Міністерство освіти і науки України
Національний університет «Одеська політехніка»
Інститут комп’ютерних систем
Кафедра інформаційних систем

Пояснювальна записка
до курсової роботи з дисципліни
«Об’єктно-орієнтоване програмування»
на тему: «Система управління орендою техніки»

Виконав:
Студент групи АІ-233
Михайлов Д. П.
Перевірили:
Годовиченко М.А.

Одеса 2025

Вступ

У сучасних умовах розвитку ринку послуг оренди техніки спостерігається стрімке зростання попиту на автоматизацію бізнес-процесів. Це пов’язано з

необхідністю швидко, якісно та ефективно обслуговувати клієнтів, забезпечувати належний облік технічних засобів, контролювати фінансові операції та оптимізувати використання ресурсів компанії. Оренда техніки охоплює широкий спектр обладнання — від будівельної та сільськогосподарської техніки до комп'ютерної, офісної й спеціалізованої апаратури. Такий різноманітний парк техніки потребує ретельного управління, що унеможливлює використання лише традиційних, паперових чи неструктурованих електронних методів обліку.

Впровадження сучасної інформаційної системи управління орендою техніки дозволяє вирішити низку важливих завдань: автоматизувати процеси бронювання та повернення обладнання, вести точний облік наявних одиниць техніки, контролювати терміни експлуатації, своєчасно проводити технічне обслуговування, а також аналізувати фінансові показники діяльності компанії. Завдяки цьому підвищується рівень сервісу для клієнтів, зменшується кількість помилок, пов'язаних із людським фактором, і зростає конкурентоспроможність підприємства.

Крім того, сучасна система управління орендою техніки повинна забезпечувати зручний інтерфейс для різних категорій користувачів (адміністраторів, менеджерів, клієнтів), підтримувати інтеграцію з іншими інформаційними системами, гарантувати безпеку та захист персональних даних. Важливою є можливість масштабування програмного продукту відповідно до потреб бізнесу, а також оперативне впровадження нових функцій.

Метою цієї роботи є розробка ефективної, надійної та гнучкої системи управління орендою техніки, яка дозволить автоматизувати основні бізнес-процеси компанії, підвищити якість обслуговування клієнтів та забезпечити прозорість діяльності підприємства. У пояснювальній записці розглядаються основні етапи створення такого програмного забезпечення: аналіз предметної області, проєктування архітектури, реалізація функціональних модулів та оцінка ефективності впровадження системи.

1. Аналіз предметної області

Система управління орендою техніки створена для автоматизації процесів компаній, що надають послуги тимчасового користування спеціалізованим обладнанням. Основна мета такої системи — забезпечити ефективну взаємодію між клієнтами, адміністраторами та технікою, а також оптимізувати облік ресурсів і фінансових операцій. В системі зберігається та обробляється інформація про техніку, клієнтів, орендні угоди та платіжні транзакції, що дозволяє підвищити якість обслуговування і зменшити ризики помилок.

Ключовими об’єктами системи є техніка, клієнти, замовлення (орендні угоди), оплати та категорії техніки. Техніка — це унікальні одиниці обладнання, кожна з яких має свої характеристики: назву, тип (наприклад, будівельна, сільськогосподарська, офісна), серійний номер, стан (доступна, в оренді, на обслуговуванні) та ціну за добу. Клієнти можуть бути як фізичними, так і юридичними особами, з інформацією про ПІБ або назву організації, контактні дані та історію оренд.

Замовлення містять дані про конкретну одиницю техніки, клієнта, дати початку і завершення оренди, загальну вартість та статус угоди (підтверджено, скасовано, завершено). Оплати відображають суму, дату, спосіб проведення транзакції (готівка або безготівка) та статус платежу. Категорії техніки дозволяють класифікувати обладнання для зручного пошуку і фільтрації.

Життєвий цикл обслуговування клієнта в системі включає кілька основних етапів. Спершу користувач проходить реєстрацію або авторизацію. Далі він може здійснити пошук техніки за категоріями, датами та ціною. Після вибору обладнання клієнт оформлює бронювання, вказуючи терміни оренди, і отримує підтвердження угоди. Наступним кроком є оплата послуг, яка може бути проведена онлайн або офлайн. Після цього клієнт отримує техніку у користування на визначений період. По завершенню терміну оренди відбувається повернення обладнання, перевірка його стану і закриття угоди. За бажанням клієнт може залишити відгук, а система зберігає історію оренд для подальшого аналізу.

Функціональні можливості системи охоплюють автоматизацію створення і скасування орендних угод, контроль доступності техніки в реальному часі, розрахунок вартості оренди з урахуванням термінів та можливих знижок, а також генерацію різноманітних звітів. Звіти включають фінансові показники, популярність категорій техніки, аналіз завантаженості обладнання та статистику клієнтської активності. Для зручності користувачів передбачена інтеграція з платіжними системами, що дозволяє здійснювати онлайн-оплати.

Для адміністраторів система надає можливість миттєво оновлювати інформацію про стан техніки, додавати нові одиниці обладнання або редагувати існуючі записи. Вони також мають доступ до детальних звітів, які допомагають у стратегічному плануванні, наприклад, у прийнятті рішень щодо закупівлі популярних категорій техніки. Крім того, система автоматично надсилає нагадування клієнтам про терміни повернення обладнання, що знижує ризик прострочень.

Загалом, впровадження системи управління орендою техніки забезпечує прозорість усіх етапів роботи компанії, зменшує вплив людського фактора, підвищує лояльність клієнтів завдяки зручному та швидкому сервісу, а також сприяє підвищенню ефективності бізнесу в цілому.

2. Проєктування системи

2.1 Опис сутностей

У системі управління орендою техніки визначено п’ять основних сутностей, які відображають ключові об’єкти предметної області:

Equipment (Техніка) — представляє одиницю обладнання, доступну для оренди. Кожна одиниця має унікальний ідентифікатор (id), назву (name), тип (type), добову ставку оренди (dailyRate) та ознаку доступності (availability), яка вказує, чи техніка наразі вільна для оренди.

Customer (Клієнт) — фізична або юридична особа, що орендує техніку. Містить унікальний ідентифікатор (id), ім’я клієнта (name) та контактний телефон (phone).

Employee (Співробітник) — працівник компанії, який опрацьовує оренду і взаємодіє з клієнтами. Має унікальний ідентифікатор (id), ім’я (name) та посаду (position).

Rental (Оренда) — замовлення на оренду техніки, що зв’язує конкретну одиницю обладнання, клієнта та співробітника, який обробляє угоду. Містить ідентифікатор (id), посилання на техніку (equipment), клієнта (customer) та співробітника (employee), дати початку (startDate) та завершення оренди (endDate), а також статус повернення (returned), що вказує, чи техніка була повернена.

Maintenance (Технічне обслуговування) — записи про проведені роботи з обслуговування техніки. Містить ідентифікатор (id), посилання на обладнання (equipment), дату проведення (date) та опис виконаних робіт (description).

2.2 Опис архітектури застосунку

Застосунок реалізовано за допомогою трирівневої архітектури, яка складається з трьох основних компонентів:

Controller — цей шар відповідає за обробку HTTP-запитів від клієнтів. Контролер приймає параметри запиту, викликає відповідні методи сервісного шару та повертає результати у форматі JSON. Він виконує роль посередника між користувацьким інтерфейсом і бізнес-логікою.

Service — містить бізнес-логіку застосунку. Тут відбуваються всі перевірки, трансформації даних, обчислення та забезпечується послідовність дій. Сервісний шар викликає методи репозиторіїв для роботи з базою даних і повертає дані у вигляді DTO (Data Transfer Object), що дозволяє відокремити внутрішню структуру даних від зовнішніх представлень.

Repository — цей шар відповідає за взаємодію з базою даних. Використовується Spring Data JPA, що дозволяє абстрагуватися від написання SQL-запитів, надаючи стандартні методи для збереження, оновлення, видалення та пошуку сутностей.

Взаємозв’язок між цими компонентами організовано таким чином: контролер залежить від сервісу, передаючи йому запити користувачів для обробки; сервіс, у свою чергу, залежить від репозиторіїв, через які здійснюється робота з базою даних; репозиторії напряму працюють із сутностями у базі.

2.3 Опис REST API

Для взаємодії з системою передбачено набір REST-запитів, які забезпечують повний функціонал керування орендою техніки:

1. Додати техніку — створення нового запису про обладнання.
2. Отримати всі одиниці техніки — отримання списку всієї техніки.
3. Оновити техніку — редагування інформації про існуючу одиницю техніки.
4. Видалити техніку — видалення запису про техніку.
5. Додати клієнта — реєстрація нового клієнта.
6. Отримати список клієнтів — отримання інформації про всіх клієнтів.
7. Оновити дані клієнта — редагування інформації про клієнта.
8. Видалити клієнта — видалення клієнта з системи.
9. Додати співробітника — реєстрація нового працівника.
10. Отримати співробітників — отримання списку співробітників.
11. Оновити співробітника — редагування даних працівника.
12. Видалити співробітника — видалення співробітника.
13. Оформити оренду — створення нового замовлення на оренду техніки.
14. Отримати оренди клієнта — перегляд усіх оренд, оформлених певним клієнтом.
15. Отримати оренди співробітника — перегляд оренд, опрацьованих певним співробітником.
16. Завершити оренду — позначення оренди як завершеної (повернення техніки).
17. Додати запис про техобслуговування — реєстрація нової події обслуговування техніки.
18. Отримати обслуговування певної одиниці техніки — перегляд історії технічного обслуговування конкретного обладнання.
19. Отримати всі активні оренди — отримання списку оренд, які наразі тривають.

20. Отримати доступну техніку на дату — пошук вільного обладнання на вказану дату.
21. Отримати кількість оренд техніки — статистика по кількості орендованих одиниць техніки.
22. Отримати загальний дохід від оренди — звіт про сумарний прибуток від орендної діяльності.
23. Отримати найбільш орендовану техніку — визначення найпопулярнішої одиниці обладнання за кількістю оренд.
24. Отримати записи з простроченим поверненням — список оренд, в яких клієнти затримали повернення техніки.
25. Видалити запис оренди — видалення замовлення на оренду з системи.

3. Реалізація програмного продукту

3.1 Моделі даних

Для відображення сутностей предметної області у програмному коді створюються відповідні моделі (класи), які відповідають таблицям у базі даних. Кожна модель містить поля, що відображають атрибути сутності, а також зв’язки між сутностями (наприклад, зв’язок "один-до-багатьох" між клієнтом та орендою). У проєкті реалізовані такі основні моделі: Customer Employee, Equipment, Maintenance та Rental.

Клас Customer представляє клієнта системи оренди обладнання та відображає відповідну таблицю у базі даних. Вона містить основні атрибути клієнта, а також зв’язок із орендою, що дозволяє відстежувати всі оренди, які належать конкретному клієнту.

java-код для класу Customer:

```
@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
@JsonIdentityInfo(generator =
ObjectIdGenerators.PropertyGenerator.class, property = "id")
public class Customer {


    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;


    private String name;
    private String phone;


    @OneToMany(mappedBy = "customer", cascade = CascadeType.ALL,
orphanRemoval = true)
    private List<Rental> rentals;
}
```

Клас Employee відображає працівника, який відповідає за оформлення оренд. Модель містить основні атрибути працівника та зв’язок із орендами, які він обробляє.

java-код для класу Employee:

```
@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
@JsonIdentityInfo(generator =
ObjectIdGenerators.PropertyGenerator.class, property = "id")
public class Employee {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String name;
    private String position;

    @OneToMany(mappedBy = "employee", cascade = CascadeType.ALL,
orphanRemoval = true)
    private List<Rental> rentals;
}
```

Клас Equipment представляє обладнання, що здається в оренду. Модель містить інформацію про обладнання, його тип, ціну за день оренди, доступність, а також зв’язки з орендами і записами про обслуговування.

java-код для класу Equipment:

```
@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
@JsonIdentityInfo(generator =
ObjectIdGenerators.PropertyGenerator.class, property = "id")
public class Equipment {
```

```
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String name;
    private String type;
    private BigDecimal dailyRate;
    private Boolean availability;

    @OneToMany(mappedBy = "equipment", cascade = CascadeType.ALL,
orphanRemoval = true)
    private List<Rental> rentals;

    @OneToMany(mappedBy = "equipment", cascade = CascadeType.ALL,
orphanRemoval = true)
    private List<Maintenance> maintenanceRecords;
}
```

Клас Maintenance описує записи про технічне обслуговування обладнання.

java-код для класу Maintenance:

```
@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
@JsonIdentityInfo(generator =
ObjectIdGenerators.PropertyGenerator.class, property = "id")
public class Maintenance {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "equipment_id")
```

```
    private Equipment equipment;

    private LocalDate date;
    private String description;
}
```

Клас Rental відображає інформацію про оренду обладнання клієнтом, включаючи зв’язки з клієнтом, працівником і обладнанням.

java-код для класуRental:

```
@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
@JsonIdentityInfo(generator =
ObjectIdGenerators.PropertyGenerator.class, property = "id")
public class Rental {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "equipment_id")
    private Equipment equipment;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "customer_id")
    private Customer customer;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "employee_id")
    private Employee employee;
```

```
    private LocalDate startDate;
    private LocalDate endDate;
    private Boolean returned;
}
```

Також для кожної з моделей було створено DTO класи. Нижче будуть наведені приклади DTO класів:

```
public record CustomerDTO(
        Long id,
        String name,
        String phone
)
public record EmployeeDTO(
        Long id,
        String name,
        String position
)
public record EquipmentDTO(
        Long id,
        String name,
        String type,
        BigDecimal dailyRate,
        Boolean availability
)
public class MaintenanceRequestDTO {
    private Long equipmentId;
    private LocalDate date;
    private String description;
}
public class MaintenanceResponseDTO {
    private Long id;
    private Long equipmentId;
    private LocalDate date;
    private String description;
```

```
    public MaintenanceResponseDTO(Maintenance maintenance) {
        this.id = maintenance.getId();
        this.equipmentId = maintenance.getEquipment().getId();
        this.date = maintenance.getDate();
        this.description = maintenance.getDescription();
    }
}
public class RentalDTO {
    private Long id;
    private Long equipmentId;
    private String equipmentName;
    private Long customerId;
    private String customerName;
    private Long employeeId;
    private String employeeName;
    private LocalDate startDate;
    private LocalDate endDate;
    private Boolean returned;
}
public class RentalRequestDTO {
    private Long equipmentId;
    private Long customerId;
    private Long employeeId;
    private LocalDate startDate;
    private LocalDate endDate;
    private Boolean returned;
}
```

3.2 Репозиторії

Для роботи з базою даних у проєкті реалізовані репозиторії, які використовують ORM-фреймворк Spring Data JPA. Репозиторії забезпечують абстракцію над SQL-запитами, дозволяючи працювати з даними на рівні об’єктів — створювати, оновлювати, видаляти та шукати сутності.

