A logo on a grey background

Description automatically generated

NextPointLogix

Вимоги до системи.

1. **Вступ**

**Мета проекту**

Метою проекту NextPointLogix є автоматизація роботи транспортно-логістичного відділу, що займається доставкою працівників на завод до початку робочої зміни і відвозить їх додому після закінчення робочої зміни. Це рішення спрямоване на підвищення ефективності планування маршрутів, зниження витрат на паливо та забезпечення зручності для всіх учасників процесу.

**Огляд системи**

NextPointLogix - це інтегроване рішення, яке включає мобільні додатки для працівників і водіїв, а також веб-додатки для логіст-оператора, фінансового менеджера та адміністратора системи. Основні функції системи включають:

* Вибір днів транспортування працівниками через мобільний додаток.
* Автоматичне складання маршрутів на основі запитів працівників.
* Перегляд маршрутів та інформації про пасажирів водіями через мобільний додаток.
* Управління маршрутами, моніторинг витрат на паливо та складання звітів через веб-додатки.

**Цільова аудиторія**

Цільовою аудиторією NextPointLogix є:

* **Працівники заводу**: Користуються мобільним додатком для вибору днів транспортування та перегляду маршрутів.
* **Водії автобусів**: Користуються мобільним додатком для перегляду маршрутів та списків пасажирів.
* **Логіст-оператори**: Використовують веб-додаток для управління маршрутами та моніторингу транспортних операцій.
* **Фінансові менеджери**: Використовують веб-додаток для моніторингу витрат, складання звітів та управління фінансовими аспектами перевезень.
* **Адміністратори системи**: Використовують веб-додаток для управління користувачами та налаштування прав доступу.

NextPointLogix покликаний забезпечити зручність, надійність та ефективність у процесі організації транспортування працівників, забезпечуючи тим самим оптимізацію ресурсів та підвищення загальної продуктивності транспортно-логістичного відділу.

**2. Загальний опис**

**2.1 Перелік компонентів системи**

Система NextPointLogix складається з таких основних компонентів:

1. **Мобільний додаток для працівників**: Призначений для працівників, які можуть за допомогою додатка вибирати дні, в які їм потрібен транспорт, переглядати маршрут та отримувати повідомлення.
2. **Мобільний додаток для водіїв**: Призначений для водіїв, які отримують інформацію про маршрут, список пасажирів, контактні дані та орієнтовний час посадки.
3. **Веб-додаток для логіст-оператора**: Використовується для контролю надходження даних від працівників, складання та коригування маршрутів, розподілу маршрутів між водіями та обліку палива.
4. **Веб-додаток для фінансового менеджера**: Призначений для моніторингу використання транспорту, розрахунку витрат на паливо та виплат водіям, а також для створення звітів.
5. **Веб-додаток для адміністратора системи**: Використовується для створення користувачів, налаштування прав доступу та загального адміністрування системи.
6. **Бекенд-сервер**: Реалізований на Python, відповідає за обробку запитів від фронтенду, управління даними та бізнес-логікою.
7. **База даних**: MySQL використовується для зберігання всіх даних системи, включаючи інформацію про користувачів, маршрути, транспортні засоби та витрати палива.
8. **Інфраструктура на AWS**: Включає сервери, бази даних та інші сервіси, необхідні для хостингу та безперебійної роботи системи.
9. **CI/CD система**: Інтегрована з GitHub для автоматизації процесів розгортання та тестування.

**2.2 Основні функції**

* **Мобільний додаток для працівників**:
  + Реєстрація та авторизація користувачів.
  + Вибір днів для транспортування за допомогою інтерактивного календаря.
  + Перегляд інформації про маршрут та час прибуття автобуса.
  + Отримання push-повідомлень про оновлення маршрутів.
* **Мобільний додаток для водіїв**:
  + Отримання та перегляд запланованих маршрутів.
  + Перегляд списків пасажирів з контактними даними.
  + Інформація про орієнтовний час посадки та прибуття на підприємство.
* **Веб-додаток для логіст-оператора**:
  + Моніторинг надходження даних від працівників.
  + Автоматичне та ручне складання маршрутів.
  + Розподіл маршрутів між водіями.
  + Ведення обліку палива на основі даних про пробіг автобусів.
* **Веб-додаток для фінансового менеджера**:
  + Моніторинг використання транспорту за день.
  + Облік пробігу автобусів та витрат палива.
  + Розрахунок виплат водіям на основі пробігу.
  + Створення та перегляд фінансових звітів за допомогою PowerBI.
* **Веб-додаток для адміністратора системи**:
  + Створення та управління обліковими записами користувачів.
  + Налаштування прав доступу для різних ролей.
  + Адміністрування загальних налаштувань системи.

**2.3 Інтеграційні вимоги**

* **REST API**: Забезпечує взаємодію між фронтенд та бекенд компонентами.
* **Інтеграція з існуючими системами**: Забезпечує можливість інтеграції з іншими внутрішніми системами компанії або зовнішніми сервісами, якщо це необхідно.

**2.4 Основні вимоги до інфраструктури**

* **Хостинг на AWS**: Використання сервісів AWS для забезпечення масштабованості, надійності та безпеки системи.
* **CI/CD процеси**: Автоматизація розгортання та тестування за допомогою GitHub Actions або іншого CI/CD інструменту.

**3. Функціональні вимоги**

**3.1 Мобільний додаток для працівників**

1. **Реєстрація та авторизація**:
   * Працівники повинні мати можливість зареєструватися у системі за допомогою електронної пошти або номеру телефону.
   * Авторизація повинна підтримувати логін через електронну пошту або номер телефону та пароль.
2. **Вибір днів для транспортування**:
   * Інтерактивний календар, де працівники можуть вибирати дні, коли їм потрібен транспорт.
   * Можливість змінювати вибір днів до певного часу до початку зміни.
3. **Перегляд інформації про маршрут**:
   * Інформація про включення у маршрут.
   * Орієнтовний час прибуття автобуса до місця посадки та на підприємство.
   * Повідомлення про зміни у маршруті.
4. **Push-повідомлення**:
   * Сповіщення про оновлення маршрутів.
   * Нагадування про наближення часу посадки.
   * Сповіщення про затримки або інші зміни.

