5 0 0 0 .708
.708l-4-4a.5.5 0 1 0-.708-.708l-4-4z"/>

```

```

        </svg>
        Main</a>
<table class="table">
<thead>
<tr class="table-dark">
    <th>ID</th>
    <th>Name</th>
    <th>PIN</th>
</tr>
</thead>
<tbody>
<?php
foreach ($couriers as $courier)
{
    ?>
<tr>
    <td><? echo $courier->id ?></td>
    <td><? echo $courier->name ?></td>
    <td><? echo $courier->pin ?></td>
</tr>
<?php
}
?>
</tbody>
</table>
<div class="text-center my-4">
<a class="bi bi-person-plus btn btn-primary" href="newCourier.php">
    <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
fill="currentColor" class="bi bi-person-plus" viewBox="0 0 16 16">
        <path d="M6 8a3 3 0 1 0 0-6 3 3 0 0 0 6zm2-3a2 2 0 1 1-4 0 2 2 0 0
1 4 0zm4 8c0 1-1 1-1 1H1s-1 0-1-1 1-4 6-4 6 3 6 4zm-1-.004c-.001-.246-.154-
.986-.832-1.664C9.516 10.68 8.289 10 6 10c-2.29 0-3.516.68-4.168 1.332-
.678.678-.83 1.418-.832 1.664h10z"/>
        <path fill-rule="evenodd" d="M13.5 5a.5.5 0 0 1 .5.5V7h1.5a.5.5 0 0
1 0 1H14v1.5a.5.5 0 0 1-1 1 0V8h-1.5a.5.5 0 0 1 0-1H13V5.5a.5.5 0 0 1 .5-.5z"/>
    </svg>
    Register new courier</a>
</div>
</html>

```

A10. Текст файла deliveries.php

```

<html lang="">
<head>
    <title></title>

```

```

        <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet"                                integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jI
W3" crossorigin="anonymous">
    </head>
    <?php

    if(!$_COOKIE['login'])
    {
        header('Location: login.php');
        die();
    }

    require_once
$_SERVER['DOCUMENT_ROOT'].'/delivery/controller/DeliveryController.php';
    require_once
$_SERVER['DOCUMENT_ROOT'].'/delivery/model/Delivery.php';

    if(isset($_POST['address']))
    {
        Delivery::newDelivery($_POST['address'],            $_POST['phone'],
$_POST['lat'], $_POST['lng'], $_POST['date']);
        header('Location: deliveries.php');
    }

    $deliveries = json_decode(DeliveryController::getAllPendingDeliveries());

    ?>
    <div class="container-fluid">
        <a class="btn-sm bi bi-arrow-90deg-up my-5 btn btn-danger"
href="index.php">
            <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
fill="currentColor" class="bi bi-arrow-90deg-up" viewBox="0 0 16 16">
                <path fill-rule="evenodd" d="M4.854 1.146a.5.5 0 0 0-.708
0l-4a.5.5 0 1 0 .708.708L4 2.707V12.5A2.5 2.5 0 0 0 6.5 15h8a.5.5 0 0 0
0-1h-8A1.5 1.5 0 0 1 5 12.5V2.707l3.146 3.147a.5.5 0 1 0 .708-.708l-4-4z"/>
            </svg>
            Main</a>
        <table class="table">
            <thead>
            <tr class="table-dark">
                <th>Address</th>
                <th>Phone</th>
                <th>Date/time</th>
            </tr>
        </table>
    </div>

```

```

</tr>
</thead>
<tbody>
<?php

foreach ($deliveries as $delivery)
{
    ?>
    <tr>
        <td><? echo $delivery->address ?></td>
        <td><? echo $delivery->phone ?></td>
        <td><? echo $delivery->delivery_time ?></td>
    </tr>
    <?php
}
?>
</tbody>
</table>
<div class="text-center my-4">
    <a class="btn btn-primary" href="newDelivery.php">
        <svg          xmlns="http://www.w3.org/2000/svg"          width="16"
height="16" fill="currentColor" class="bi bi-truck" viewBox="0 0 16 16">
            <path d="M0 3.5A1.5 1.5 0 0 1 1.5 2h9A1.5 1.5 0 0 1 12
3.5V5h1.02a1.5 1.5 0 0 1 1.17.563l1.481 1.85a1.5 1.5 0 0 1 .329.938V10.5a1.5 1.5
0 0 1-1.5 1.5H14a2 2 0 1 1-4 0H5a2 2 0 1 1-3.998-.085A1.5 1.5 0 0 1 0 10.5v-
7zm1.294 7.456A1.999 1.999 0 0 1 4.732 11h5.536a2.01 2.01 0 0 1 .732-
.732V3.5a.5.5 0 0 0-.5-.5h-9a.5.5 0 0 0-.5.5v7a.5.5 0 0 0 .294.456zM12 10a2 2 0 0
1 1.732 1h.768a.5.5 0 0 0 .5-.5V8.35a.5.5 0 0 0-.11-.312l-1.48-1.85A.5.5 0 0 0
13.02 6H12v4zm-9 1a1 1 0 1 0 0 2 1 1 0 0 0 0-2zm9 0a1 1 0 1 0 0 2 1 1 0 0 0 0-
2z"/>
            </svg>
            Create new delivery</a>
    </div>
</div>
<script src="newdelivery.js"></script>
</html>

```

A11. Текст файла login.php