Основні репозиторії проєкту:

CustomerRepository

```
public interface CustomerRepository extends
JpaRepository<Customer, Long> {
}
```

Наслідує базові CRUD-методи (save, findById, findAll, delete тощо) від JpaRepository. Дозволяє працювати з клієнтами без необхідності писати SQL-запити.

EmployeeRepository

```
public interface EmployeeRepository extends
JpaRepository<Employee, Long> {
}
```

Забезпечує стандартні операції для працівників.

EquipmentRepository

```
public interface EquipmentRepository extends
JpaRepository<Equipment, Long> {

    @Query("SELECT e FROM Equipment e WHERE e.availability = true
OR e.id NOT IN " +
            "(SELECT r.equipment.id FROM Rental r WHERE " +
            "(r.startDate <= :date AND r.endDate >= :date))")
    List<Equipment> findAvailableOnDate(LocalDate date);
}
```

Містить кастомний метод findAvailableOnDate(LocalDate date), який повертає обладнання, доступне для оренди на задану дату. Запит перевіряє, чи обладнання позначене як доступне (availability = true) або не зайняте орендою у вказаний період.

MaintenanceRepository

```
public interface MaintenanceRepository extends
JpaRepository<Maintenance, Long> {
```

```
    List<Maintenance> findByEquipmentId(Long equipmentId);
}
```

Додатковий метод findByEquipmentId повертає всі записи обслуговування для конкретного обладнання.

RentalRepository

```
public interface RentalRepository extends JpaRepository<Rental,
Long> {

    List<Rental> findByCustomer_Id(Long customerId);

    List<Rental> findByEmployee_Id(Long employeeId);

    List<Rental> findByReturnedFalse();

    @Query("SELECT r.equipment.id, COUNT(r) FROM Rental r GROUP BY
r.equipment.id")
    List<Object[]> countRentalsByEquipment();

    @Query("SELECT r.equipment.id FROM Rental r GROUP BY
r.equipment.id ORDER BY COUNT(r) DESC")
    List<Long> findMostRentedEquipment();

    @Query("SELECT r FROM Rental r WHERE r.endDate < CURRENT_DATE
AND r.returned = false")
    List<Rental> findOverdueRentals();
}
```

Методи для пошуку оренд за клієнтом (findByCustomer_Id), працівником (findByEmployee_Id), а також для пошуку оренд, які ще не повернені (findByReturnedFalse).

Кастомні запити:

countRentalsByEquipment — підрахунок кількості оренд по кожному обладнанню.

findMostRentedEquipment — пошук найбільш популярного обладнання за кількістю оренд.

findOverdueRentals — пошук оренд із простроченим терміном повернення.

3.3 Сервіси

У проєкті реалізовано сервісний шар, який відповідає за бізнес-логіку, обробку даних, перевірки, трансформацію моделей у DTO та навпаки. Сервіси взаємодіють із репозиторіями для виконання CRUD-операцій, забезпечують узгодженість дій та є посередниками між контролерами й репозиторіями. Кожна основна сутність має відповідний сервіс, який інкапсулює логіку роботи з цією сутністю.

CustomerService

Сервіс для роботи з клієнтами (Customer). Реалізує основні операції:

— Додавання нового клієнта(зберігає клієнта в базі даних і повертає DTO):

```
public CustomerDTO addCustomer(Customer customer)
```

— Отримання всіх клієнтів(повертає список усіх клієнтів у вигляді DTO):

```
public List<CustomerDTO> getAllCustomers()
```

— Отримання клієнта за id:

```
public Optional<CustomerDTO> getCustomerById(Long id)
```

— Оновлення даних клієнта(оновлює ім’я та телефон клієнта):

```
public CustomerDTO updateCustomer(Long id, Customer
updatedCustomer)
```

— Видалення клієнта:

```
public void deleteCustomer(Long id)
```

— Конвертація між моделлю та DTO:

```
public CustomerDTO toDTO(Customer customer)
public Customer toEntity(CustomerDTO dto)
```

EmployeeService

Сервіс для роботи з працівниками (Employee). Основні функції аналогічні до сервісу клієнтів:

— Додавання, отримання списку, отримання за id, оновлення, видалення працівника.

— Конвертація між моделлю та DTO.

EquipmentService

Сервіс для роботи з обладнанням (Equipment):

— Додавання нового обладнання.

— Отримання списку всього обладнання.

— Отримання обладнання за id.

— Оновлення даних обладнання.

— Видалення обладнання.

— Отримання обладнання, доступного для оренди на конкретну дату:

```
public List<EquipmentDTO> getAvailableEquipmentOnDate(LocalDate
date)
```

— Конвертація між моделлю та DTO.

MaintenanceService

Сервіс для роботи із записами обслуговування (Maintenance):

— Додавання нового запису обслуговування на основі DTO:

```
public Maintenance addMaintenanceFromDTO(MaintenanceRequestDTO
dto)
```

— Отримання всіх записів обслуговування для конкретного обладнання:

```
public List<MaintenanceResponseDTO>
getMaintenanceByEquipmentDTO(Long equipmentId)
```

— Отримання запису за id та видалення запису.

RentalService

Сервіс для роботи з орендами (Rental):

— Створення нової оренди як напряму, так і з DTO-запиту:

```
public Rental createRental(Rental rental)
public Rental createRentalFromDTO(RentalRequestDTO dto)
```

— Отримання оренд за клієнтом, працівником, активних оренд.

— Отримання оренди за id.

— Завершення оренди (позначення як поверненої):

```
public Rental completeRental(Long rentalId)
```

— Видалення оренди.

—Підрахунок кількості оренд по обладнанню, визначення найбільш популярного обладнання.

— Підрахунок загального доходу від оренд:

```
public double getTotalIncome()
```

— Отримання прострочених оренд.

— Конвертація між моделлю та DTO.

3.4 Контролери

Контролери у проєкті реалізують REST API для взаємодії з основними сутностями системи. Вони відповідають за обробку HTTP-запитів, отримання параметрів з URL чи тіла запиту, виклик відповідних методів сервісного шару та повернення результатів у форматі JSON. Контролери забезпечують зручний і стандартизований інтерфейс для клієнтських застосунків і сторонніх систем.
Основні контролери системи

CustomerController

— POST /api/customers — створення нового клієнта.
Приймає об'єкт CustomerDTO у тілі запиту, конвертує його у модель, зберігає через сервіс та повертає створеного клієнта у вигляді DTO.

— GET /api/customers — отримання списку всіх клієнтів.
Повертає список CustomerDTO.

— GET /api/customers/{id} — отримання клієнта за ідентифікатором.
Якщо клієнта знайдено, повертає CustomerDTO, інакше — статус 404.

— PUT /api/customers/{id} — оновлення даних клієнта.
Приймає CustomerDTO, оновлює дані клієнта за id.

— DELETE /api/customers/{id} — видалення клієнта за id.

EmployeeController

— POST /api/employees — створення нового працівника.

— GET /api/employees — отримання списку працівників.

— GET /api/employees/{id} — отримання працівника за id.

— PUT /api/employees/{id} — оновлення даних працівника.

— DELETE /api/employees/{id} — видалення працівника.

EquipmentController

— POST /api/equipment — створення нового обладнання.

— GET /api/equipment — отримання списку всього обладнання.

— GET /api/equipment/{id} — отримання обладнання за id.

— PUT /api/equipment/{id} — оновлення даних обладнання.

— DELETE /api/equipment/{id} — видалення обладнання.

— GET /api/equipment/available?date=YYYY-MM-DD — отримання списку обладнання, доступного для оренди на вказану дату.

MaintenanceController

— POST /api/maintenance — створення запису обслуговування обладнання.

— GET /api/maintenance/{equipmentId} — отримання всіх записів обслуговування для конкретного обладнання.

— GET /api/maintenance/{id}/detail — отримання детальної інформації про запис обслуговування за id.

— DELETE /api/maintenance/{id} — видалення запису обслуговування.

RentalController

— POST /api/rentals — створення нової оренди на основі RentalRequestDTO.

— GET /api/rentals/customer/{customerId} — отримання всіх оренд для конкретного клієнта.

— GET /api/rentals/employee/{employeeId} — отримання всіх оренд, оформлених певним працівником.

— GET /api/rentals/active — отримання всіх активних (не повернених) оренд.

— GET /api/rentals/{id} — отримання оренди за id.

— PUT /api/rentals/{id}/complete — позначення оренди як поверненої.

— DELETE /api/rentals/{id} — видалення оренди.

— GET /api/rentals/count-by-equipment — отримання статистики кількості оренд по кожному обладнанню.

— GET /api/rentals/total-income — отримання загального доходу від оренд.

— GET /api/rentals/most-rented-equipment — отримання списку найбільш популярного обладнання.

— GET /api/rentals/overdue — отримання списку прострочених оренд.

3.5 Обробка помилок і валідація

У системі реалізовано механізми обробки помилок та валідації вхідних даних для забезпечення коректної роботи застосунку, запобігання некоректним операціям і надання користувачам зрозумілих повідомлень про помилки.

Валідація вхідних даних здійснюється як на рівні DTO/моделей (наприклад, через анотації @NotNull, @Email, @Size у класах DTO), так і на рівні сервісного шару, де додатково перевіряється коректність бізнес-логіки (наявність сутностей, унікальність значень, допустимість операцій тощо). У разі виявлення помилкових або некоректних даних система повертає відповідний HTTP-статус (наприклад, 400 Bad Request) та інформативне повідомлення про суть помилки.

Обробка помилок реалізується через використання конструкцій try-catch у сервісах та контролерах, а також глобальних обробників винятків (наприклад, через анотацію @ControllerAdvice). При виникненні типових ситуацій — таких як відсутність сутності у базі (EntityNotFoundException), спроба створення дублікату (DataIntegrityViolationException), некоректний формат даних — система повертає відповідні коди помилок (404, 409, 400) та текстові пояснення.

Приклад обробки помилок у контролері:

```
@PutMapping("/{id}")
public ResponseEntity<CustomerDTO> updateCustomer(@PathVariable
Long id, @RequestBody CustomerDTO customerDTO) {
    try {
        Customer updated = customerService.toEntity(customerDTO);
        return
ResponseEntity.ok(customerService.updateCustomer(id, updated));
    } catch (RuntimeException e) {
        return ResponseEntity.notFound().build();
    }
}
```

У цьому прикладі, якщо клієнта з вказаним id не знайдено, повертається статус 404.

4. Реєстрація та автентифікація

У системі реалізовано повноцінний механізм реєстрації, автентифікації та авторизації користувачів із підтримкою ролей (адміністратор, користувач) і

захистом REST API за допомогою JWT (JSON Web Token). Це забезпечує безпечний доступ до ресурсів застосунку та розмежування прав користувачів. Основні компоненти безпеки

— Модель користувача (AppUser) — містить поля username, password, role та реалізує інтерфейс UserDetails для інтеграції зі Spring Security.

— Репозиторій користувачів (UserRepository) — забезпечує збереження та пошук користувачів у базі даних.

— Сервіси безпеки (AppDetailsService, AppService, CustomOAuth2Service) — відповідають за завантаження користувача, реєстрацію, кодування паролів, підтримку OAuth2.

— JWT-компоненти (AuthTokenGenerator, AuthTokenFilter) — генерують, перевіряють та обробляють JWT-токени для захисту REST API.

— Spring Security конфігурація (SecurityConfig) — визначає правила доступу до маршрутів, налаштовує фільтри, логін/логаут, інтеграцію з OAuth2.

Реєстрація користувача

Реєстрація доступна як через REST API, так і через форму:

REST API:

```
POST /api/register
{
  "username": "user@example.com",
  "password": "password"
}
```

https://equipment-rental-coursework.onrender.com/login

Якщо ім’я користувача вже зайняте — повертається помилка 400. Пароль зберігається у зашифрованому вигляді.

Форма:

```
GET /register — повертає сторінку реєстрації.
POST /register — обробляє дані форми, кодує пароль, зберігає користувача з вибраною роллю.
```

Вхід та автентифікація

REST API:

```
POST /api/login
{
  "username": "user@example.com",
  "password": "password"
}
```

У відповідь повертається JWT-токен, який потрібно використовувати для доступу до захищених ресурсів (у заголовку Authorization: Bearer ...).

Форма:

```
GET /login — повертає сторінку входу.
POST /form-login-token — автентифікує користувача, повертає токен для подальшої роботи.
```

OAuth2: підтримується вхід через Google та інші OAuth2-провайдери. Якщо користувач входить перший раз — створюється відповідний запис у базі.

Авторизація та захист ресурсів

Доступ до більшості маршрутів дозволений лише автентифікованим користувачам.

— Доступ до /admin — лише для користувачів з роллю ADMIN.

— Доступ до /user — лише для користувачів з роллю USER.

— Доступ до /api/register, /api/login, /register, /login, /oauth2/** — відкритий для всіх.

— Для REST API використовується JWT-фільтр, який перевіряє токен у кожному запиті.

Обробка помилок безпеки

— При невірному логіні або паролі повертається відповідне повідомлення про помилку.

— При спробі доступу до захищених ресурсів без токена або з недійсним токеном — статус 401 Unauthorized.

— При спробі доступу до маршруту без достатніх прав — статус 403 Forbidden.

5. Тестування програмного продукту

Для перевірки правильності роботи функціоналу обробки зображень у системі було проведено ручне тестування через Postman та автоматичне тестування за допомогою юніт-тестів. Тестування охоплювало: завантаження файлів, перегляд, видалення, перевірку формату файлу та його розміру, перевірку авторизації доступу.

Реєстрація нового користувача

Метод: POST

URL: http://localhost:8080/api/register

Очікування: Користувач з логіном та паролем зберігається в базі даних, пароль шифрується, присвоюється роль ROLE_USER.