**3.2 Мобільний додаток для водіїв**

1. **Реєстрація та авторизація**:
   * Водії повинні мати можливість зареєструватися у системі за допомогою електронної пошти або номеру телефону.
   * Авторизація повинна підтримувати логін через електронну пошту або номер телефону та пароль.
2. **Прийом та перегляд маршрутів**:
   * Перегляд запланованих маршрутів для кожного дня.
   * Детальна інформація про точки посадки та висадки працівників.
3. **Перегляд списків пасажирів**:
   * Список пасажирів з контактними даними (ім’я, телефон).
   * Орієнтовний час посадки та прибуття на підприємство.
4. **Push-повідомлення**:
   * Сповіщення про зміни у маршрутах.
   * Нагадування про наближення часу посадки.

**3.3 Веб-додаток для логіст-оператора**

1. **Моніторинг надходження даних від працівників**:
   * Перегляд вибору днів транспортування працівниками у реальному часі.
   * Сповіщення про нові або змінені запити на транспортування.
2. **Складання та коригування маршрутів**:
   * Автоматичне складання маршрутів на основі даних від працівників.
   * Можливість ручного коригування маршрутів.
   * Відображення довжини маршруту та орієнтовного часу проходження.
3. **Розподіл маршрутів між водіями**:
   * Призначення водіїв на маршрути.
   * Можливість перегляду і коригування розподілу у реальному часі.
4. **Облік палива**:
   * Введення даних про розхід палива на кілометр пробігу.
   * Автоматичний розрахунок загального витрату палива для кожного маршруту.

**3.4 Веб-додаток для фінансового менеджера**

1. **Моніторинг використання транспорту**:
   * Перегляд даних про використання транспорту за день.
   * Відображення пробігу автобусів та витрат палива.
2. **Розрахунок витрат**:
   * Розрахунок виплат водіям на основі пробігу.
   * Розрахунок витрат на паливо.
3. **Створення та перегляд фінансових звітів**:
   * Можливість створення звітів за допомогою PowerBI.
   * Перегляд готових звітів про витрати та пробіг.

**3.5 Веб-додаток для адміністратора системи**

1. **Управління користувачами**:
   * Створення нових облікових записів користувачів.
   * Перегляд та редагування інформації про користувачів.
   * Видалення користувачів.
2. **Налаштування прав доступу**:
   * Призначення ролей та прав доступу для різних категорій користувачів.
   * Управління правами доступу до різних функцій системи.
3. **Адміністрування загальних налаштувань системи**:
   * Налаштування системних параметрів.
   * Моніторинг стану системи та вирішення технічних проблем.

**4. Нефункціональні вимоги**

**4.1 Продуктивність**

1. **Час відгуку**:
   * Система повинна забезпечувати час відгуку менше 2 секунд для більшості операцій користувачів.
   * Час завантаження основних сторінок веб-додатків не повинен перевищувати 3 секунд.
2. **Обробка запитів**:
   * Система повинна підтримувати обробку до 1000 одночасних запитів без значного зниження продуктивності.

**4.2 Масштабованість**

1. **Горизонтальне масштабування**:
   * Система повинна підтримувати горизонтальне масштабування для збільшення продуктивності та обробки більшої кількості запитів.
2. **Обробка даних**:
   * Система повинна бути здатна обробляти зростаючі обсяги даних без зниження продуктивності.

**4.3 Безпека**

1. **Аутентифікація та авторизація**:
   * Використання безпечних методів аутентифікації та авторизації для захисту облікових записів користувачів.
   * Підтримка багаторівневої аутентифікації (наприклад, двофакторна аутентифікація).
2. **Захист даних**:
   * Всі передані дані повинні бути зашифровані за допомогою HTTPS.
   * Зберігання чутливих даних (наприклад, паролів) повинно виконуватись у зашифрованому вигляді.
3. **Управління доступом**:
   * Контроль доступу на основі ролей (RBAC) для забезпечення відповідного рівня доступу для різних користувачів.

**4.4 Надійність**

1. **Безперервність роботи**:
   * Система повинна мати час безперервної роботи (uptime) не менше 99,9%.
2. **Відновлення після збоїв**:
   * Система повинна мати можливість автоматичного відновлення після збоїв.
   * Регулярне резервне копіювання даних повинно забезпечувати можливість відновлення даних у разі втрати.

**4.5 Юзабіліті**

1. **Інтерфейс користувача**:
   * Інтерфейси повинні бути інтуїтивно зрозумілими та зручними у використанні.
   * Підтримка локалізації інтерфейсів на кілька мов.
2. **Доступність**:
   * Система повинна відповідати стандартам доступності (наприклад, WCAG) для забезпечення використання користувачами з обмеженими можливостями.

**4.6 Сумісність**

1. **Браузери**:
   * Веб-додатки повинні підтримувати основні сучасні браузери (Google Chrome, Mozilla Firefox, Microsoft Edge, Safari).
2. **Операційні системи**:
   * Мобільні додатки повинні бути сумісні з останніми версіями Android та iOS.

**4.7 Обслуговуваність**

1. **Документація**:
   * Система повинна мати повну документацію для розробників, адміністраторів та користувачів.
2. **Логування та моніторинг**:
   * Система повинна забезпечувати логування ключових подій та можливість моніторингу стану системи у реальному часі.

**4.8 Законодавчі вимоги**

1. **Захист персональних даних**:
   * Система повинна відповідати вимогам законодавства щодо захисту персональних даних (наприклад, GDPR).

**5. Архітектура системи**

**5.1 Огляд архітектури**

Архітектура системи NextPointLogix побудована за принципом розподіленої системи, що забезпечує високу масштабованість, надійність та ефективність. Система складається з кількох основних компонентів: мобільні додатки для працівників і водіїв, веб-додатки для логіст-оператора, фінансового менеджера та адміністратора, бекенд-сервер та база даних. Кожен з цих компонентів виконує специфічні функції та взаємодіє з іншими компонентами через API.