```

<html lang="">
<head>
    <title></title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet"                                integrity="sha384-

```

```

1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jI
W3" crossorigin="anonymous">
    </head>

    <?php

    if($_COOKIE['login'])
    {
        header('Location: index.php');
    }

    ?>

    <div class="container-fluid">
        <div style="margin-top: 13%" class="mx-auto">
            <input class="form-control" type="text" id="login"
placeholder="Логин">
            <input class="form-control" type="password" id="password"
placeholder="Пароль">
            <button class="btn btn-success" onclick="login()">Login</button>
        </div>

    </div>

    <script src="login.js"></script>
</html>

```

A12. Текст файла login.js

```

function login()
{
    let login = document.getElementById("login").value;
    let password = document.getElementById("password").value;

    let xhr = new XMLHttpRequest();
    xhr.open("POST", 'https://ruslan.inf.ua/delivery/api.php?action=login',
true);

    xhr.setRequestHeader("Content-type", "application/x-www-form-
urlencoded");

    xhr.onreadystatechange = function()
    {
        if(xhr.readyState === XMLHttpRequest.DONE && xhr.status ===
200)

```



```

        <span class="input-group-text" id="inputGroup-sizing-
default">Consumption</span>
        <input class="form-control" type="number"
name="consumption" placeholder="0.0">
    </div>
    <div class="text-center">
        <input class="btn btn-success" type="submit">
    </div>
</form>
</div>
<div class="col-4"></div>
</div>
</div>
</html>

```

A14. Текст файла newDelivery.php

```

<html lang="">
<head>
    <title></title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jI
W3" crossorigin="anonymous">
        <script type="text/javascript"
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyCMs85TJaFC3Sy-
eeTrwWVtiLeiq3BeqxE&callback=initAutocomplete&libraries=places&v=weekly
"></script>
    </head>

    <div class="container-fluid">
        <div style="margin-top: 13%" class="row">
            <div class="col-4"></div>
            <div class="col-4 mx-auto">
                <div class="text-center">
                    <svg xmlns="http://www.w3.org/2000/svg" width="100"
height="100" fill="currentColor" class="bi bi-truck" viewBox="0 0 16 16">
                        <path d="M0 3.5A1.5 1.5 0 0 1 1.5 2h9A1.5 1.5 0 0 1 12
3.5V5h1.02a1.5 1.5 0 0 1 1.17.563l1.481 1.85a1.5 1.5 0 0 1 .329.938V10.5a1.5 1.5
0 0 1-1.5 1.5H14a2 2 0 1 1-4 0H5a2 2 0 1 1-3.998-.085A1.5 1.5 0 0 1 0 10.5v-
7zm1.294 7.456A1.999 1.999 0 0 1 4.732 11h5.536a2.01 2.01 0 0 1 .732-
.732V3.5a.5.5 0 0 0-.5-.5h-9a.5.5 0 0 0-.5v7a.5.5 0 0 0 .294.456zM12 10a2 2 0 0
1 1.732 1h.768a.5.5 0 0 0 .5-.5V8.35a.5.5 0 0 0-.11-.312l-1.48-1.85A.5.5 0 0 0
13.02 6H12v4zm-9 1a1 1 0 1 0 0 2 1 1 0 0 0 0-2zm9 0a1 1 0 1 0 0 2 1 1 0 0 0 0-
2z"/>

```

```

        </svg>
    </div>
    <form class="mt-2" action="deliveries.php" target="_self"
method="post">
        <div class="input-group mb-3">
            <span class="input-group-text" id="inputGroup-sizing-
default">Address</span>
            <input class="form-control" type="text" name="address"
placeholder="Shkilna 4">
        </div>
        <div class="input-group mb-3">
            <span class="input-group-text" id="inputGroup-sizing-
default">Phone</span>
            <input class="form-control" type="tel" name="phone"
placeholder="+380XXXXXXXXXX">
        </div>
        <div class="input-group mb-3">
            <span class="input-group-text" id="inputGroup-sizing-
default">Date</span>
            <input class="form-control" type="datetime-local"
name="date">
        </div>
        <div class="text-center">
            <input id="lat" class="form-control" type="hidden"
name="lat">
            <input id="lng" class="form-control" type="hidden"
name="lng">
            <input class="btn btn-success" type="submit">
        </div>
    </form>
</div>
<div class="col-4"></div>
</div>
</div>
</html>

```

A15. Текст файлу newdelivery.js

```

function initAutocomplete()
{
    var addressIField = document.getElementById('address');

    var autocomplete = new
google.maps.places.Autocomplete(addressIField, {
    componentRestrictions: {country: ["ua"]},
    fields: ["address_components", "geometry"],

```