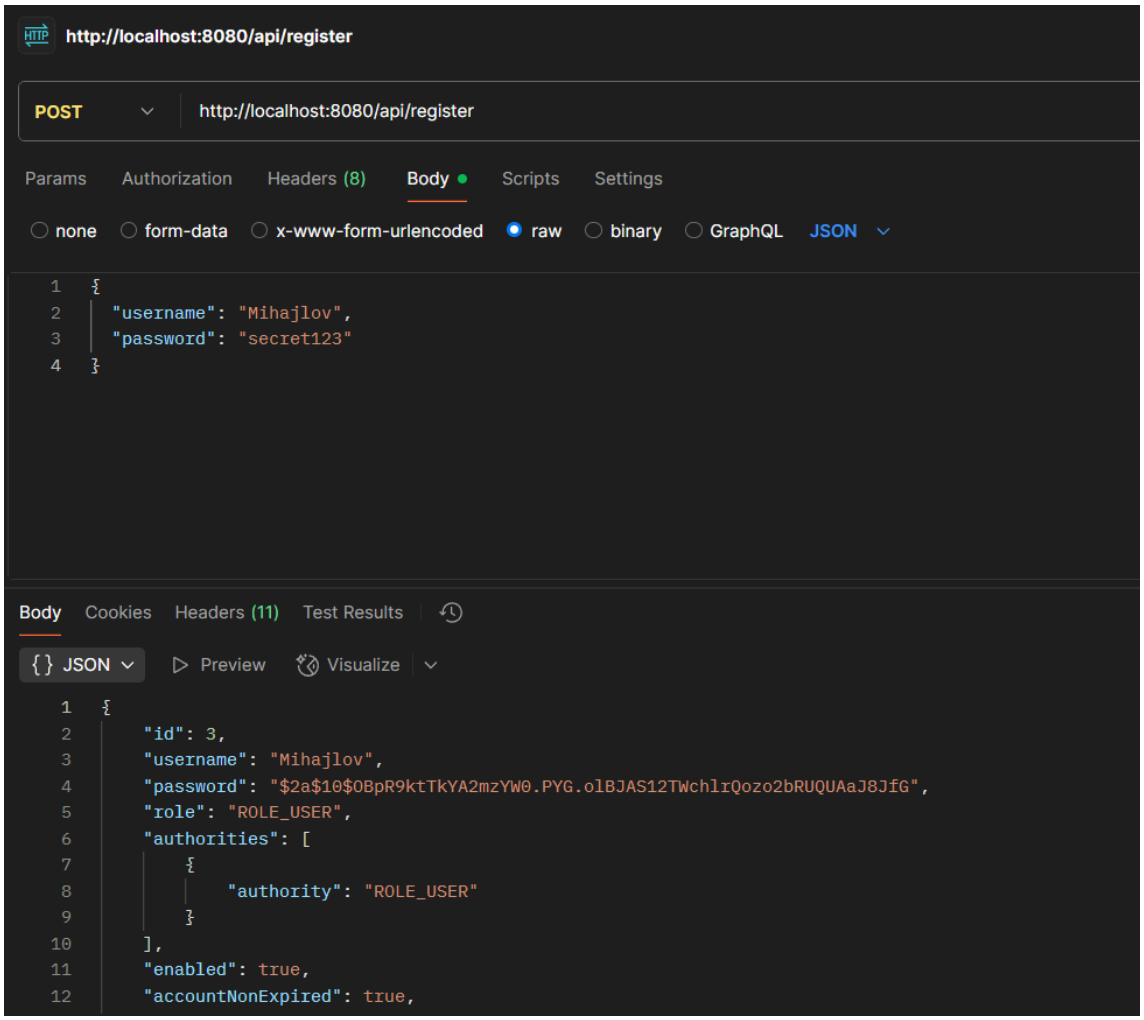

Рисунок 1 – Успішна реєстрація нового користувача через Postman

Авторизація користувача (JWT login)

Метод: POST

URL: http://localhost:8080/api/login

Очікування: У відповідь повертається JWT токен, який використовується для подальших авторизованих запитів.

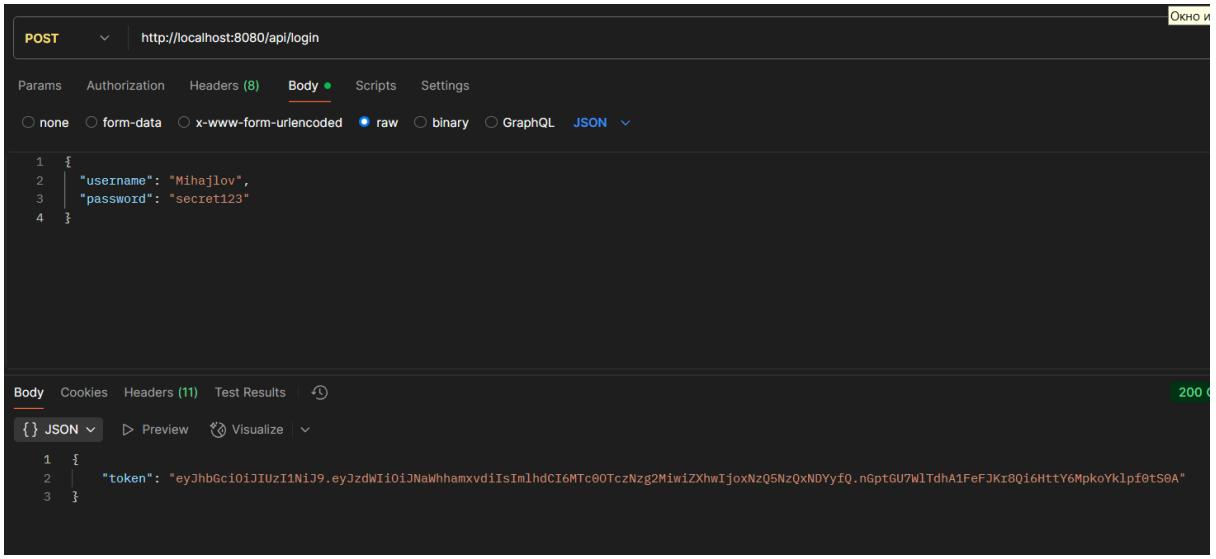

Рисунок 2 – Успішна авторизація та отримання токена

Створення нової одиниці техніки

Метод: POST

URL: http://localhost:8080/api/equipment

Очікування: У відповідь повертається об'єкт техніки з унікальним ідентифікатором (`id`) та переданими параметрами: назва, тип, добова вартість. Цей запит дозволяє додати нову одиницю обладнання до системи управління орендою.

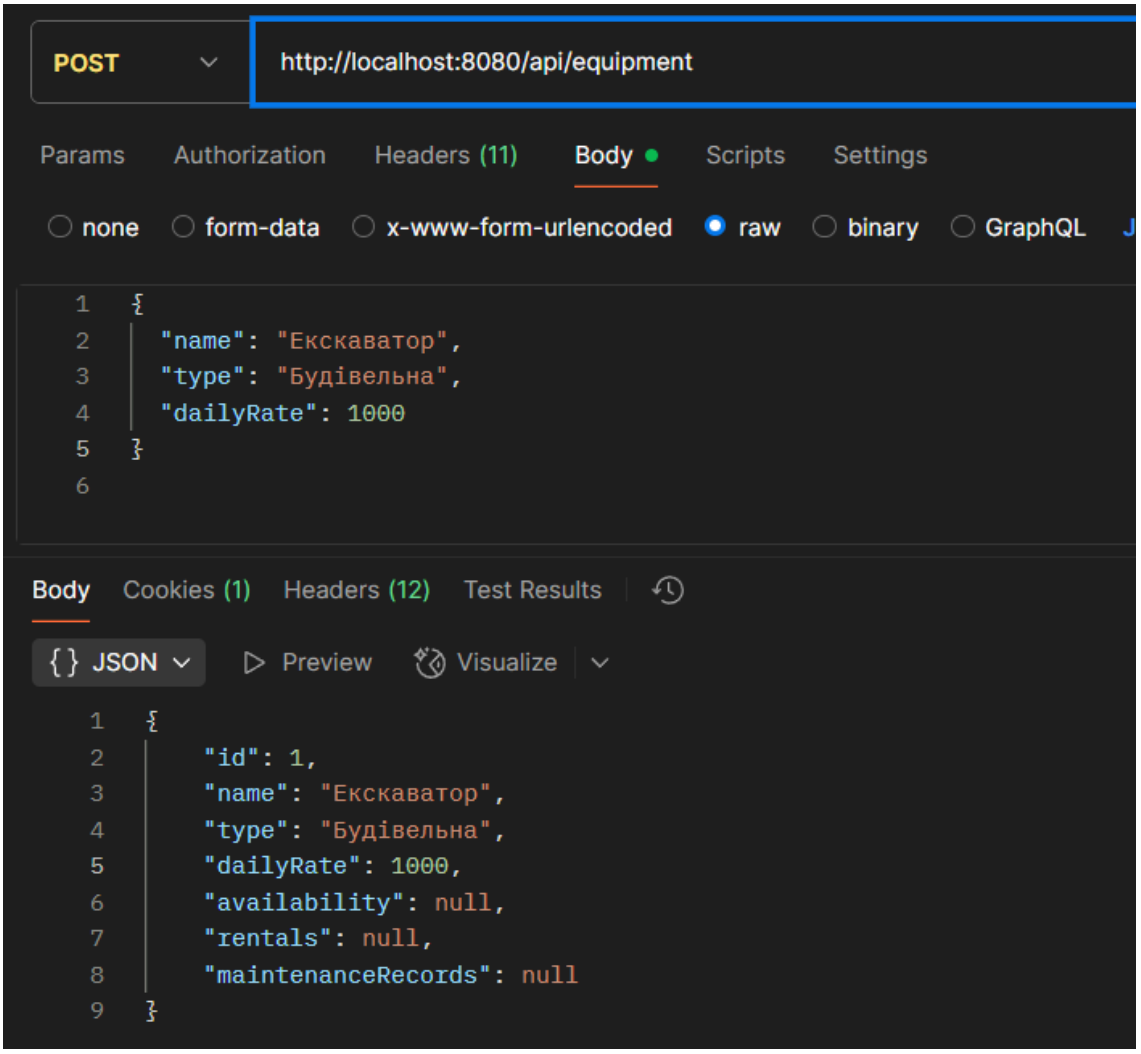

Рисунок 3 – Успішне створення нової одиниці техніки

Отримання списку обладнання

Метод: GET

URL: http://localhost:8080/api/equipment

Очікування: У відповідь повертається масив об’єктів обладнання у форматі JSON. Кожен об’єкт містить ідентифікатор, назву, тип, денну ставку оренди, наявність, історію оренд та обслуговування.

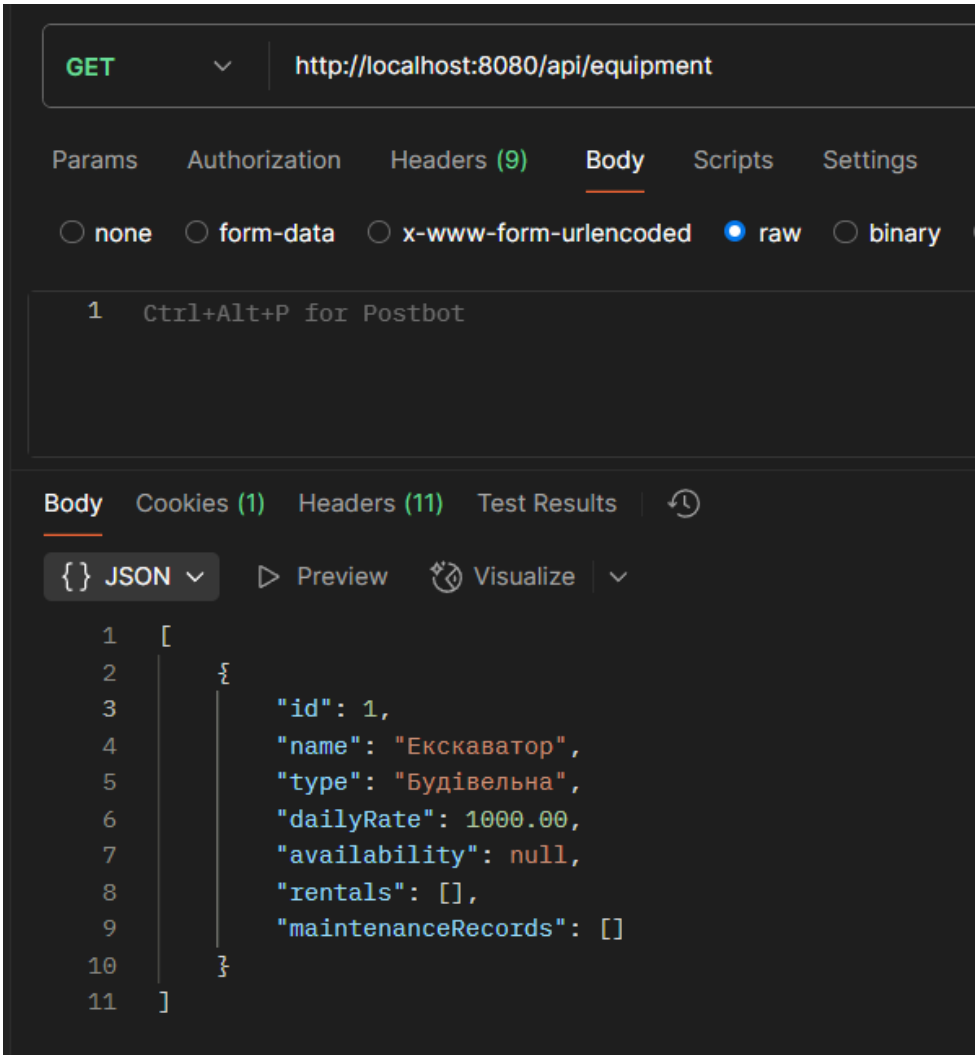

Рисунок 4 – Успішне отримання списку обладнання

Оновлення даних обладнання

Метод: PUT

URL: http://localhost:8080/api/equipment/1

Очікування: У відповідь повертається оновлений об’єкт обладнання з внесеними змінами.