**5.2 Компоненти системи**

1. **Мобільний додаток для працівників**
   * **Платформа**: React Native
   * **Функціональність**: вибір днів для транспортування, перегляд маршрутів, отримання сповіщень.
2. **Мобільний додаток для водіїв**
   * **Платформа**: React Native
   * **Функціональність**: перегляд маршрутів, списків пасажирів, отримання сповіщень.
3. **Веб-додаток для логіст-оператора**
   * **Платформа**: React
   * **Функціональність**: моніторинг даних від працівників, складання та коригування маршрутів, облік палива.
4. **Веб-додаток для фінансового менеджера**
   * **Платформа**: React
   * **Функціональність**: моніторинг використання транспорту, розрахунок витрат, створення звітів.
5. **Веб-додаток для адміністратора системи**
   * **Платформа**: React
   * **Функціональність**: управління користувачами, налаштування прав доступу, адміністрування системи.
6. **Бекенд-сервер**
   * **Платформа**: Python
   * **Функціональність**: обробка запитів від фронтенду, управління даними та бізнес-логікою.
   * **Фреймворк**: Flask або Django
7. **База даних**
   * **Платформа**: MySQL
   * **Функціональність**: зберігання даних про користувачів, маршрути, транспортні засоби, витрати палива.
8. **Інфраструктура на AWS**
   * **Компоненти**: EC2 для серверів, RDS для бази даних, S3 для зберігання файлів, IAM для управління доступом.
   * **Функціональність**: забезпечення масштабованості, надійності та безпеки.
9. **CI/CD система**
   * **Платформа**: GitHub Actions
   * **Функціональність**: автоматизація розгортання та тестування.

**5.3 Взаємодія компонентів**

1. **Мобільні додатки**:
   * Працівники та водії взаємодіють з бекенд-сервером через REST API для отримання та відправлення даних.
   * Отримання даних про маршрути та сповіщення в реальному часі здійснюється через вебсокети або push-повідомлення.
2. **Веб-додатки**:
   * Логіст-оператор, фінансовий менеджер та адміністратор взаємодіють з бекенд-сервером через REST API для управління даними та отримання звітів.
   * Веб-додатки побудовані на PowerApp, що забезпечує швидку розробку та інтеграцію з іншими службами Microsoft.
3. **Бекенд-сервер**:
   * Обробка запитів від фронтенд компонентів, виконання бізнес-логіки та взаємодія з базою даних MySQL.
   * Використання бібліотек для обробки запитів, автентифікації та авторизації користувачів, управління даними.
4. **База даних**:
   * Зберігання структурованих даних, забезпечення надійного доступу та виконання запитів.
   * Регулярне резервне копіювання даних для забезпечення відновлення у разі втрати.

**5.4 Інтеграція з зовнішніми сервісами**

1. **AWS сервіси**:
   * Використання EC2 для хостингу бекенд-серверів.
   * Використання RDS для управління базою даних MySQL.
   * Використання S3 для зберігання файлів та медіа-контенту.
   * Використання IAM для управління доступом та безпеки.
2. **CI/CD інструменти**:
   * Інтеграція з GitHub для управління версіями коду та автоматизації процесів розгортання та тестування.
   * Використання GitHub Actions для налаштування безперервної інтеграції та розгортання.

**5.5 Схема архітектури**

*(Схема додана окремим файлом)*

*Код малюнка схеми*

[Graphviz Online (dreampuf.github.io)](https://dreampuf.github.io/GraphvizOnline/#digraph%20G%20%7B%0A%20%20rankdir%3DLR%3B%0A%20%20node%20%5Bshape%3Dbox%5D%3B%0A%20%20%0A%20%20subgraph%20cluster_frontend%20%7B%0A%20%20%20%20label%20%3D%20%22Front-end%22%3B%0A%20%20%20%20mobile_app_employee%20%5Blabel%3D%22%D0%9C%D0%BE%D0%B1%D1%96%D0%BB%D1%8C%D0%BD%D0%B8%D0%B9%20%D0%B4%D0%BE%D0%B4%D0%B0%D1%82%D0%BE%D0%BA%5Cn%D0%B4%D0%BB%D1%8F%20%D0%BF%D1%80%D0%B0%D1%86%D1%96%D0%B2%D0%BD%D0%B8%D0%BA%D1%96%D0%B2%22%5D%3B%0A%20%20%20%20mobile_app_driver%20%5Blabel%3D%22%D0%9C%D0%BE%D0%B1%D1%96%D0%BB%D1%8C%D0%BD%D0%B8%D0%B9%20%D0%B4%D0%BE%D0%B4%D0%B0%D1%82%D0%BE%D0%BA%5Cn%D0%B4%D0%BB%D1%8F%20%D0%B2%D0%BE%D0%B4%D1%96%D1%97%D0%B2%22%5D%3B%0A%20%20%20%20web_app_logist%20%5Blabel%3D%22%D0%92%D0%B5%D0%B1-%D0%B4%D0%BE%D0%B4%D0%B0%D1%82%D0%BE%D0%BA%5Cn%D0%B4%D0%BB%D1%8F%20%D0%BB%D0%BE%D0%B3%D1%96%D1%81%D1%82-%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%B0%22%5D%3B%0A%20%20%20%20web_app_finance%20%5Blabel%3D%22%D0%92%D0%B5%D0%B1-%D0%B4%D0%BE%D0%B4%D0%B0%D1%82%D0%BE%D0%BA%5Cn%D0%B4%D0%BB%D1%8F%20%D1%84%D1%96%D0%BD%D0%B0%D0%BD%D1%81%D0%BE%D0%B2%D0%BE%D0%B3%D0%BE%20%D0%BC%D0%B5%D0%BD%D0%B5%D0%B4%D0%B6%D0%B5%D1%80%D0%B0%22%5D%3B%0A%20%20%20%20web_app_admin%20%5Blabel%3D%22%D0%92%D0%B5%D0%B1-%D0%B4%D0%BE%D0%B4%D0%B0%D1%82%D0%BE%D0%BA%5Cn%D0%B4%D0%BB%D1%8F%20%D0%B0%D0%B4%D0%BC%D1%96%D0%BD%D1%96%D1%81%D1%82%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%B0%22%5D%3B%0A%20%20%7D%0A%20%20%0A%20%20subgraph%20cluster_backend%20%7B%0A%20%20%20%20label%20%3D%20%22Back-end%22%3B%0A%20%20%20%20backend_server%20%5Blabel%3D%22%D0%91%D0%B5%D0%BA%D0%B5%D0%BD%D0%B4-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%22%5D%3B%0A%20%20%7D%0A%20%20%0A%20%20subgraph%20cluster_database%20%7B%0A%20%20%20%20label%20%3D%20%22%D0%91%D0%B0%D0%B7%D0%B0%20%D0%B4%D0%B0%D0%BD%D0%B8%D1%85%22%3B%0A%20%20%20%20database%20%5Blabel%3D%22MySQL%22%5D%3B%0A%20%20%7D%0A%20%20%0A%20%20subgraph%20cluster_aws%20%7B%0A%20%20%20%20label%20%3D%20%22AWS%22%3B%0A%20%20%20%20ec2%20%5Blabel%3D%22EC2%22%5D%3B%0A%20%20%20%20rds%20%5Blabel%3D%22RDS%22%5D%3B%0A%20%20%20%20s3%20%5Blabel%3D%22S3%22%5D%)