```

        types: ['address'],
        bounds: defaultBounds,
        strictBounds: true
    });
    address1Field.focus();
    autocomplete.addListener("place_changed", fillInAddress);
}

function fillInAddress() {
    const place = autocomplete.getPlace();

    address1Field.value = address1;
    document.getElementById("lat").value = place.geometry.location.lat();
    document.getElementById("lng").value = place.geometry.location.lng();
}

```

A15. Текст файла newUser.php

```

<html lang="">
<head>
    <title></title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jI
W3" crossorigin="anonymous">
    </head>

    <div class="container-fluid">
        <div style="margin-top: 13%" class="row">
            <div class="col-4"></div>
            <div class="col-4 mx-auto">
                <div class="text-center">
                    <svg xmlns="http://www.w3.org/2000/svg" width="100"
height="100" fill="currentColor" class="bi bi-person-plus" viewBox="0 0 16 16">
                        <path d="M6 8a3 3 0 1 0 0-6 3 3 0 0 0 6 0 0 0 0 6zm2-3a2 2 0 1 1-4 0
2 2 0 0 1 4 0zm4 8c0 1-1 1-1 1H1s-1 0-1 1 1-4 6-4 6 3 6 4zm-1-.004c-.001-.246-
.154-.986-.832-1.664C9.516 10.68 8.289 10 6 10c-2.29 0-3.516.68-4.168 1.332-
.678.678-.83 1.418-.832 1.664h10z"/>
                        <path fill-rule="evenodd" d="M13.5 5a.5.5 0 0 1
.5.5V7h1.5a.5.5 0 0 1 0 1 1H14v1.5a.5.5 0 0 1-1 1 0V8h-1.5a.5.5 0 0 1 0-
1H13V5.5a.5.5 0 0 1 .5-.5z"/>
                    </svg>
                </div>
            </div>
        </div>
    </div>

```

```

        <form      class="mt-2"      action="users.php"      target="_self"
method="post">
            <div class="input-group mb-3">
                <span      class="input-group-text"      id="inputGroup-sizing-
default">Login</span>
                <input      class="form-control"      type="text"      name="login"
placeholder="Login">
            </div>
            <div class="input-group mb-3">
                <span      class="input-group-text"      id="inputGroup-sizing-
default">Password</span>
                <input      class="form-control"      type="password"
name="password" placeholder="*****">
            </div>
            <div class="text-center">
                <input class="btn btn-success" type="submit">
            </div>
        </form>
    </div>
    <div class="col-4"></div>
</div>
</html>

```

A16. Текст файла routes.php

```

<html lang="">
<head>
    <title></title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet"                                integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jI
W3" crossorigin="anonymous">
    </head>
<?php

if(!$_COOKIE['login'])
{
    header('Location: login.php');
    die();
}

require_once
$_SERVER['DOCUMENT_ROOT'].'/delivery/controller/RouteController.php';

```

```

        require_once
$_SERVER['DOCUMENT_ROOT'].'/delivery/model/Route.php';
        require_once
$_SERVER['DOCUMENT_ROOT'].'/delivery/controller/UserController.php';
        require_once
$_SERVER['DOCUMENT_ROOT'].'/delivery/model/User.php';
        require_once
$_SERVER['DOCUMENT_ROOT'].'/delivery/controller/DeliveryController.php';
        require_once
$_SERVER['DOCUMENT_ROOT'].'/delivery/model/Delivery.php';

        if(isset($_POST['new']))
        {
            $coursiers = UserController::getCouriers();

            Delivery::startRouteCalculation(47.81772575251996,
35.18423565166421, $coursiers);
            header('Location: routes.php');
        }

        $routes = json_decode(RouteController::getAllActiveRoutes());

?>
<div class="container-fluid">
    <a class="btn-sm bi bi-arrow-90deg-up my-5 btn btn-danger"
href="index.php">
        <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
fill="currentColor" class="bi bi-arrow-90deg-up" viewBox="0 0 16 16">
            <path fill-rule="evenodd" d="M4.854 1.146a.5.5 0 0 0-.708 0l-4
4a.5.5 0 1 0 .708.708L4 2.707V12.5A2.5 2.5 0 0 0 6.5 15h8a.5.5 0 0 0 0-1h-8A1.5
1.5 0 0 1 5 12.5V2.707l3.146 3.147a.5.5 0 1 0 .708-.708l-4-4z"/>
        </svg>
        Main</a>
    <table class="table">
    <thead>
    <tr class="table-dark">
        <th>ID</th>
        <th>Distance</th>
        <th>Courier</th>
        <th>Logistician</th>
    </tr>
    </thead>
    <tbody>
    <?php
    foreach ($routes as $route)

```

```

{
    ?>
    <tr>
        <td><? echo $route->id ?></td>
        <td><? echo $route->distance ?></td>
        <td><? echo $route->name ?></td>
        <td><? echo $route->login ?></td>
    </tr>
    <?php
}
?>
</tbody>
</table>
<div class="text-center my-4">
    <form target="_self" method="post">
        <input type="hidden" name="new">
        <input class="btn btn-success" type="submit">
    </form>
</div>