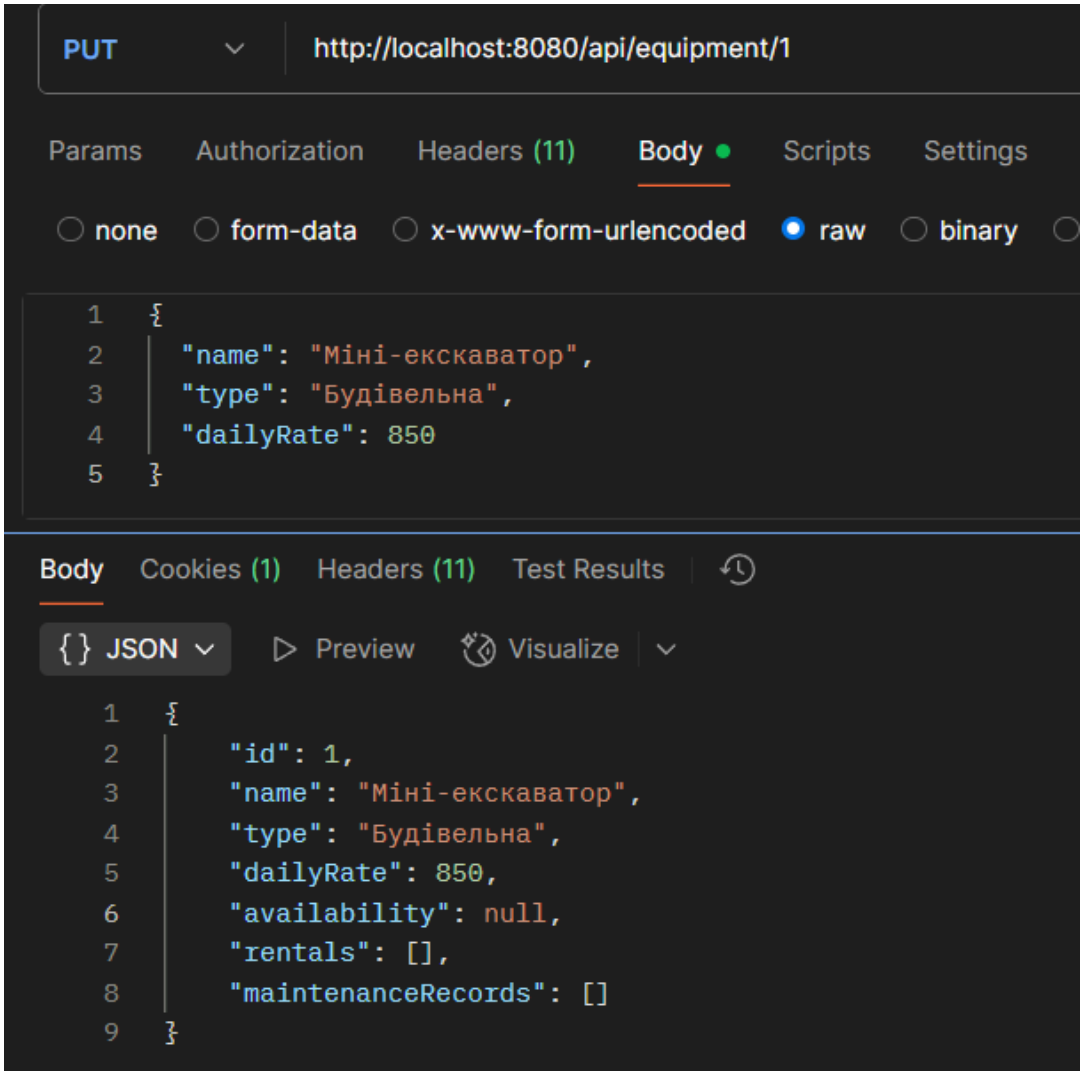

Рисунок 5 – Успішне оновлення інформації про обладнання

Оновлення обладнання за ідентифікатором

Метод: PUT

URL: http://localhost:8080/api/equipment/1

Очікування: Сервер оновлює дані про обладнання з ID = 1 відповідно до тіла запиту. У відповіді повертається оновлений JSON-об’єкт.

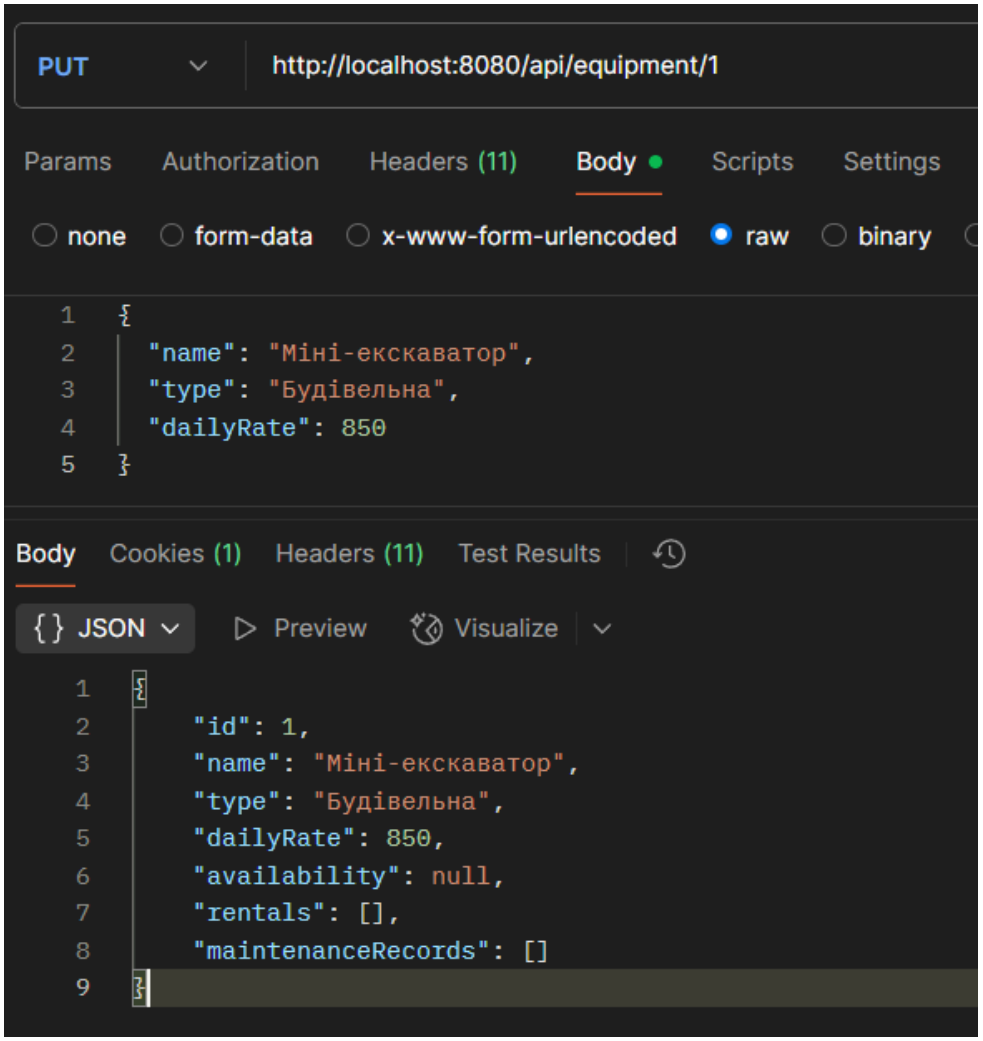

Рисунок 6 – Успішне оновлення даних обладнання за допомогою PUT-запиту

Видалення обладнання за ID

Метод: DELETE

URL: http://localhost:8080/api/equipment/1

Очікування: У відповідь очікується підтвердження успішного видалення обладнання з ID = 1. Запит не містить тіла (body) або може містити необов’язкові дані, залежно від реалізації API. Авторизаційні заголовки повинні бути додані для захищених маршрутів.

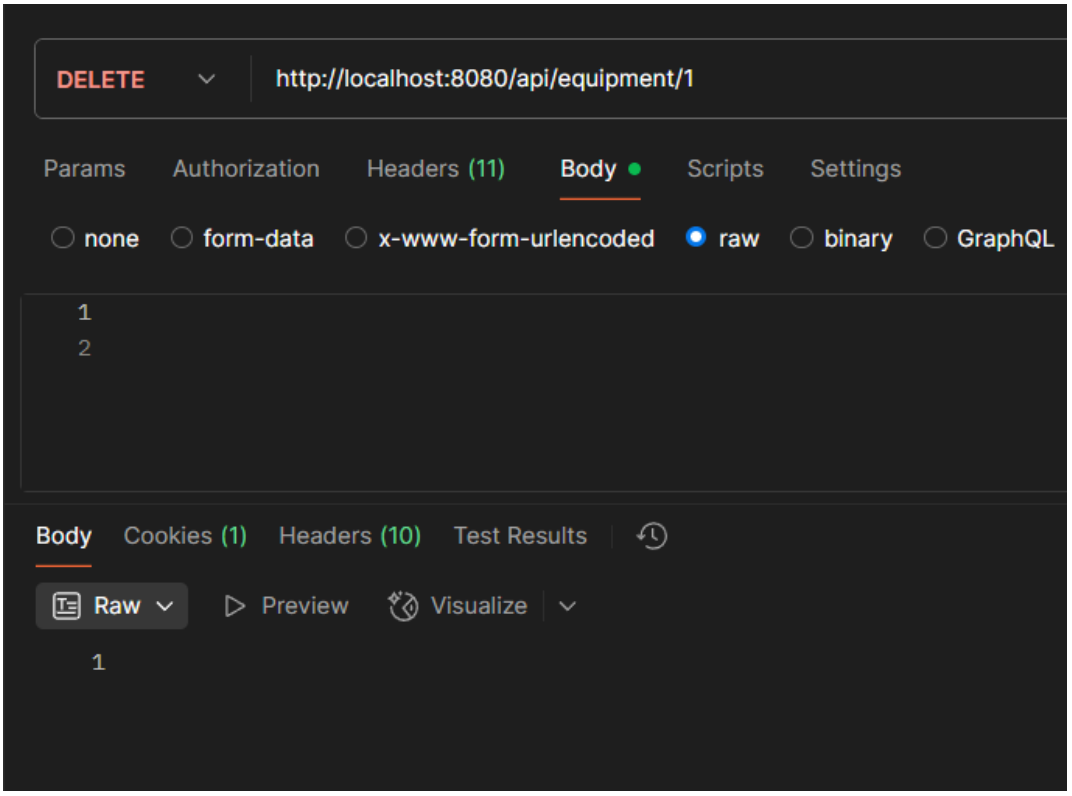

Рисунок 7 – Успішний запит на видалення обладнання

Створення нового клієнта

Метод: POST

URL: http://localhost:8080/api/customers

Очікування: У тілі запиту передаються дані нового клієнта у форматі JSON. У відповідь повертається створений клієнт з присвоєним унікальним ID.

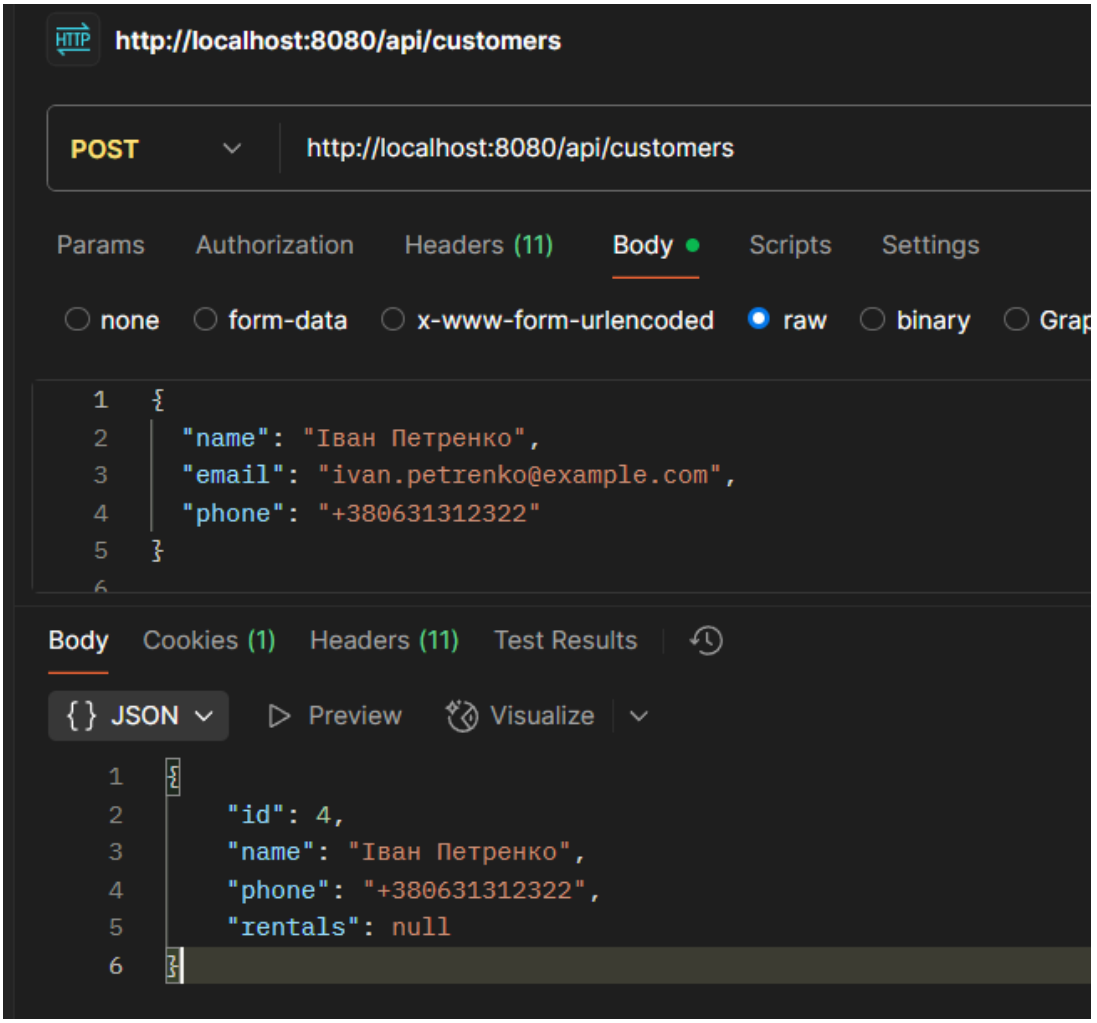

Рисунок 8 – Успішне створення нового клієнта

Отримання списку клієнтів

Метод: GET

URL: http://localhost:8080/api/customers

Очікування: У відповідь повертається список усіх зареєстрованих клієнтів у форматі JSON.