digraph G {

rankdir=LR;

node [shape=box];

subgraph cluster\_frontend {

label = "Front-end";

mobile\_app\_employee [label="Мобільний додаток\nдля працівників"];

mobile\_app\_driver [label="Мобільний додаток\nдля водіїв"];

web\_app\_logist [label="Веб-додаток\nдля логіст-оператора"];

web\_app\_finance [label="Веб-додаток\nдля фінансового менеджера"];

web\_app\_admin [label="Веб-додаток\nдля адміністратора"];

}

subgraph cluster\_backend {

label = "Back-end";

backend\_server [label="Бекенд-сервер"];

}

subgraph cluster\_database {

label = "База даних";

database [label="MySQL"];

}

subgraph cluster\_aws {

label = "AWS";

ec2 [label="EC2"];

rds [label="RDS"];

s3 [label="S3"];

iam [label="IAM"];

}

subgraph cluster\_ci\_cd {

label = "CI/CD";

github\_actions [label="GitHub Actions"];

}

Більш розгорнутий варіант

digraph G {

rankdir=LR;

node [shape=box, style=filled, fillcolor=lightblue];

edge [color=black];

# Підсистема фронтенду

subgraph cluster\_frontend {

label = "Фронтенд";

color=blue;

mobile\_app\_employee [label="Мобільний додаток\nдля працівників\n(React Native)", shape=cylinder];

mobile\_app\_driver [label="Мобільний додаток\nдля водіїв\n(React Native)", shape=cylinder];

web\_app\_logist [label="Веб-додаток\nдля логіст-оператора\n(React Native)"];

web\_app\_finance [label="Веб-додаток\nдля фінансового менеджера\n(React Native)"];

web\_app\_admin [label="Веб-додаток\nдля адміністратора\n(React Native)"];

}

# Підсистема бекенду

subgraph cluster\_backend {

label = "Бекенд";

color=green;

backend\_server [label="Бекенд-сервер\n(Python, Flask)\nAPI: REST, WebSocket"];

auth\_service [label="Сервіс аутентифікації\n(OAuth2)"];

notification\_service [label="Сервіс повідомлень\n(Push-повідомлення)"];

}

# Підсистема бази даних

subgraph cluster\_database {

label = "База даних";

color=orange;

database [label="MySQL\nТаблиці: користувачі, маршрути, транспорт, витрати"];

}

# Підсистема AWS

subgraph cluster\_aws {

label = "AWS";

color=gray;

ec2 [label="EC2\nІнстанції серверів"];

rds [label="RDS\nУправління БД"];

s3 [label="S3\nЗберігання файлів"];

iam [label="IAM\nУправління доступом"];

}

# Підсистема CI/CD

subgraph cluster\_cicd {

label = "CI/CD";

color=purple;

github\_actions [label="GitHub Actions\nPipeline: build, test, deploy"];

}

# Зв'язки

mobile\_app\_employee -> backend\_server [label="REST API"];

mobile\_app\_driver -> backend\_server [label="REST API"];

web\_app\_logist -> backend\_server [label="REST API"];

web\_app\_finance -> backend\_server [label="REST API"];

web\_app\_admin -> backend\_server [label="REST API"];

backend\_server -> database [label="SQL"];

backend\_server -> auth\_service [label="OAuth2"];

backend\_server -> notification\_service [label="WebSocket"];

backend\_server -> ec2 [label="Deployment"];

github\_actions -> backend\_server [label="Deployment"];

}

**6. Інтеграційні вимоги**

**6.1 Загальний огляд**

Інтеграційні вимоги визначають, як система NextPointLogix взаємодіє з іншими системами та сервісами для забезпечення ефективної роботи та обміну даними. Інтеграція охоплює як внутрішні компоненти системи (мобільні та веб-додатки, бекенд-сервер), так і зовнішні сервіси та інфраструктуру.

**6.2 Інтеграція компонентів системи**

1. **Мобільні додатки з бекенд-сервером**
   * **Технології**: REST API, WebSocket
   * **Функціональність**:
     + Мобільний додаток для працівників та водіїв використовує REST API для надсилання запитів і отримання даних з бекенд-сервера.
     + Використання WebSocket для отримання реального часу сповіщень і оновлень.
2. **Веб-додатки з бекенд-сервером**
   * **Технології**: REST API
   * **Функціональність**:
     + Веб-додатки для логіст-оператора, фінансового менеджера та адміністратора взаємодіють з бекенд-сервером через REST API для управління даними та отримання звітів.
3. **Бекенд-сервер з базою даних**
   * **Технології**: SQL
   * **Функціональність**:
     + Бекенд-сервер виконує запити до бази даних MySQL для зберігання, оновлення та отримання інформації про користувачів, маршрути, транспортні засоби та витрати пального.
4. **Бекенд-сервер з інфраструктурою AWS**
   * **Технології**: AWS SDK, API
   * **Функціональність**:
     + Бекенд-сервер використовує AWS SDK для взаємодії з AWS ресурсами, такими як EC2 (для хостингу серверів), RDS (для бази даних), S3 (для зберігання файлів) та IAM (для управління доступом).
5. **CI/CD система з кодовою базою**
   * **Технології**: GitHub Actions
   * **Функціональність**:
     + GitHub Actions автоматизує процеси розгортання, тестування та перевірки коду. Інтеграція забезпечує автоматичне оновлення коду на серверах при комітах у репозиторій.