</div>
</html>

```

A17. Текст файла users.php

```

<html lang="">
<head>
    <title></title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jI
W3" crossorigin="anonymous">
    </head>
    <?php

    if(!$_COOKIE['login'])
    {
        header('Location: login.php');
        die();
    }

    require_once
$_SERVER['DOCUMENT_ROOT'].'/delivery/controller/UserController.php';
    require_once
$_SERVER['DOCUMENT_ROOT'].'/delivery/model/User.php';

```

```

if(isset($_POST['login']))
{
    User::registerUser($_POST['login'], $_POST['password']);
    header('Location: users.php');
}

$users = json_decode(UserController::getUsers());

?>
<div class="container-fluid">
    <a class="btn-sm bi bi-arrow-90deg-up my-5 btn btn-danger"
href="index.php">
        <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
fill="currentColor" class="bi bi-arrow-90deg-up" viewBox="0 0 16 16">
            <path fill-rule="evenodd" d="M4.854 1.146a.5.5 0 0 0-.708 0l-4
4a.5.5 0 1 0 .708.708L4 2.707V12.5A2.5 2.5 0 0 0 6.5 15h8a.5.5 0 0 0 0-1h-8A1.5
1.5 0 0 1 5 12.5V2.707l3.146 3.147a.5.5 0 1 0 .708-.708l-4-4z"/>
        </svg>
        Main</a>
    <table class="table">
        <thead>
            <tr class="table-dark">
                <th>Login</th>
                <th>Location</th>
            </tr>
        </thead>
        <tbody>
            <?php
foreach ($users as $user)
{
    ?>
    <tr>
        <td><? echo $user->login ?></td>
        <td><? echo $user->lat . ',' . $user->lng?></td>
    </tr>
    <?php
}
?>
        </tbody>
    </table>
    <div class="text-center my-4">
        <a class="bi bi-person-plus btn btn-primary" href="newUser.php">
            <svg
                xmlns="http://www.w3.org/2000/svg"
                width="16"
height="16" fill="currentColor" class="bi bi-person-plus" viewBox="0 0 16 16">

```

```

        <path d="M6 8a3 3 0 1 0 0-6 3 3 0 0 0 6zm2-3a2 2 0 1 1-4 0 2
2 0 0 1 4 0zm4 8c0 1-1 1-1 1H1s-1 0-1-1 1-4 6-4 6 3 6 4zm-1-.004c-.001-.246-
.154-.986-.832-1.664C9.516 10.68 8.289 10 6 10c-2.29 0-3.516.68-4.168 1.332-
.678.678-.83 1.418-.832 1.664h10z"/>
        <path fill-rule="evenodd" d="M13.5 5a.5.5 0 0 1 .5.5V7h1.5a.5.5
0 0 1 0 1H14v1.5a.5.5 0 0 1-1 1 0V8h-1.5a.5.5 0 0 1 0-1H13V5.5a.5.5 0 0 1 .5-
.5z"/>
    </svg>
    Register new user</a>
</div>
</div>
</html>

```

A18. Текст файла AndroidManifest.xml

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="ua.com.ruslantereshchenko.delivery">

    <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Delivery">
        <activity
            android:name=".LoginActivity"
            android:screenOrientation="portrait"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER"
/>

            </intent-filter>
        </activity>
        <activity
            android:name=".MainActivity"
            android:screenOrientation="portrait"

```

```

        android:exported="true">
    </activity>
</application>

</manifest>

```

A19. Текст файла LoginActivity.java

```

package ua.com.ruslantereshchenko.delivery;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;

import java.util.HashMap;
import java.util.Map;

public class LoginActivity extends AppCompatActivity
{
    SharedPreferences settings;

    EditText idEditText;
    EditText pinEditText;

    Button loginButton;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        settings = getSharedPreferences("delivery", 0);

        if(!settings.getString("id", "0").equals("0"))

```

```

    {
        Intent i = new Intent(getApplicationContext(), MainActivity.class);
        startActivity(i);
    }

    idEditText = findViewById(R.id.idEditText);
    pinEditText = findViewById(R.id.pinEditText);

    loginButton = findViewById(R.id.loginButton);

    loginButton.setOnClickListener(view -> authorize());
}

void authorize()
{
    RequestQueue queue =
Volley.newRequestQueue(getApplicationContext());

    String url = "https://ruslan.inf.ua/delivery/api.php?action=authorize";
    StringRequest stringRequest = new
StringRequest(Request.Method.POST, url, response -> {
    settings.edit().putString("id",
idEditText.getText().toString()).apply();
    Intent i = new Intent(getApplicationContext(), MainActivity.class);
    startActivity(i);
}, error -> {
    AlertDialog.Builder builder = new
AlertDialog.Builder(LoginActivity.this);
    builder.setTitle("Error")
        .setMessage("Invalid data")
        .setCancelable(false)
        .setNegativeButton("OK", (dialog, which) -> {
        });
    AlertDialog alert = builder.create();
    alert.show();
})) {
    protected Map<String, String> getParams() {
        Map<String, String> MyData = new HashMap<String, String>();
        MyData.put("id", idEditText.getText().toString());
        MyData.put("pin", pinEditText.getText().toString());
        return MyData;
    }
};

queue.add(stringRequest);
}