Кожен клієнт містить унікальний ідентифікатор (id), ім’я (name), номер телефону (phone) та список оренд (rentals).

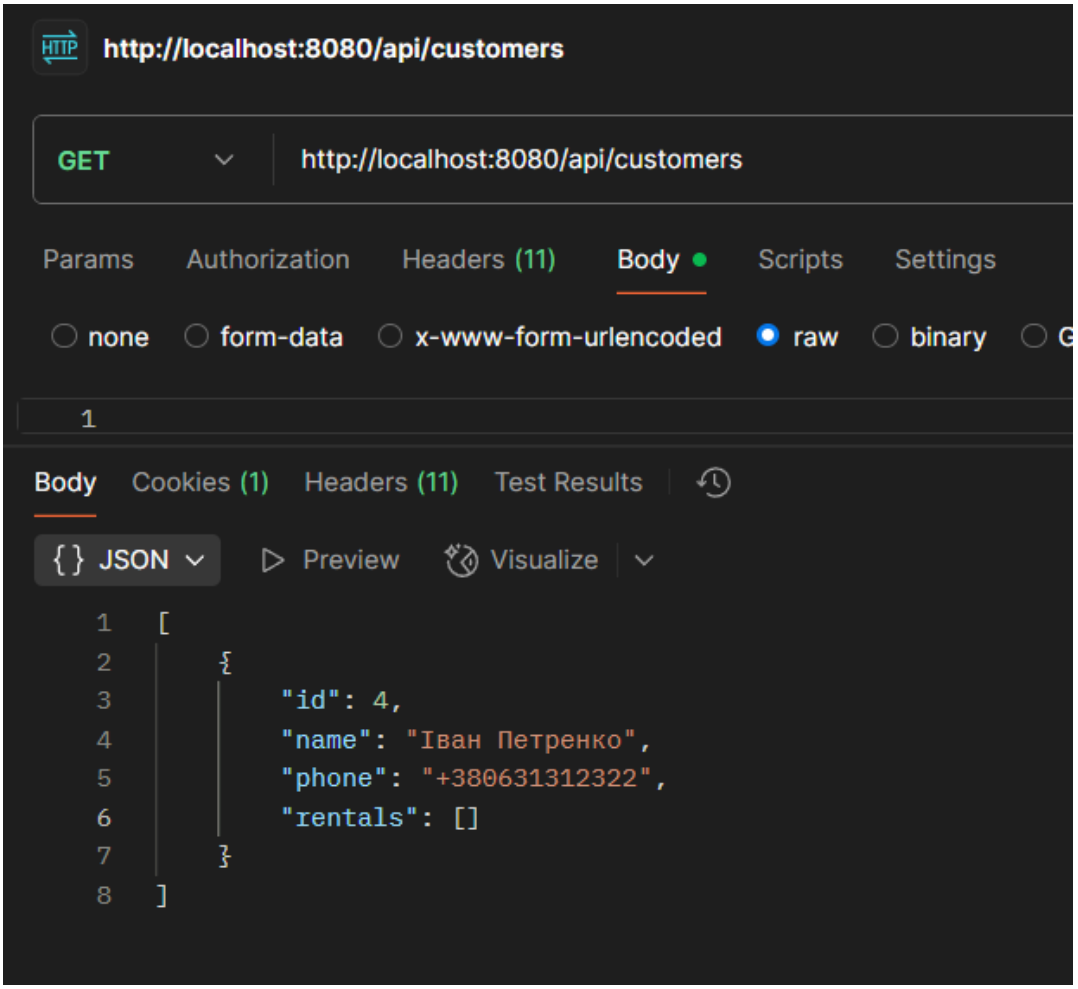

Рисунок 9 – Успішне отримання списку клієнтів

Оновлення інформації про клієнта

Метод: PUT

URL: http://localhost:8080/api/customers/4

Очікування: В тілі запиту передаються оновлені дані клієнта у форматі JSON. У відповідь сервер повертає оновлений об’єкт клієнта з підтвердженням змін.

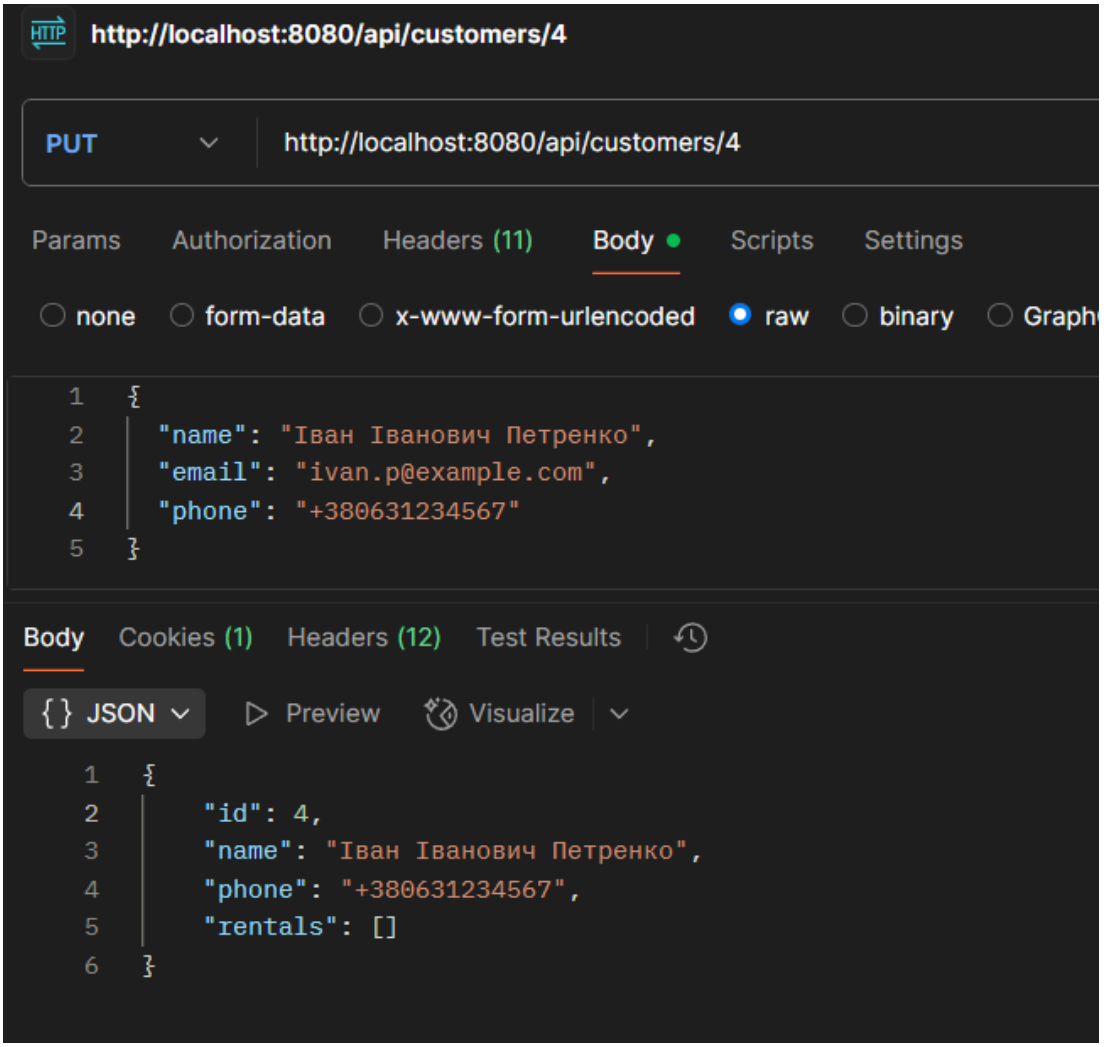

Рисунок 10 – Успішне оновлення даних клієнта з ID = 4

Видалення клієнта

Метод: DELETE

URL: http://localhost:8080/api/customers/4

Очікування: Видаляється клієнт з ідентифікатором 4. У відповіді може повертатися підтвердження успішного видалення або бути порожньою, що також вказує на успішну операцію.

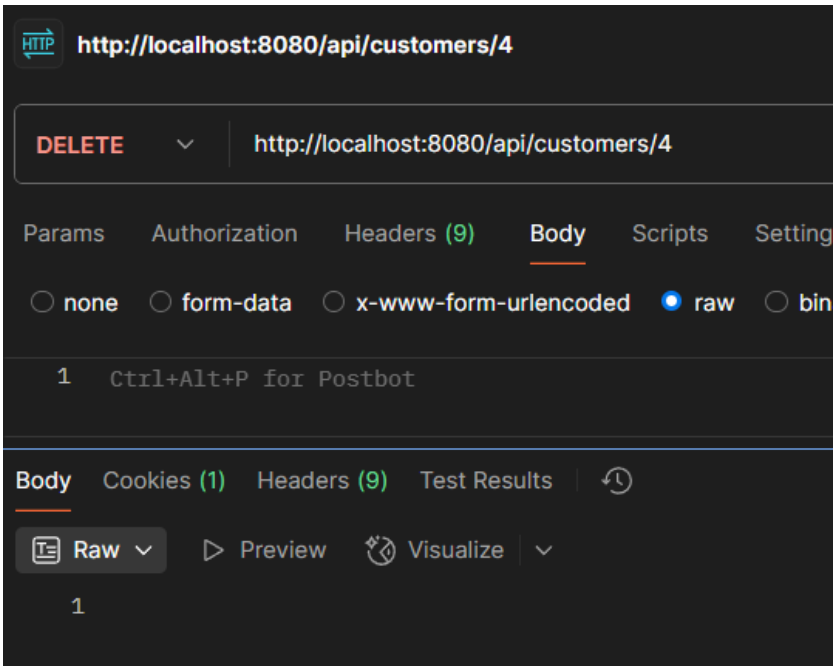

Рисунок 11 – Успішне видалення клієнта з ID = 4

Створення нового працівника

Метод: POST

URL: http://localhost:8080/api/employees

Очікування: У тілі запиту передаються дані нового працівника. У відповідь сервер повертає створений об’єкт з присвоєним унікальним ідентифікатором.

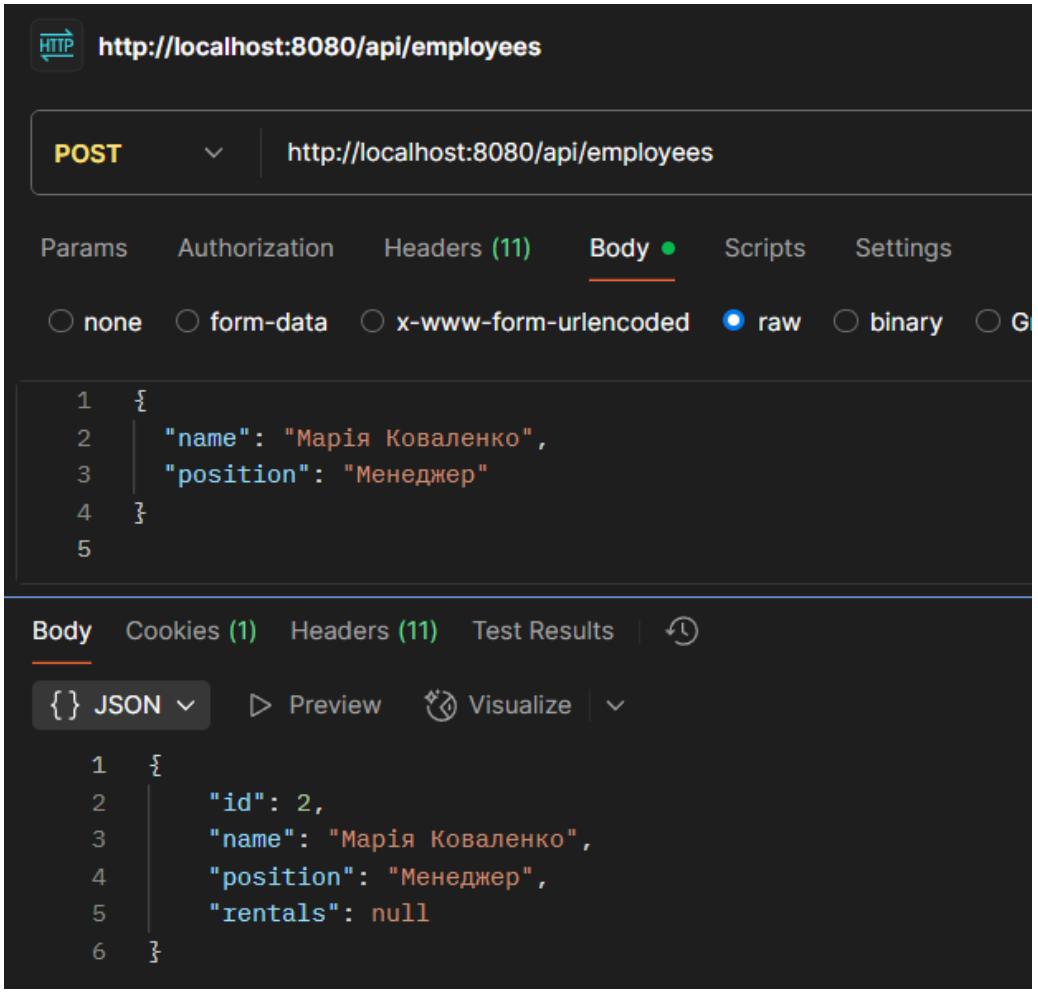

Рисунок 12 – Успішне створення працівника з ім’ям "Марія Коваленко"

Отримання списку працівників

Метод: GET

URL: http://localhost:8080/api/employees

Очікування: У відповідь повертається масив об’єктів усіх наявних працівників у системі.