**6.3 Інтеграція з зовнішніми сервісами**

1. **Системи сповіщень**
   * **Технології**: Push-нотифікації, SMS API
   * **Функціональність**:
     + Використання сторонніх сервісів для надсилання push-нотифікацій до мобільних додатків для працівників і водіїв.
     + Інтеграція з SMS API для надсилання текстових повідомлень у випадку, якщо push-нотифікації не доставлені.
2. **Аналітика та звітність**
   * **Технології**: PowerBI, Google Analytics
   * **Функціональність**:
     + Інтеграція з PowerBI для створення та перегляду аналітичних звітів фінансовим менеджером.
     + Використання Google Analytics для відстеження користувацької активності у веб-додатках.
3. **Інтеграція з бухгалтерськими програмами**
   * **Технології**: API бухгалтерських програм (наприклад, 1С)
   * **Функціональність**:
     + **Формування інвойсів**: Автоматичне створення та передача інвойсів на оплату послуг водіїв (приватних підприємців) та оплату пального до бухгалтерських систем.
     + **Обробка фінансових даних**: Інтеграція з бухгалтерськими програмами для обліку витрат і доходів, ведення фінансових записів і звітності.

**6.4 Безпека інтеграцій**

1. **Аутентифікація та авторизація**
   * **Технології**: OAuth 2.0, JWT
   * **Функціональність**:
     + Використання OAuth 2.0 для управління доступом до API.
     + Використання JSON Web Tokens (JWT) для автентифікації запитів.
2. **Шифрування даних**
   * **Технології**: TLS/SSL
   * **Функціональність**:
     + Забезпечення захисту даних при передачі між компонентами системи через використання TLS/SSL шифрування.
3. **Моніторинг та аудит**
   * **Технології**: AWS CloudWatch, ELK Stack
   * **Функціональність**:
     + Використання AWS CloudWatch для моніторингу продуктивності та доступності серверів.
     + Використання ELK Stack (Elasticsearch, Logstash, Kibana) для збору та аналізу логів.

**7. Вимоги до даних**

**7.1 Загальний огляд**

Вимоги до даних описують, які дані система NextPointLogix повинна збирати, зберігати, обробляти та управляти ними. Цей розділ визначає типи даних, їх структуру, обсяг, вимоги до зберігання та забезпечення конфіденційності та цілісності даних.

**7.2 Типи даних**

1. **Дані працівників**
   * **Опис**: Інформація про працівників, яка включає особисті дані (ім’я, прізвище, адреса, контактні дані), робочий графік та вимоги до транспортування.
   * **Формат**: Текстові дані, дати, часові мітки.
   * **Зберігання**: База даних MySQL.
2. **Дані про маршрути**
   * **Опис**: Інформація про маршрути, яка включає точки посадки та висадки, орієнтовний час прибуття та відправлення.
   * **Формат**: Географічні координати, текстові дані, часові мітки.
   * **Зберігання**: База даних MySQL, можливе використання зовнішніх картографічних сервісів.
3. **Дані про транспортні засоби**
   * **Опис**: Інформація про автобуси, яка включає їх реєстраційні номери, технічні характеристики, витрати пального та пробіг.
   * **Формат**: Текстові дані, числові значення.
   * **Зберігання**: База даних MySQL.
4. **Фінансові дані**
   * **Опис**: Дані про витрати на паливо, виплати водіям, інвойси та рахунки.
   * **Формат**: Числові значення, текстові дані, документи (PDF, Excel).
   * **Зберігання**: База даних MySQL, документи можуть зберігатися на AWS S3.
5. **Дані аналітики та звітності**
   * **Опис**: Дані для створення звітів і аналітичних панелей, що включають статистику використання транспорту, витрати, ефективність маршрутів.
   * **Формат**: Числові значення, текстові дані, графіки.
   * **Зберігання**: База даних MySQL, звіти можуть бути збережені у PowerBI.

**7.3 Структура даних**

1. **База даних MySQL**
   * **Таблиці**:
     + Employees: Зберігає інформацію про працівників.
     + Routes: Зберігає інформацію про маршрути.
     + Vehicles: Зберігає інформацію про транспортні засоби.
     + FuelExpenses: Зберігає дані про витрати пального.
     + DriverPayments: Зберігає дані про виплати водіям.
     + Analytics: Зберігає дані для аналітики та звітності.
2. **Документи**
   * **Формати**: PDF, Excel
   * **Зберігання**: AWS S3

**7.4 Вимоги до зберігання даних**

1. **Резервне копіювання**
   * **Опис**: Дані повинні резервно копіюватися щодня з можливістю відновлення до 30 днів.
   * **Формат**: Автоматичні резервні копії бази даних, зберігання на AWS S3.
2. **Архівування**
   * **Опис**: Історичні дані повинні бути архівовані та зберігатися протягом не менше ніж 5 років.
   * **Формат**: Архіви бази даних, документи.

**7.5 Забезпечення конфіденційності та цілісності даних**

1. **Конфіденційність**
   * **Опис**: Забезпечення захисту особистих даних працівників та фінансових даних за допомогою шифрування та контролю доступу.
   * **Технології**: TLS/SSL для передачі даних, шифрування бази даних.
2. **Цілісність**
   * **Опис**: Забезпечення цілісності даних шляхом перевірки на коректність, запобігання несанкціонованим змінам та помилкам.
   * **Технології**: Механізми перевірки цілісності даних у базі даних, контрольні суми.
3. **Доступ**
   * **Опис**: Управління доступом до даних на основі ролей користувачів (логіст-оператор, фінансовий менеджер, адміністратор).
   * **Технології**: OAuth 2.0, JWT для аутентифікації та авторизації.

**8. Вимоги до розгортання та інфраструктури**

**8.1 Загальний огляд**

Вимоги до розгортання та інфраструктури описують умови, необхідні для успішного впровадження та підтримки системи NextPointLogix. Цей розділ визначає вимоги до середовища хостингу, інфраструктури, моніторингу, резервного копіювання та відновлення, а також управління середовищами розробки та продукції.

**8.2 Інфраструктура**

1. **Хостинг**
   * **Опис**: Система NextPointLogix буде хоститися на хмарній платформі AWS.
   * **Сервіси AWS**:
     + **EC2**: Для хостингу бекенд-серверів і веб-додатків.
     + **RDS**: Для управління базою даних MySQL.
     + **S3**: Для зберігання файлів, резервних копій та документів.
     + **CloudWatch**: Для моніторингу та логування.
     + **IAM**: Для управління доступом до ресурсів AWS.
2. **Мережа**
   * **Опис**: Застосування Virtual Private Cloud (VPC) для забезпечення ізоляції та безпеки внутрішньої мережі.
   * **Компоненти**:
     + **Subnets**: Розподіл сервісів по різних підмережах для підвищення безпеки.
     + **Security Groups**: Налаштування правил доступу до ресурсів.
3. **Зберігання даних**
   * **Опис**: Використання реляційної бази даних MySQL на AWS RDS.
   * **Резервне копіювання**: Автоматичне резервне копіювання бази даних щодня.
4. **Моніторинг і управління**
   * **Опис**: Використання AWS CloudWatch для моніторингу продуктивності та здоров'я системи.
   * **Компоненти**:
     + **Dashboards**: Візуалізація ключових метрик.
     + **Alarms**: Налаштування тривог для критичних подій.