```

```
}
```

20. Текст файла MainActivity.java

```
package ua.com.ruslantereshchenko.delivery;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

import android.Manifest;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.net.Uri;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.HashMap;
import java.util.Map;

public class MainActivity extends AppCompatActivity
{
    SharedPreferences settings;

    JSONArray deliveries = new JSONArray();

    TextView addressTextView;
```

```

Button callButton;
Button navigateButton;
Button doneButton;

```

```

String id = "";
String phone = "";
String lat = "";
String lng = "";

```

```

LocationManager locationManager;
LocationListener locationListener;
Boolean locationProvided = false;
String currentLocation = "";

```

```

@Override
public void onBackPressed()
{
    super.onBackPressed();

    finishAffinity();
}

```

```

@Override
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    settings = getSharedPreferences("delivery", 0);

    addressTextView = findViewById(R.id.addressTextView);
    callButton = findViewById(R.id.callButton);
    navigateButton = findViewById(R.id.navigateButton);
    doneButton = findViewById(R.id.doneButton);

    callButton.setOnClickListener(view -> callAction(phone));
    navigateButton.setOnClickListener(view -> navigateAction(lat, lng));
    doneButton.setOnClickListener(view -> doneAction());

    getCurrentRoute();

    locationListener = new LocationListener() {
        @Override
        public void onLocationChanged(@NonNull Location location) {
            currentLocation = location.getLatitude() + "," +
location.getLongitude();

```

```

    }

    @Override
    public void onProviderEnabled(@NonNull String provider) {
        locationProvided = true;
    }

    @Override
    public void onProviderDisabled(@NonNull String provider) {
        locationProvided = false;

        getLocation();
    }

    @Override
    public void onStatusChanged(String provider, int status, Bundle
extras) {
        //Toast.makeText(MainActivity.this, "onStatusChanged",
        Toast.LENGTH_LONG).show();
    }
};

    getLocation();
    cancelLocation();
}

    void getCurrentRoute()
    {
        RequestQueue queue =
Volley.newRequestQueue(getApplicationContext());

        String url =
"https://ruslan.inf.ua/delivery/api.php?action=getactiveroute";
        StringRequest stringRequest = new
StringRequest(Request.Method.POST, url, response -> {
            try
            {
                JSONObject result = new JSONObject(response);
                getDeliveriesInRoute(result.getString("id"));
            }
            catch (JSONException e)
            {

            }

        }, error -> {

```

```

        AlertDialog.Builder builder = new
AlertDialog.Builder(MainActivity.this);
        builder.setTitle("Error")
            .setMessage("Invalid data")
            .setCancelable(false)
            .setNegativeButton("OK", (dialog, which) -> {
                });
        AlertDialog alert = builder.create();
        alert.show();
    }) {
        protected Map<String, String> getParams() {
            Map<String, String> params = new HashMap<String, String>();
            params.put("courier_id", settings.getString("id", "0"));
            return params;
        }
    };

    queue.add(stringRequest);
}

void getDeliveriesInRoute(String route_id)
{
    RequestQueue queue =
Volley.newRequestQueue(getApplicationContext());

    String url =
"https://ruslan.inf.ua/delivery/api.php?action=getdeliveries";
    StringRequest stringRequest = new
StringRequest(Request.Method.POST, url, response -> {
        Log.i("getDeliveriesInRoute", response);
        try
        {
            deliveries = new JSONArray(response);

            for(int i = 0; i < deliveries.length(); i++)
            {
                if(!deliveries.getJSONObject(i).isNull("delivery_time_fixed"))
                {
                    Log.i("delivery_time_fixed", "removed");
                    deliveries.remove(i);
                    i = i-1;
                }
            }
        }

        showDelivery();
    })
}

```

```

        catch (JSONException e)
        {
            Log.i("delivery_time_fixed", "ex");
        }
    }, error -> {
        AlertDialog.Builder builder = new
AlertDialog.Builder(MainActivity.this);
        builder.setTitle("Error")
            .setMessage("Invalid data")
            .setCancelable(false)
            .setNegativeButton("OK", (dialog, which) -> {
                });
        AlertDialog alert = builder.create();
        alert.show();
    }) {
        protected Map<String, String> getParams() {
            Map<String, String> params = new HashMap<String, String>();
            params.put("id", route_id);
            return params;
        }
    };

    queue.add(stringRequest);
}

void showDelivery()
{
    if(deliveries.length() > 0)
    {
        try {
            addressTextView.setText("Address: " +
deliveries.getJSONObject(0).getString("address"));
            id = deliveries.getJSONObject(0).getString("id");
            phone = deliveries.getJSONObject(0).getString("phone");
            lat = deliveries.getJSONObject(0).getString("lat");
            lng = deliveries.getJSONObject(0).getString("lng");

            addressTextView.setVisibility(View.VISIBLE);
            callButton.setVisibility(View.VISIBLE);
            navigateButton.setVisibility(View.VISIBLE);
            doneButton.setVisibility(View.VISIBLE);
        }
        catch (JSONException e)
        {
        }
    }
}