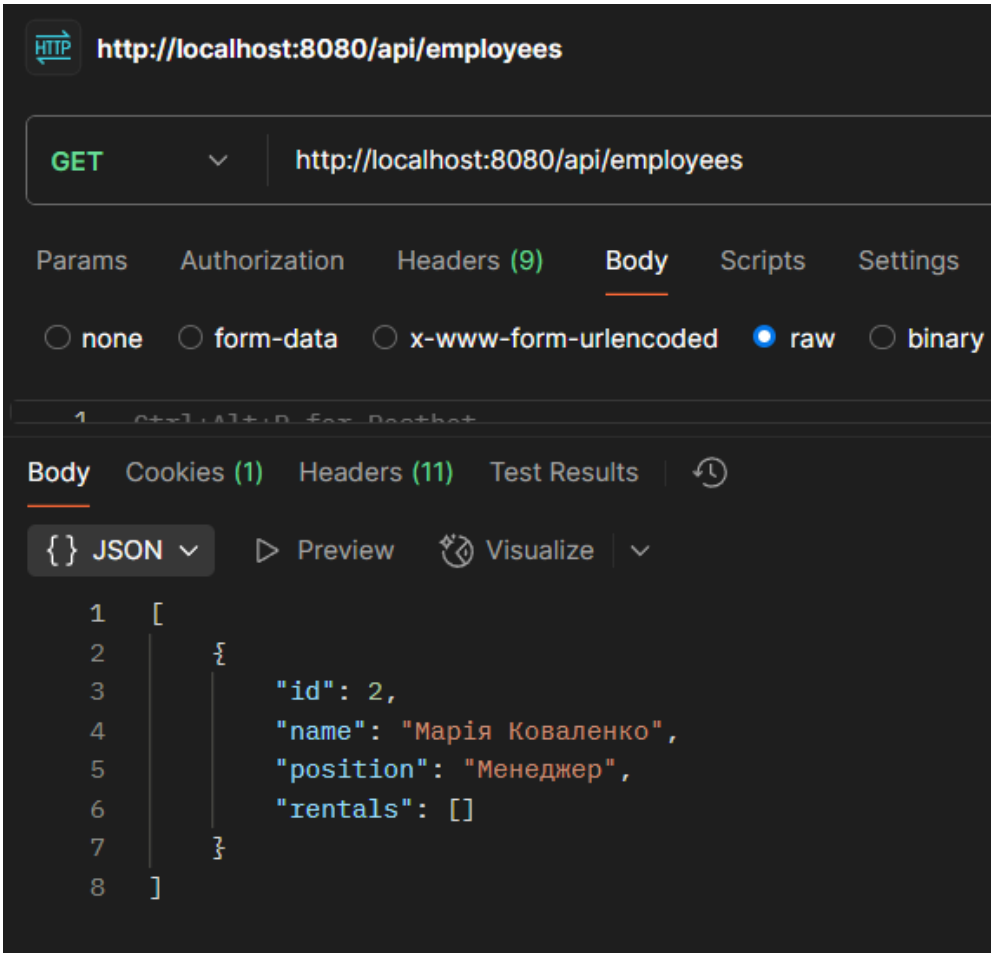

Рисунок 13 – Успішне отримання списку працівників

Оновлення інформації про працівника

Метод: PUT

URL: http://localhost:8080/api/employees/2

Очікування: В тілі запиту передаються оновлені дані працівника з ідентифікатором 2. У відповідь сервер повертає оновлений об’єкт працівника з підтвердженням змін.

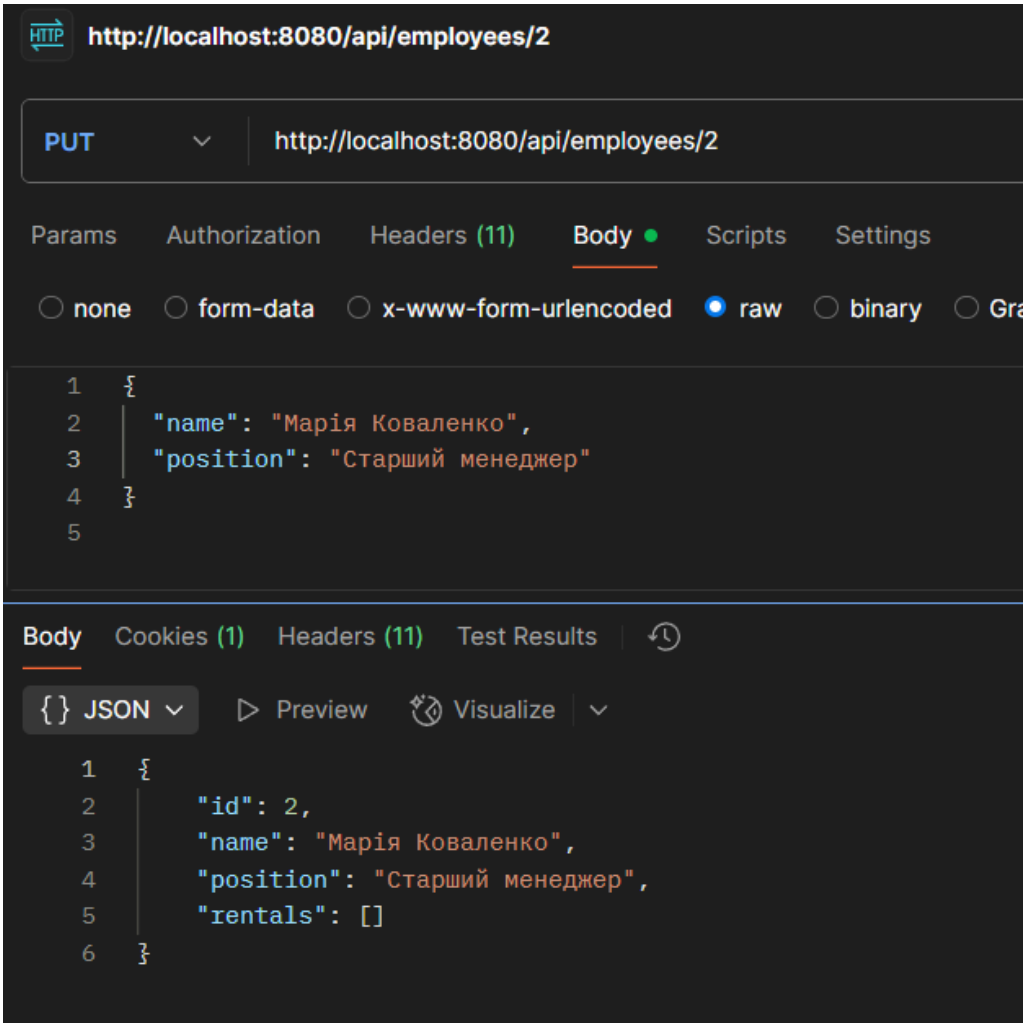

Рисунок 8 – Успішне оновлення даних працівника з ID = 2

Видалення працівника

Метод: DELETE

URL: http://localhost:8080/api/employees/2

Очікування: Видаляється працівник з ідентифікатором 2. Успішне видалення повертає порожню відповідь або підтвердження.

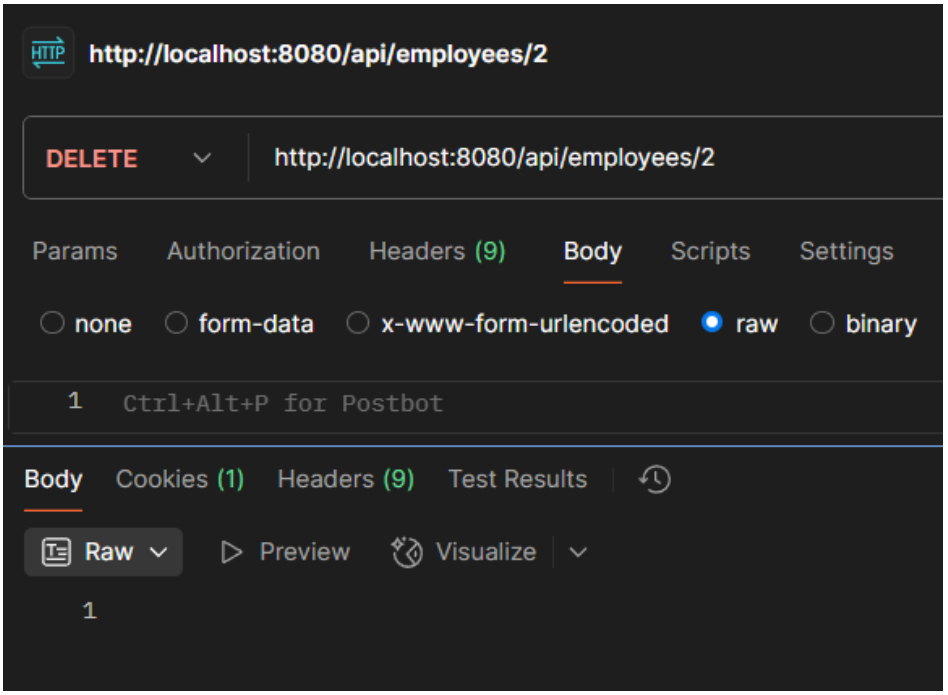

Рисунок 15 – Успішне видалення працівника з ID = 2

Створення оренди обладнання

Метод: POST

URL: http://localhost:8080/api/rentals

Тіло запиту:

```
{
  "equipmentId": 1,
  "customerId": 1,
  "employeeId": 1,
  "startDate": "2024-06-01",
  "endDate": "2024-06-10",
  "returned": false
}
```

Тіло відповіді:

```
{
  "id": 1,
  "equipmentId": 1,
```

```
  "equipmentName": "Екскаватор",
  "customerId": 1,
  "customerName": "Іван Петренко",
  "employeeId": 1,
  "employeeName": "Марія Коваленко",
  "startDate": "2024-06-01",
  "endDate": "2024-06-10",
  "returned": false
}
```


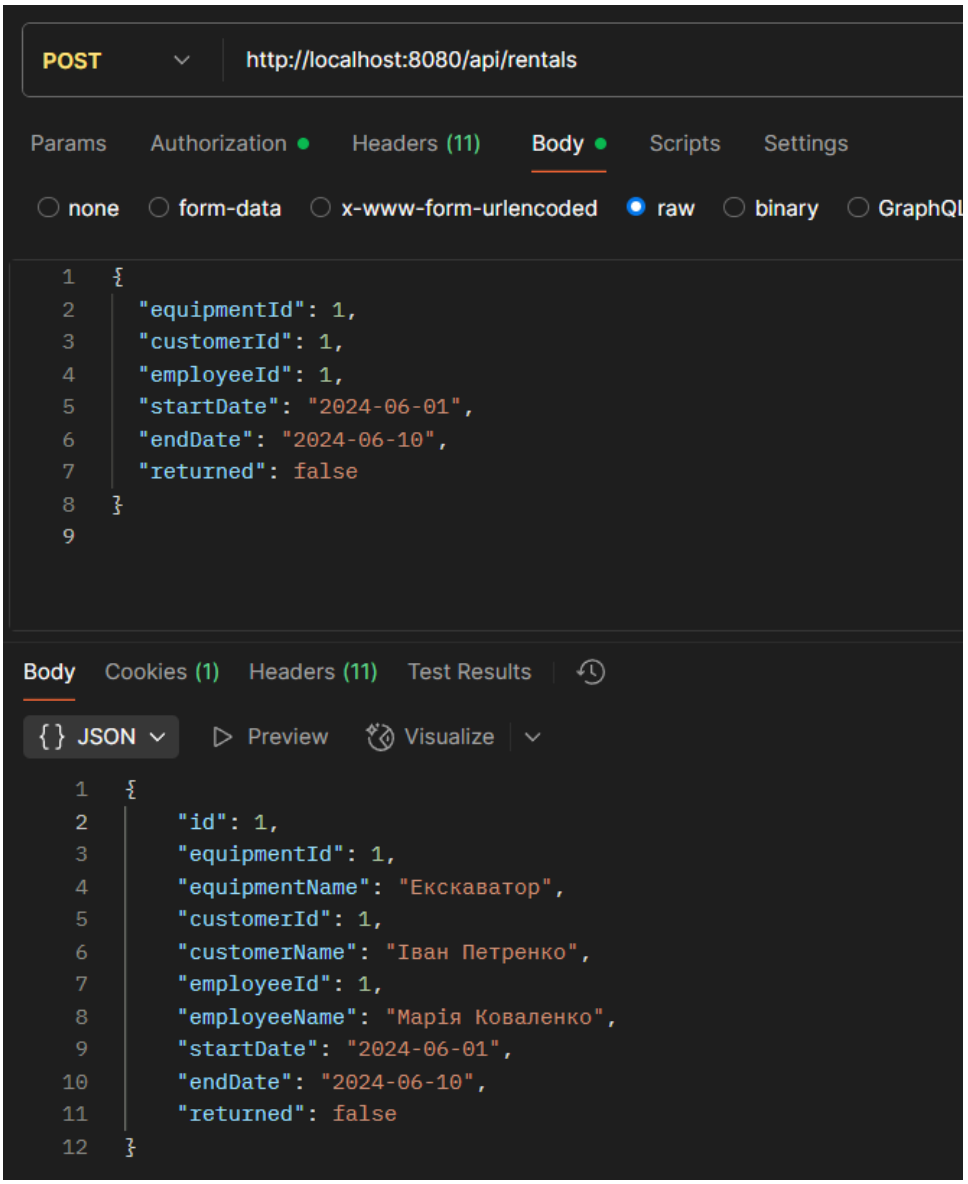

Рисунок 16 – Успішне створення нового запису про оренду обладнання

Отримання оренд за ідентифікатором клієнта
Метод: GET
URL: http://localhost:8080/api/rentals/customer/1

Тіло відповіді:


```
[
  {
    "id": 1,
    "equipmentId": 1,
    "equipmentName": "Екскаватор",
    "customerId": 1,
    "customerName": "Іван Петренко",
    "employeeId": 1,
    "employeeName": "Марія Коваленко",
    "startDate": "2024-06-01",
    "endDate": "2024-06-10",
    "returned": false
  }
]
```