**8.3 Розгортання**

1. **Середовище розробки**
   * **Опис**: Оточення для розробки та тестування повинно бути ізольоване від продукційного середовища.
   * **Компоненти**:
     + **Dev/Testing EC2 Instances**: Сервери для розробки та тестування.
     + **GitHub Actions**: Для автоматизації процесів CI/CD.
2. **Середовище продукції**
   * **Опис**: Основне середовище, де буде працювати кінцева версія системи.
   * **Компоненти**:
     + **Production EC2 Instances**: Сервери для продуктивної роботи.
     + **Load Balancer**: Для розподілу навантаження між серверами.
3. **CI/CD**
   * **Опис**: Впровадження процесу безперервної інтеграції та доставки через GitHub Actions.
   * **Компоненти**:
     + **Pipeline**: Автоматичні побудови, тести та деплоя коду.
     + **Deployment**: Автоматичне розгортання на AWS EC2.

**8.4 Резервне копіювання та відновлення**

1. **Резервне копіювання**
   * **Опис**: Резервне копіювання даних бази даних, документів та конфігураційних файлів.
   * **Частота**: Щоденне резервне копіювання бази даних, щотижневе резервне копіювання файлів.
   * **Зберігання**: Резервні копії зберігаються в AWS S3.
2. **Відновлення**
   * **Опис**: Процедури для відновлення даних у випадку збою або втрати даних.
   * **Час відновлення**: Максимально допустимий час відновлення даних - 4 години.

**8.5 Безпека**

1. **Аутентифікація та авторизація**
   * **Опис**: Використання AWS IAM для управління доступом до ресурсів.
   * **Механізми**: Мультфакторна аутентифікація, управління ролями.
2. **Шифрування**
   * **Опис**: Шифрування даних як в спокої (at rest), так і під час передачі (in transit).
   * **Технології**: AES-256 для зберігання даних, TLS/SSL для передачі даних.
3. **Оновлення та патчинг**
   * **Опис**: Регулярне оновлення серверів та застосунків для виправлення вразливостей.
   * **Процес**: Автоматичні оновлення системи безпеки, ручне оновлення компонентів додатка.

**9. Тестові вимоги**

**9.1 Тестування**

Тестування є критичним етапом розробки, який забезпечує перевірку всіх аспектів системи NextPointLogix, щоб впевнитися в її якості та відповідності вимогам. Основні типи тестування включають функціональне, інтеграційне та навантажувальне тестування.

1. **Функціональне тестування**
   * **Опис**: Перевірка функціональності системи відповідно до вимог документації. Мета - впевнитися, що кожна функція працює згідно з визначеними вимогами.
   * **Основні сценарії**:
     + **Мобільний додаток працівника**:
       - Реєстрація та авторизація користувача.
       - Вибір днів для транспортування.
       - Перегляд маршруту і орієнтовного часу прибуття.
     + **Мобільний додаток водія**:
       - Отримання та перегляд плану маршруту.
       - Перегляд інформації про точки посадки та висадки.
       - Оновлення статусу виконання маршруту.
     + **Веб-додаток логіст-оператора**:
       - Формування і редагування маршрутів.
       - Перегляд статистики і витрат.
       - Розподіл маршрутів між водіями.
     + **Веб-інтерфейс фінансового менеджера**:
       - Перегляд витрат на паливо.
       - Нарахування виплат водіям.
       - Формування звітів.
     + **Веб-інтерфейс адміністратора системи**:
       - Управління користувачами та ролями.
       - Налаштування доступу.
2. **Інтеграційне тестування**
   * **Опис**: Перевірка взаємодії між різними компонентами системи, щоб впевнитися, що вони працюють разом правильно.
   * **Основні сценарії**:
     + **Інтеграція мобільних додатків з бекендом**:
       - Перевірка обміну даними між мобільними додатками та сервером.
     + **Інтеграція веб-додатків з бекендом**:
       - Перевірка коректності передачі даних від веб-додатків до серверів.
     + **Інтеграція з базою даних MySQL**:
       - Перевірка коректності зберігання та отримання даних.
     + **Інтеграція з зовнішніми системами**:
       - Перевірка інтеграції з бухгалтерськими програмами (наприклад, 1С) для формування інвойсів.
3. **Навантажувальне тестування**
   * **Опис**: Перевірка системи на здатність обробляти великі обсяги навантаження та її продуктивність під високим навантаженням.
   * **Основні сценарії**:
     + **Тестування навантаження на API**:
       - Перевірка швидкості та стабільності API під високим навантаженням.
     + **Тестування продуктивності бази даних**:
       - Оцінка часу відгуку бази даних при високому обсязі запитів.
     + **Тестування масштабованості веб-додатків**:
       - Перевірка здатності веб-додатків підтримувати велику кількість одночасних користувачів.

**9.2 Критерії прийняття**

Критерії прийняття визначають, коли система NextPointLogix вважається готовою до впровадження в експлуатацію. Це включає перевірку відповідності вимогам, функціональності, якості та надійності.

1. **Функціональні вимоги**
   * **Опис**: Всі функціональні вимоги, описані в документації, повинні бути реалізовані та перевірені.
   * **Критерії**:
     + Успішне проходження всіх сценаріїв функціонального тестування.
     + Відсутність критичних помилок, що впливають на основну функціональність.
2. **Інтеграція**
   * **Опис**: Усі компоненти системи повинні коректно взаємодіяти один з одним.
   * **Критерії**:
     + Успішне проходження інтеграційного тестування.
     + Відсутність помилок у взаємодії між модулями та зовнішніми системами.
3. **Продуктивність**
   * **Опис**: Система повинна відповідати вимогам до швидкості, масштабованості та надійності.
   * **Критерії**:
     + Успішне проходження навантажувального тестування.
     + Система повинна витримувати заявлене навантаження без значних знижень продуктивності.
4. **Безпека**
   * **Опис**: Система повинна відповідати вимогам безпеки, включаючи захист даних та аутентифікацію.
   * **Критерії**:
     + Перевірка на відповідність стандартам безпеки.
     + Відсутність виявлених вразливостей у тестах безпеки.
5. **Документація**
   * **Опис**: Наявність і коректність документації для користувачів та технічних фахівців.
   * **Критерії**:
     + Наявність повної документації для всіх компонентів системи.
     + Актуальність і точність документації.