```

```

    }
    else
    {
        addressTextView.setVisibility(View.INVISIBLE);
        callButton.setVisibility(View.INVISIBLE);
        navigateButton.setVisibility(View.INVISIBLE);
        doneButton.setVisibility(View.INVISIBLE);
    }
}

void callAction(String phone)
{
    Intent intent = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:" +
phone));
    startActivity(intent);
}

void navigateAction(String lat, String lng)
{
    Intent intent = new Intent(Intent.ACTION_VIEW,
Uri.parse("http://maps.google.com/maps?daddr=" + lat + "," + lng));
    startActivity(intent);
}

void doneAction()
{
    getLocation();

    while (currentLocation.length() == 0)
    {
        try {
            wait(10000);
        } catch (InterruptedException e) {
            //e.printStackTrace();
        }
    }
}

RequestQueue queue =
Volley.newRequestQueue(getApplicationContext());

String url =
"https://ruslan.inf.ua/delivery/api.php?action=confirmdelivery";
StringRequest stringRequest = new
StringRequest(Request.Method.POST, url, response -> {
    deliveries.remove(0);
    cancelLocation();
}

```

```

        showDelivery();
    }, error -> {
        AlertDialog.Builder builder = new
AlertDialog.Builder(MainActivity.this);
        builder.setTitle("Error")
            .setMessage("Invalid data")
            .setCancelable(false)
            .setNegativeButton("OK", (dialog, which) -> {
                });
        AlertDialog alert = builder.create();
        alert.show();
    }) {
        protected Map<String, String> getParams() {
            Map<String, String> params = new HashMap<String, String>();
            params.put("id", id);
            params.put("lat", currentLocation.split(",")[0]);
            params.put("lng", currentLocation.split(",")[1]);
            return params;
        }
    };

    queue.add(stringRequest);
}

void getLocation()
{
    locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
    if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION)
!=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION)
!=
PackageManager.PERMISSION_GRANTED) {
        if
(ActivityCompat.shouldShowRequestPermissionRationale(MainActivity.this,
Manifest.permission.ACCESS_FINE_LOCATION)) {
            ActivityCompat.requestPermissions(MainActivity.this, new
String[] {Manifest.permission.ACCESS_FINE_LOCATION}, 801);
            ActivityCompat.requestPermissions(MainActivity.this, new
String[] {Manifest.permission.ACCESS_COARSE_LOCATION}, 801);
        } else {
            ActivityCompat.requestPermissions(MainActivity.this, new
String[] {Manifest.permission.ACCESS_FINE_LOCATION}, 801);
            ActivityCompat.requestPermissions(MainActivity.this, new
String[] {Manifest.permission.ACCESS_COARSE_LOCATION}, 801);

```

```

        }

        locationProvided = false;
    } else {

locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDED, 0, 0, locationListener);

        locationProvided = true;
    }
}

void cancelLocation()
{
    if(locationManager != null && locationListener != null)
    {
        locationManager.removeUpdates(locationListener);

        locationProvided = false;
    }
}

@Override
public void onRequestPermissionsResult(int requestCode, String
permissions[], int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions,
grantResults);

    switch (requestCode) {
        case 801: {
            if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                locationProvided = true;

                if (ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED) {

locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDED, 0, 0, locationListener);

                    locationProvided = true;
                }
            } else {
                locationProvided = false;
            }
        }
    }
}

```

```

        return;
    }

}

}
}

```

A21. Текст файла activity_login.xml

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".LoginActivity">

<EditText
    android:id="@+id/idEditText"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:ems="10"
    android:hint="ID"
    android:inputType="number"
    android:minHeight="48dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<EditText
    android:id="@+id/pinEditText"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:ems="10"
    android:hint="PIN"
    android:inputType="number"
    android:minHeight="48dp"
    app:layout_constraintEnd_toEndOf="parent"

```

```

app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/idEditText" />

```

```

<Button
    android:id="@+id/loginButton"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:text="Login"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>

```

A22. Текст файла activity_main.xml

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

```

```

<TextView
    android:id="@+id/addressTextView"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:text="Address: "
    android:visibility="invisible"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

```

```

<LinearLayout
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="4dp"
    app:layout_constraintBottom_toTopOf="@+id/doneButton"

```

```

app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent">

```

```

<Button
    android:id="@+id/callButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:padding="4dp"
    android:text="Call"
    android:visibility="invisible" />

```

```

<Button
    android:id="@+id/navigateButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:padding="4dp"
    android:text="Navigate"
    android:visibility="invisible" />

```

```

</LinearLayout>

```

```

<Button
    android:id="@+id/doneButton"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="4dp"
    android:layout_marginEnd="4dp"
    android:layout_marginBottom="4dp"
    android:text="Delivered"
    android:visibility="invisible"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>

```

ДОДАТОК Б
Слайди презентації

Національний університет «Запорізька політехніка»

1

Кафедра програмних засобів
Випускна робота магістра

ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ВІДСЛІДКУВАННЯ ДОСТАВКИ ЗАМОВЛЕНЬ КУР'ЄРСЬКОЮ СЛУЖБОЮ

Виконав: студент групи КНТ-110м Терещенко Руслан Максимович
Керівник: доцент Степаненко Олександр Олексійович
2021

Рисунок Б.1 - Слайд 1

Мета, об'єкт, предмет і задачі дослідження

2

Об'єкт дослідження – розрахунок оптимального маршруту доставки замовлень кур'єрською службою та контроль часу та коректності місця доставки.