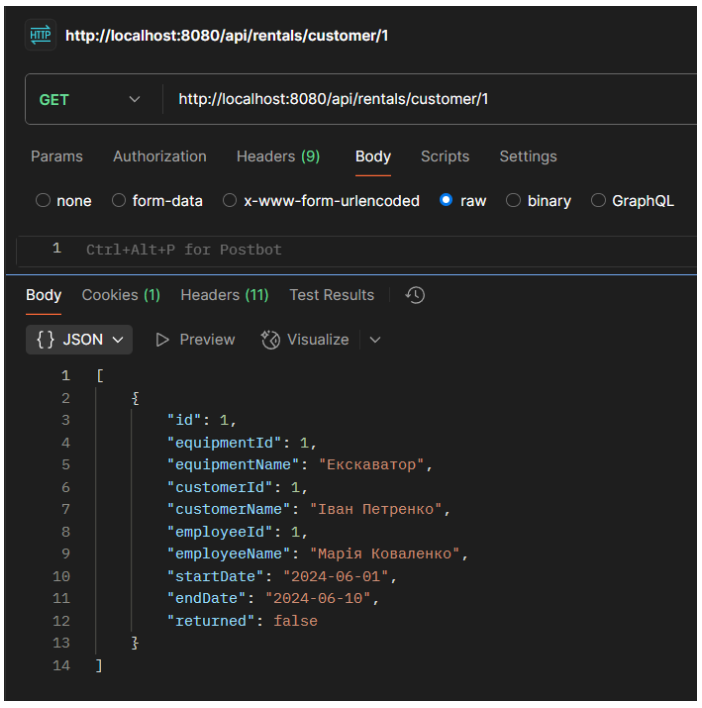

Рисунок 17 – Отримання списку оренд, оформлених клієнтом з ID = 1

Отримання оренд за ідентифікатором працівника

Метод: GET

URL: `http://localhost:8080/api/rentals/employee/1`

Тіло відповіді:


```
[
  {
    "id": 1,
    "equipmentId": 1,
    "equipmentName": "Екскаватор",
    "customerId": 1,
    "customerName": "Іван Петренко",
    "employeeId": 1,
    "employeeName": "Марія Коваленко",
    "startDate": "2024-06-01",
    "endDate": "2024-06-10",
    "returned": false
  }
]
```

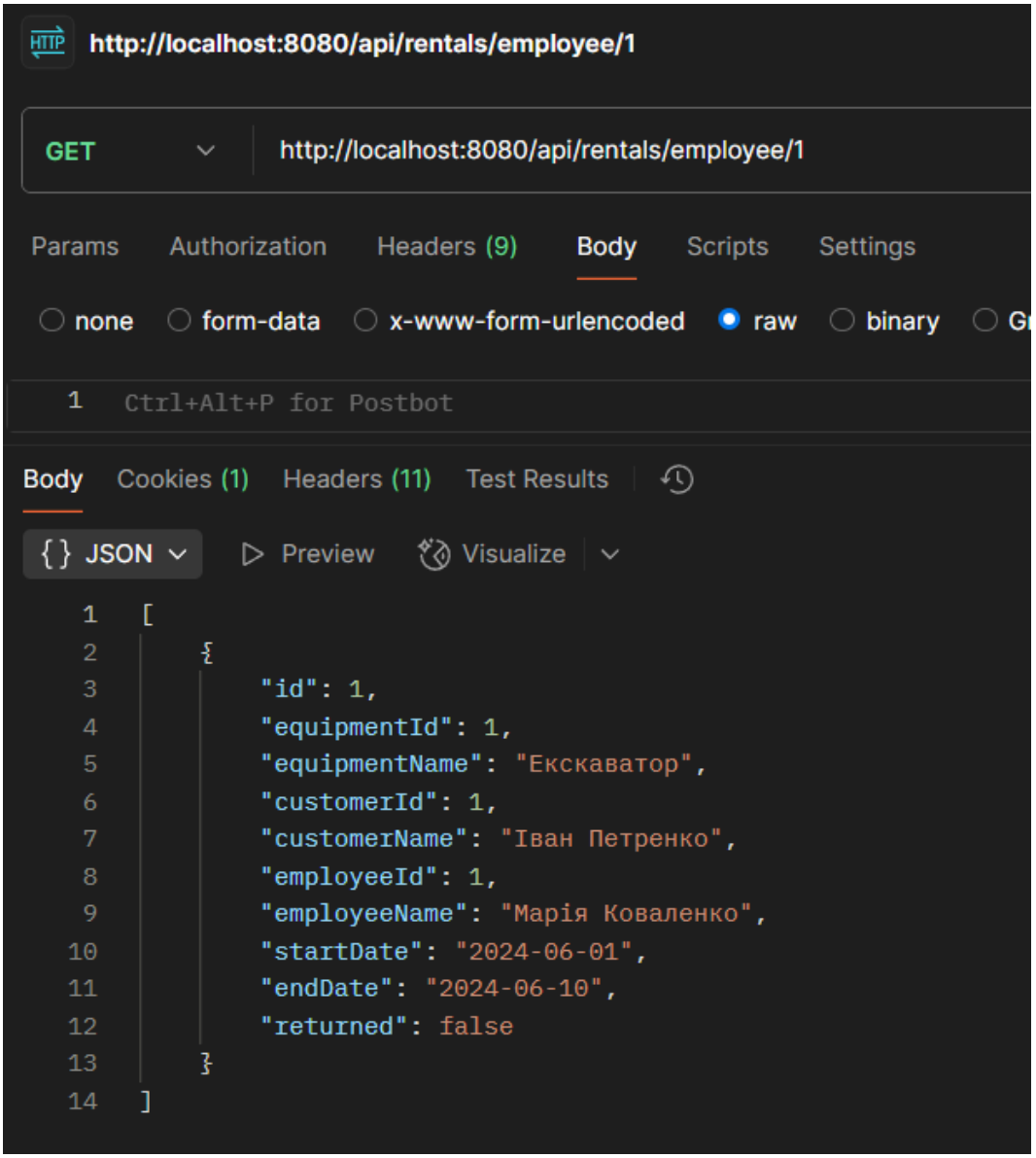

Рисунок 18 – Отримання списку оренд, оформлених працівником з ID = 1

Завершення оренди

Метод: PUT

URL: http://localhost:8080/api/rentals/1/complete

Тіло відповіді:


```
{
  "id": 1,
  "equipmentId": 1,
  "equipmentName": "Екскаватор",
  "customerId": 1,
```

```
  "customerName": "Іван Петренко",
  "employeeId": 1,
  "employeeName": "Марія Коваленко",
  "startDate": "2024-06-01",
  "endDate": "2024-06-10",
  "returned": true
}
```


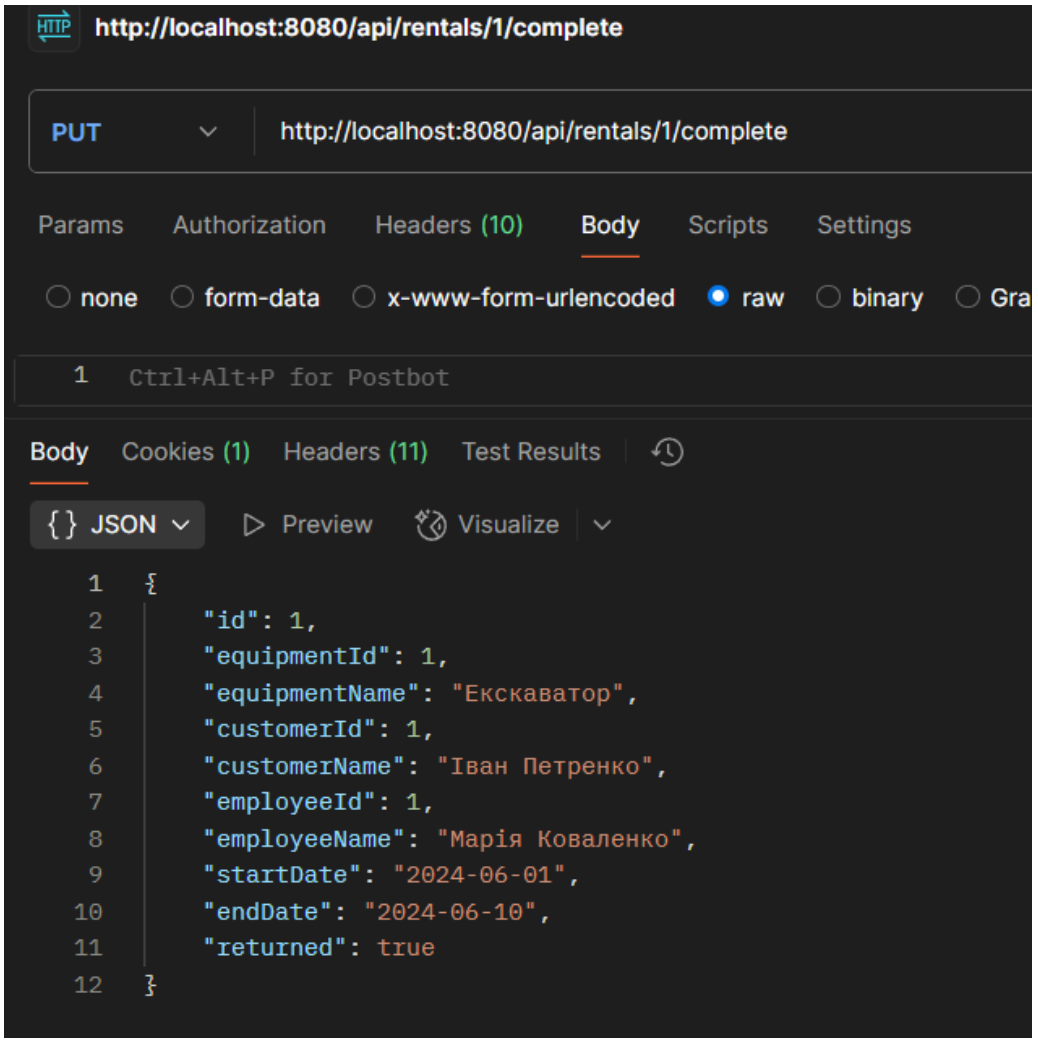

Рисунок 19 – Завершення оренди з ID = 1 (поле returned змінено на true)

Додавання запису про технічне обслуговування

Метод: POST

URL: http://localhost:8080/api/maintenance

Тіло запиту:

```
{
  "equipmentId": 1,
  "date": "2024-06-05",
  "description": "Планове ТО двигуна"
}
```


Відповідь:


```
{
  "id": 1,
  "equipmentId": 1,
  "date": "2024-06-05",
  "description": "Планове ТО двигуна"
}
```


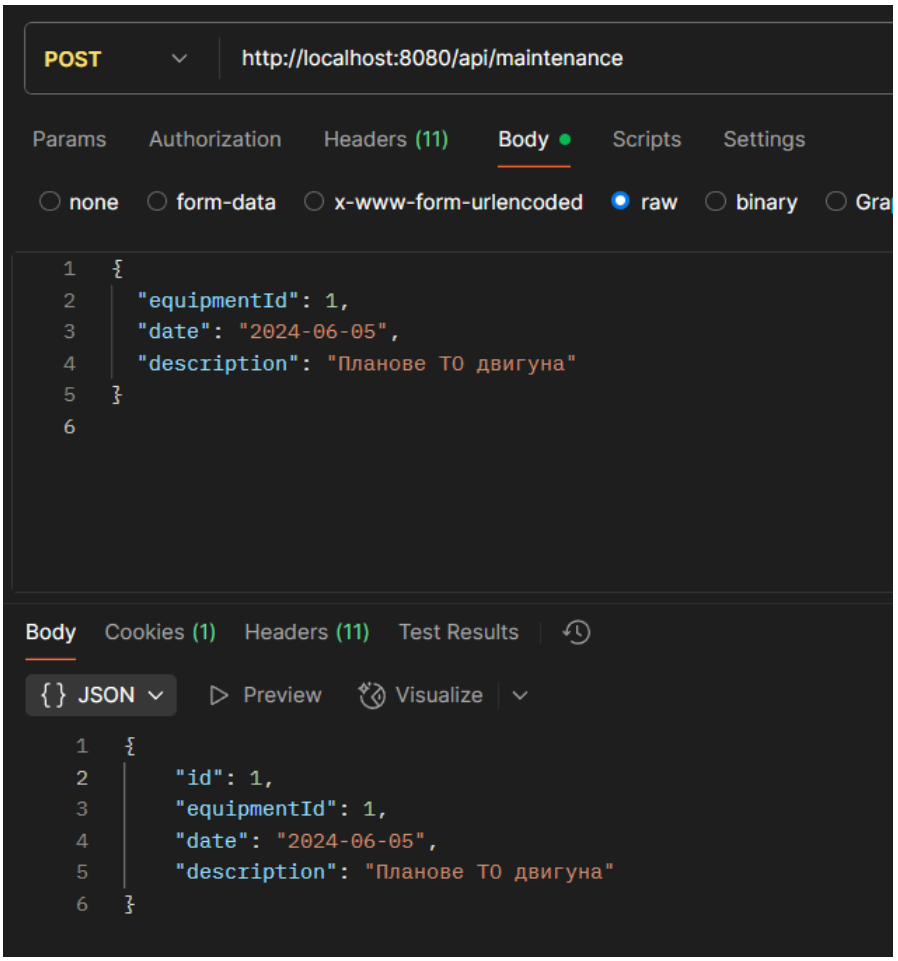

Рисунок 20 – Додавання інформації про планове технічне обслуговування для обладнання з ID = 1

Отримання інформації про технічне обслуговування обладнання
Метод: GET
URL: http://localhost:8080/api/maintenance/1

Результат:


```
[
  {
    "id": 1,
    "equipmentId": 1,
    "date": "2024-06-05",
    "description": "Планове ТО двигуна"
  }
]
```


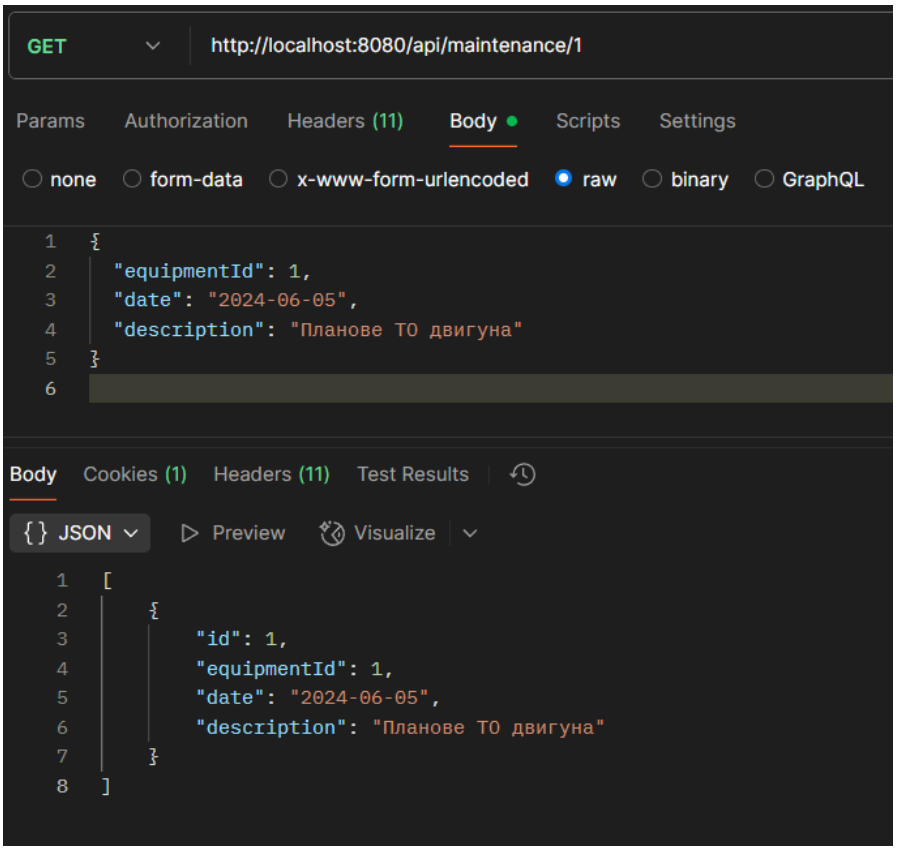

Рисунок 21 – Отримання інформації про обслуговування обладнання з ID = 1

Отримання списку активних оренд
Метод: GET
URL: http://localhost:8080/api/rentals/active