**Етапи розробки проекту NextPointLogix**

**1. Планування**

* **Визначення вимог**: Зібрати всі функціональні та нефункціональні вимоги до системи. Провести детальний аналіз бізнес-процесів.
* **Складання ТЗ**: Створити технічне завдання (ТЗ), яке включає опис усіх компонентів системи та їх функцій.

**2. Проектування архітектури**

* **Архітектура системи**: Розробити архітектуру системи, включаючи всі компоненти (мобільні додатки, веб-додатки, бекенд, база даних, інтеграції).
* **ER-діаграма**: Спроектувати схему бази даних (ER-діаграму) для MySQL.
* **API дизайн**: Спроектувати REST API для взаємодії між фронтендом та бекендом.

**3. Розробка бекенду**

* **Середовище розробки**: Налаштувати середовище розробки для Python (Django або Flask).
* **Структура проекту**: Визначити структуру проекту та розробити початкові модулі.
* **Розробка API**: Реалізувати REST API для роботи з даними (працівники, маршрути, водії, витрати пального тощо).

**4. Розробка мобільних додатків**

* **Середовище розробки**: Налаштувати середовище розробки для React Native.
* **Прототипування**: Створити прототипи інтерфейсів для мобільних додатків (працівники, водії).
* **Розробка додатків**: Реалізувати основні функції мобільних додатків (вибір днів, перегляд маршрутів, push-повідомлення).

**5. Розробка веб-додатків**

* Середовище розробки: Налаштувати середовище розробки для React Native.
* **Прототипування**: Створити прототипи інтерфейсів для веб-додатків (логіст-оператор, фінансовий менеджер, адміністратор системи).
* **Розробка додатків**: Реалізувати основні функції веб-додатків (управління маршрутами, облік пального, створення користувачів).

**6. Інтеграція компонентів**

* **Інтеграція фронтенду з бекендом**: Забезпечити взаємодію мобільних та веб-додатків з бекендом через REST API.
* **Тестування**: Провести інтеграційне тестування всіх компонентів системи для забезпечення коректної роботи.

**7. Розгортання**

* **Налаштування AWS**: Налаштувати середовище для хостингу на AWS (EC2, RDS, S3 тощо).
* **CI/CD**: Налаштувати автоматичне розгортання з використанням GitHub Actions або іншого CI/CD інструменту.
* **Розгортання додатків**: Розгорнути всі компоненти системи (бекенд, база даних, мобільні та веб-додатки).

**8. Підтримка та розвиток**

* **Моніторинг та підтримка**: Впровадити системи моніторингу для забезпечення стабільної роботи системи (наприклад, AWS CloudWatch).
* **Зворотний зв’язок та оновлення**: Збирати зворотний зв’язок від користувачів і регулярно оновлювати систему для покращення функціональності та усунення помилок.

**Ресурси та інструменти**

* **Документація**: Створення та підтримка документації для розробників та користувачів.
* **Управління проектом**: Використання інструментів управління проектами (наприклад, Jira або Trello) для відстеження завдань і прогресу.

Архітектура системи

**1. Мобільний додаток для працівників (React Native)**

* **Функції**:
  + Реєстрація та авторизація.
  + Вибір днів для транспортування в календарі.
  + Перегляд інформації про маршрут та орієнтовний час прибуття.
  + Отримання push-повідомлень.
* **Технології**:
  + **Мова програмування**: JavaScript або TypeScript.
  + **Бібліотеки**:
    - **React Navigation**: Для навігації.
    - **React Native Calendar**: Для календаря.
    - **Firebase Cloud Messaging**: Для push-повідомлень.
    - **Axios** або **Fetch API**: Для роботи з REST API.

**2. Мобільний додаток для водіїв (React Native)**

* **Функції**:
  + Прийом та перегляд маршрутів.
  + Доступ до списків пасажирів та їх контактних даних.
  + Перегляд орієнтовного часу посадки та прибуття.
* **Технології**:
  + **Мова програмування**: JavaScript або TypeScript.
  + **Бібліотеки**:
    - **React Navigation**: Для навігації.
    - **Google Maps API**: Для навігації і відображення маршрутів.
    - **Axios** або **Fetch API**: Для роботи з REST API.

**3. Веб-додаток для логіст-оператора**

* **Функції**:
  + Контроль надходження даних від працівників.
  + Складання та коригування маршрутів.
  + Розподіл маршрутів між водіями.
  + Облік пального та витрат на пробіг.
* **Технології**:
  + **Front-end:** PowerApps (з інтеграцією через REST API або інші методи інтеграції).
  + **Back-end**: Python (Django або Flask).
  + **База даних**: MySQL.

**4. Веб-інтерфейс для фінансового менеджера**

* **Функції**:
  + Моніторинг використання транспорту.
  + Розрахунок витрат на паливо і нарахування виплат водіям.
  + Генерація звітів (через PowerBI).
* **Технології**:
  + **Front-end:** PowerApps (з інтеграцією з PowerBI для звітів).
  + **Back-end**: Python (Django або Flask).
  + **База даних**: MySQL.

**5. Веб-інтерфейс для адміністратора системи**

* **Функції**:
  + Створення та управління користувачами.
  + Налаштування прав доступу.
* **Технології**:
  + **Front-end:** PowerApps.
  + **Back-end**: Python (Django або Flask).
  + **База даних**: MySQL.

**Архітектура**

1. **Frontend**:
   * **Мобільні додатки**: Створені з використанням React Native (JavaScript або TypeScript).
   * Веб-додатки: Розроблені з використанням React, інтегровані з бекендом через REST API.
2. **Backend**:
   * **Серверна частина**: Розроблена на Python (Django).
   * **REST API**: Для комунікації між фронтендом (мобільними та веб-додатками) і бекендом.
   * **Обробка логіки маршрутизації, розрахунків витрат пального і фінансових звітів**.
3. **Database**:
   * **MySQL**: Зберігання даних про працівників, водіїв, маршрути, витрати пального і фінансові дані.
4. **Cloud Server**:
   * **AWS**: Хостинг всіх компонентів системи, включаючи базу даних, сервери додатків та API.
5. **CI/CD**:
   * **GitHub**: Для зберігання коду і управління процесом CI/CD.