Предмет дослідження – програмні системи керування логістикою служб доставки.

Мета роботи – створення програмної системи для розрахунку оптимального маршруту доставки замовлень кур'єрською службою та мобільного додатку для контролю коректності часу та місцезнаходження доставки замовлення.

Рисунок Б.2 - Слайд 2

Методи пошуку оптимального маршруту			3
Метод	Переваги	Недоліки	
Метод повного перебору	Найбільша точність	Вимогливий до ресурсів при виконанні, що впливає на час виконання алгоритму	
Метод мінімального остовного дерева	Висока точність	Необхідно заздалегідь знати дистанцію між об'єднаними точками, що неможливо отримати без попереднього використання інших методів	
Метод найближчого сусіда	Достатньо висока точність	Може видавати неоптимальний кульчевий маршрут	

Модифікація методу найближчого сусіда дозволяє збільшити точність оптимального маршруту за рахунок розподілення маршрутів між кур'єрами

Рисунок Б.3 - Слайд 3



Рисунок Б.4 - Слайд 4

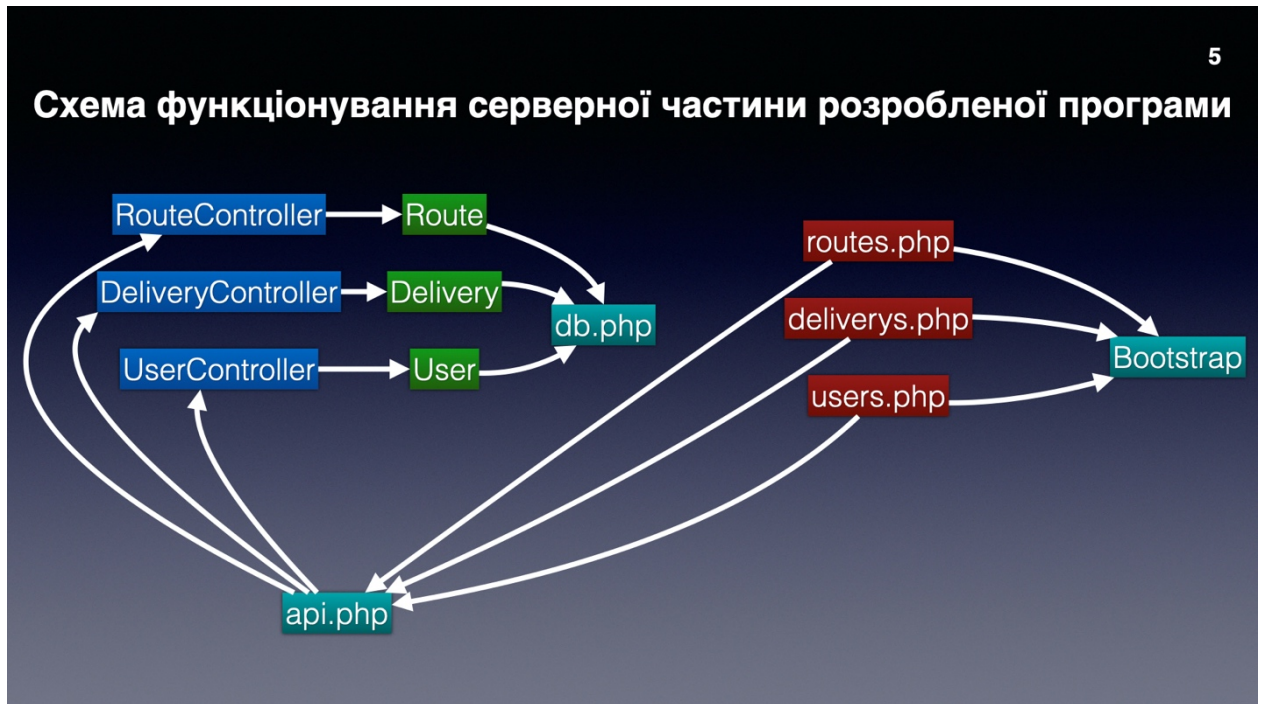


Рисунок Б.5 - Слайд 5

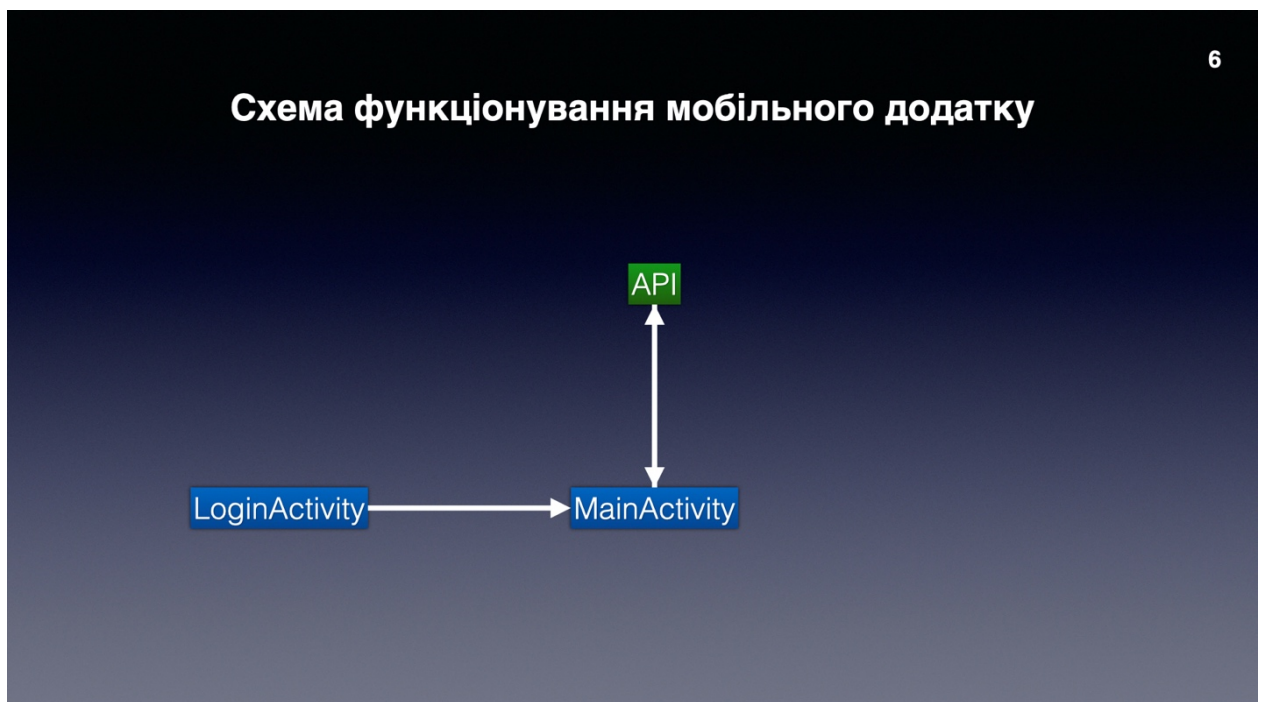


Рисунок Б.6 - Слайд 6

Концептуальна модель бази даних

7



Рисунок Б.7 - Слайд 7

Вибір мови програмування

8

	PHP	PYTHON
Переваги над іншими мовами	Легкість встановлення Швидкість роботи Просте підключення до баз даних	Підтримка більшої кількості бібліотек Захищеність
	Java	Kotlin
Переваги над іншими мовами	Поширеність серед розробників Підтримка більшої кількості бібліотек	Простота коду

Рисунок Б.8 - Слайд 8

Інтерфейс серверної частини програми

9

The left screenshot shows a web browser window with the URL `https://rustan.inf.ua/delivery/admin/couriers.php`. It displays a table with the following data:

ID	Name	PIN
1	Test1	8627
2	Test2	1234
3	Test3	1234
4	Test4	1234
5	Testweb	9622
6	Testweb	8063
7	Testweb	1984
8	Test11111	7966
9	Test11111	8583
10	Test11111	6072

The right screenshot shows a web browser window with the URL `https://rustan.inf.ua/delivery/admin/newDelivery.php`. It displays a form for adding a new delivery with the following fields:

- Address:** Shklna 4
- Phone:** +3800000000000000
- Date:** ДД.ММ.ГГГГ, --:--

There is a green button labeled "Отправить" (Send) at the bottom.

Рисунок Б.9 - Слайд 9

Інтерфейс мобільного додатку

10

The left screenshot shows a mobile app interface with a purple header labeled "Delivery". It has two input fields for "ID" and "PIN". At the bottom, there is a purple button labeled "LOGIN".

The right screenshot shows a mobile app interface with a purple header labeled "Delivery". It displays the address "Address: 34, Testova str.". At the bottom, there are two purple buttons labeled "CALL" and "NAVIGATE", and a purple button labeled "DELIVERED".

Рисунок Б.10 - Слайд 10

Висновки

11

В результаті роботи було розроблено систему для автоматичного пошуку оптимального маршруту доставки та контроль фактично виконаної доставки. Систему було протестовано в умовах, близьких до реальних. Отримано такі результати, як зменшення середньої дистанції маршруту, зменшення затрат пального кур'єром. При реалізації проекту було досягнуто мети, яка полягала в створенні програмної системи для розрахунку оптимального маршруту доставки замовлень кур'єрською службою та мобільного додатку для контролю коректності часу та місцезнаходження доставки замовлення.

Система може бути використана на будь-яких підприємствах, що займаються наданням послуг з кур'єрської доставки.

Систему було розроблено на мобільну платформу Android з використанням API.

Також було розроблено панель керування для створення даних для передачі в додаток.

Систему було протестовано в умовах, наближених до реальних, та було підтверджено доцільність використання системи у реальних умовах.

Рисунок Б.11 - Слайд 11

12

Дякую за увагу!

Рисунок Б.12 - Слайд 12