# AI Integration

The NextPointLogix system will integrate AI to optimize the transportation logistics processes. The AI will handle the following tasks:

## Route Creation

A neural network will be responsible for creating the transportation routes based on the data provided by the employees. The operator will be able to make adjustments to the routes if necessary.

## Fuel Consumption Estimation

The AI will estimate the fuel consumption based on the distance and route information, considering the bus's fuel efficiency.

## Time Calculations

The AI will calculate the approximate time for passenger pickup, travel time, and estimated arrival times at the destination.

Документація проекту, нотатки, хід реалізації, допоміжні файли будуть розміщені на Confluence.

<https://id.atlassian.com/invite/p/confluence?id=w0PBdCgFRVemiLCKXG6a5w>

**Aрхітектура обчислювального ядра.**

**блок-схематичний план**

## ✅ Структура оптимізатора: функціональні блоки

| **№** | **Назва блоку** | **Призначення** |
| --- | --- | --- |
| 1 | **Прийомка та валідація даних** | Перевірити вхідний масив заявок на повноту, правильність та консистентність |
| 2 | **Визначення сценарію задачі** | Встановити, чи обчислюється 1 маршрут, чи кілька; чи є обмеження |
| 3 | **Завантаження обмежень** | Отримати UserSettings з БД для відповідного користувача |
| 4 | **Перевірка відповідності обмеженням** | Перевірити, чи відповідає набір даних умовам (поки що — на 1 маршрут) |
| 5 | **Перетворення заявок на точки** | Заявки → координати (pickup/dropoff) + агрегація по зупинках |
| 6 | **Google Directions API** | Отримання маршрутів: стандартний + з optimize:true |
| 7 | **Поділ на кілька маршрутів** | Алгоритм розбиття (кластеризація, жадібна оптимізація, тощо) |
| 8 | **Збереження результатів** | Через API зберегти до RoutePlanDraft, RouteDraftList |

## 🔄 Пропонований цикл роботи алгоритму

kotlin

CopyEdit

OPTIMIZER ENGINE (build\_optimized\_routes):

1. validate\_requests(data)

→ if errors → return {"errors": [...]}

2. determine\_mode(data)

→ mode = "single" or "multi"

3. settings = load\_user\_constraints(user\_id)

4. if mode == "single":

if not check\_constraints(data, settings):

return {"error": "violated constraints"}

5. coordinates = convert\_requests\_to\_points(data)

6. route\_result = fetch\_google\_route(coordinates, optimize=True)

7. if mode == "multi":

route\_clusters = cluster\_requests(data, settings)

route\_result = [fetch\_google\_route(r) for r in route\_clusters]

8. if save\_flag:

save\_to\_draft(route\_result)

9. return route\_result

## 🧱 Архітектура модулів

| **Модуль** | **Що містить** |
| --- | --- |
| optimizer.py | build\_optimized\_routes(...) - головна функція |
| validation.py | validate\_requests(...) |
| modes.py | determine\_mode(...) |
| constraints.py | load\_user\_constraints(...), check\_constraints(...) |
| conversion.py | convert\_requests\_to\_points(...) |
| google\_maps.py | fetch\_google\_route(...) |
| clustering.py | cluster\_requests(...) |
| persistence.py | save\_to\_draft(...) |

## 📌 Що ще варто врахувати

### 1. **Рівень відмовостійкості**

* кожен блок повинен логувати помилки
* функції повинні повертати чітко визначену структуру (наприклад, {success: False, errors: [...]})

### 2. **Можливість асинхронного виконання**

* для подальшої підтримки Celery/Task Queue можна зробити підготовку даних → task\_id

### 3. **Тип оптимізації (на майбутнє)**

* можна дозволити strategy="fuel", strategy="duration", strategy="vehicles"
* залежно від стратегії — адаптувати cluster\_requests або навіть вибирати інший engine

## ✅ Висновок

Ти врахував усе основне. Ось додаткові **підсилювачі**, які можна додати:

* підтримка попередньо обчислених зупинок (щоб не повторювати виклики Google Maps)
* кешування маршрутів
* збереження логів обчислень (для аналізу та навчання)
* перевірка, чи заявка вже включена в план (Request.used\_in\_plan == True)

### ✅ **Алгоритм побудови оптимізованого маршруту (оновлений)**

#### **1. Розширення структури Point**

* Point тепер має містити:
  + lat, lng, id
  + ➕ point\_type: "pickup" або "dropoff" — **визначає, яку саме точку подано**

#### **2. Визначення точки роботи (work\_point)**

* Фільтруються всі точки з point\_type == "dropoff" якщо TO\_WORK, або pickup якщо WORK\_TO\_HOME.
* Якщо всі координати приблизно однакові (різниця < 0.0001) — беремо першу.
* Якщо кілька варіантів — беремо найчастішу координату (lat,lng) серед таких точок як **work\_point**.

#### **3. Визначення напряму**

* Заявки містять поле direction, яке має значення TO\_WORK або WORK\_TO\_HOME.

#### **4. Визначення другої точки маршруту**

* Обчислюється **найвіддаленіша точка від work\_point** серед усіх pickup/dropoff точок, **крім самого work\_point** — назвемо її furthest\_from\_work.

#### **5. Формування повного списку точок**

* Створюємо послідовність:

python

CopyEdit

[work\_point, ...всі pickup/dropoff точки, furthest\_from\_work]

(тобто загальна кількість = 2 \* len(requests) + 2)

#### **6. Запит на маршрут Google Directions API**

* Прокладаємо маршрут по точках work\_point -> всі -> furthest\_from\_work, Google повертає waypoint\_order.

#### **7. Присвоєння sequence\_number заявкам**

* Для кожної заявки:
  + визначаємо, де її pickup і dropoff координати зустрілися у optimized\_route.stops
  + Для TO\_WORK: сортуємо заявки **за віддаленістю до work\_point**, найближчий сідає останнім
  + Для WORK\_TO\_HOME: першим висаджується той, хто найближче

#### **8. Повернення результату**

* Повертається:
  + **всі заявки з оновленим sequence\_number**
  + **точки work\_point і furthest\_from\_work**
  + **дані маршруту: stops, distance, duration**