Результат:


```
[
  {
    "id": 2,
    "equipmentId": 1,
    "equipmentName": "Екскаватор",
    "customerId": 1,
    "customerName": "Іван Петренко",
    "employeeId": 1,
    "employeeName": "Марія Коваленко",
    "startDate": "2025-06-01",
    "endDate": "2025-06-30",
    "returned": false
  }
]
```


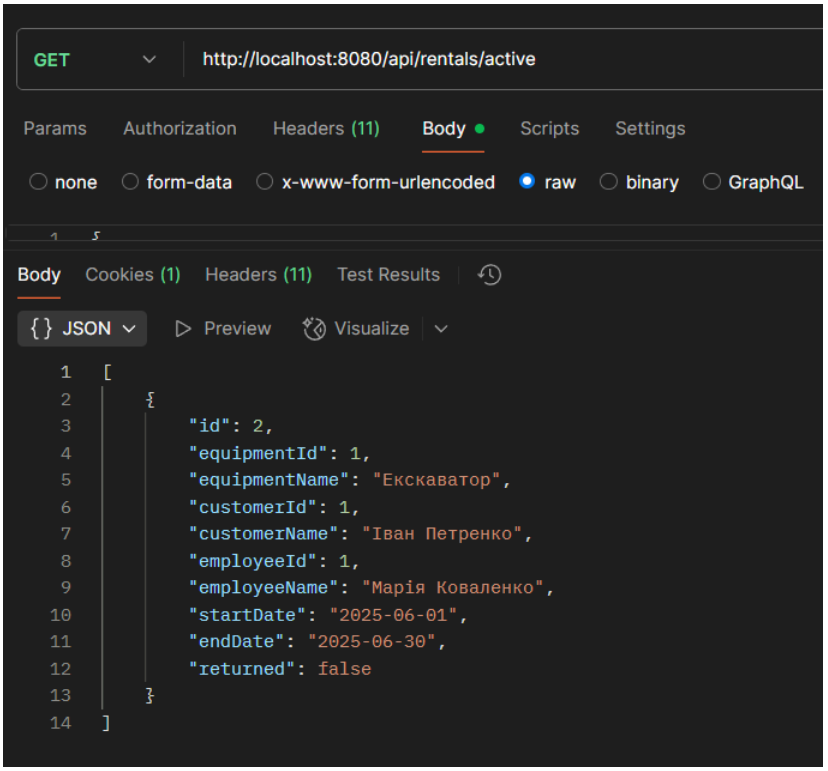

Рисунок 22 – Отримання активних записів оренди

Доступність обладнання на конкретну дату

Метод: GET

URL: http://localhost:8080/api/equipment/available?date=2024-06-15

Очікування: У відповідь повертається список обладнання, яке є доступним на вказану дату. Якщо значення поля "availability" — null, це може означати, що доступність не визначена або обробляється іншим способом.

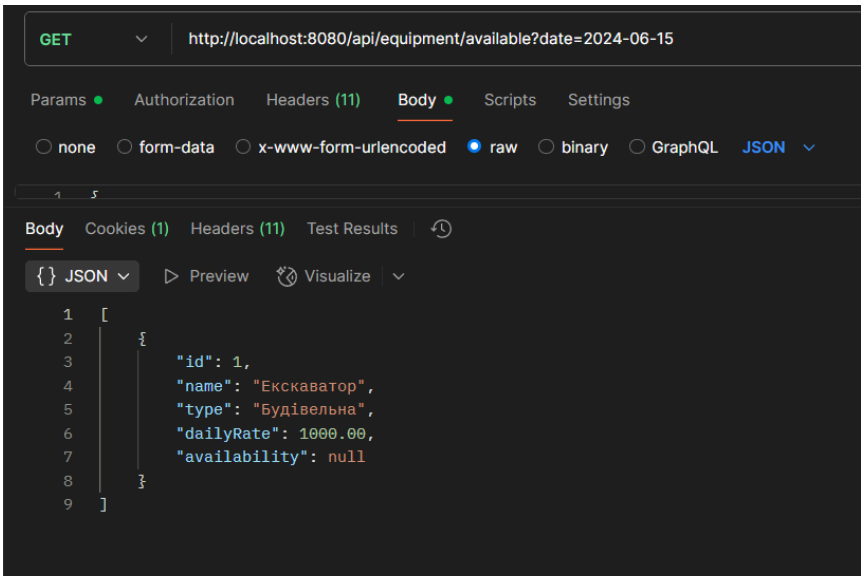

Рисунок 23 – Отримання доступного обладнання на дату 2024-06-15

Кількість оренд за кожним обладнанням

Метод: GET

URL: http://localhost:8080/api/rentals/count-by-equipment

Очікування: У відповідь повертається масив, де кожен елемент відповідає кількості оренд для конкретного обладнання. Значення у форматі масиву чисел.

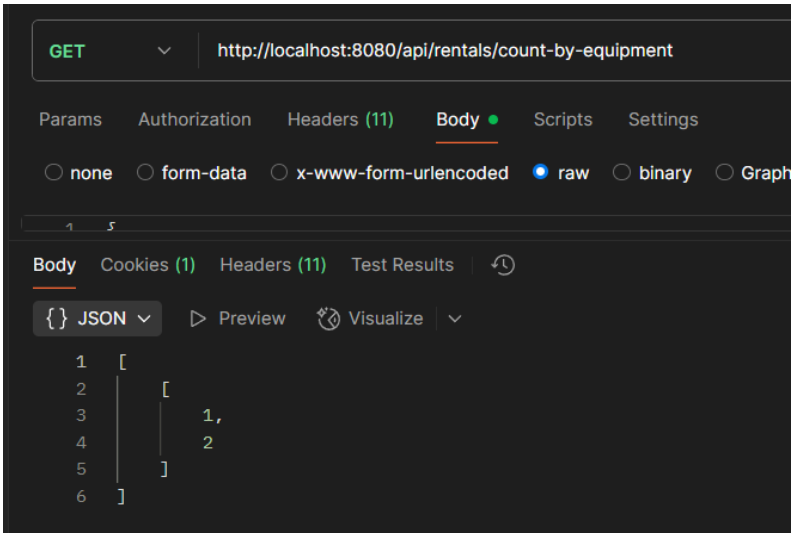

Рисунок 24 – Отримання кількості оренд по кожному обладнанню

Загальний дохід від оренди

Метод: GET

URL: http://localhost:8080/api/rentals/total-income

Очікування: У відповідь повертається загальна сума доходу, отриманого від усіх оренд, у вигляді десяткового числа.

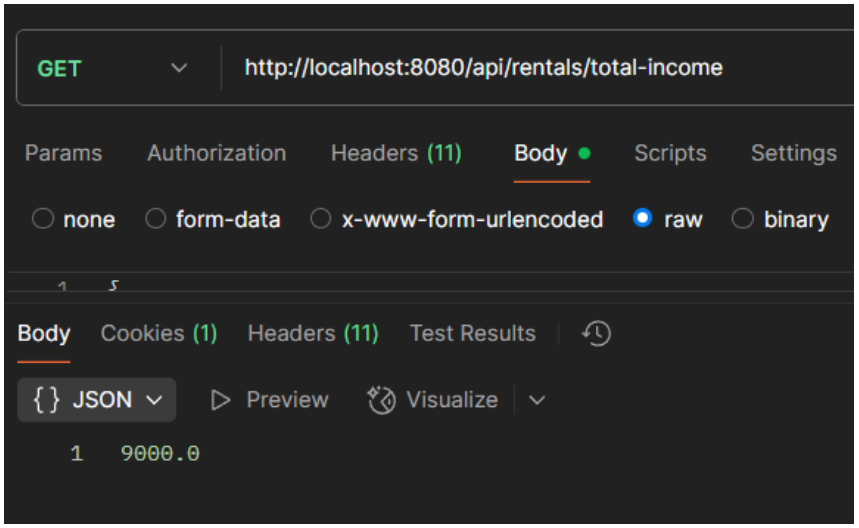

Рисунок 25 – Отримання загального доходу від оренди

Найпопулярніше обладнання (найчастіше орендоване)

Метод: GET

URL: http://localhost:8080/api/rentals/most-rented-equipment

Очікування: У відповідь повертається інформація про обладнання, яке було орендовано найчастіше. Вказуються основні характеристики: ID, назва, тип, добова вартість та статус доступності.

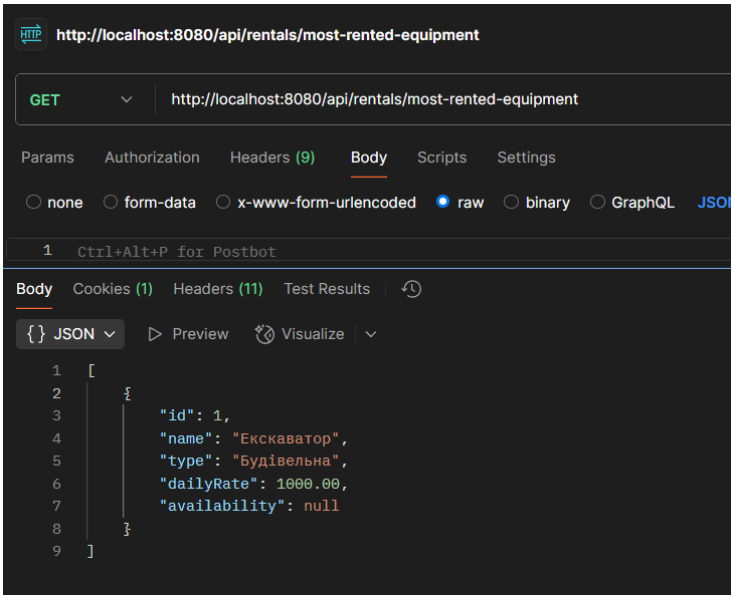

Рисунок 26 – Отримання найчастіше орендованого обладнання

Прострочені оренди обладнання

Метод: GET

URL: http://localhost:8080/api/rentals/overdue

Очікування: У відповідь повертається список оренд, термін яких завершився, але обладнання ще не повернуто. Кожна оренда містить інформацію про обладнання, клієнта, працівника, дати оренди та статус повернення.

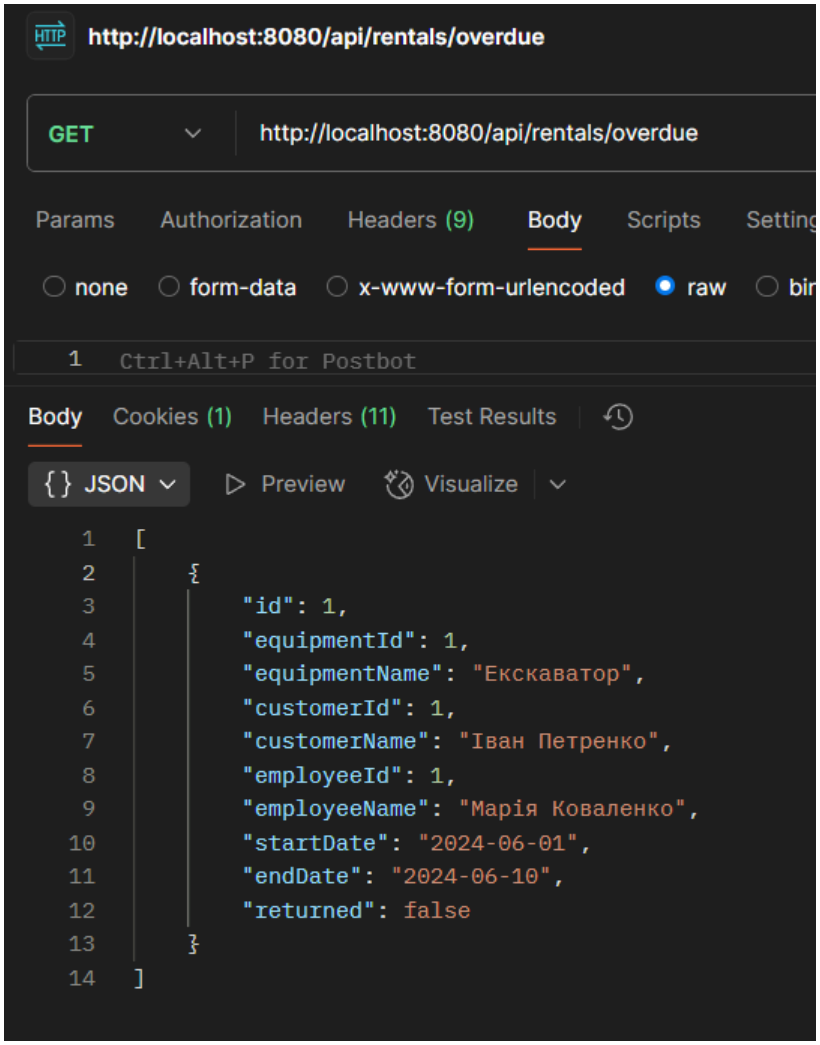

Рисунок 27 – Перегляд прострочених оренд

Видалення запису оренди

Метод: DELETE

URL: http://localhost:8080/api/rentals/1

Очікування: Видаляє запис оренди з вказаним id (у цьому випадку 1). Якщо видалення успішне, сервер повертає підтвердження (наприклад, статус 200 або 204 без тіла відповіді).

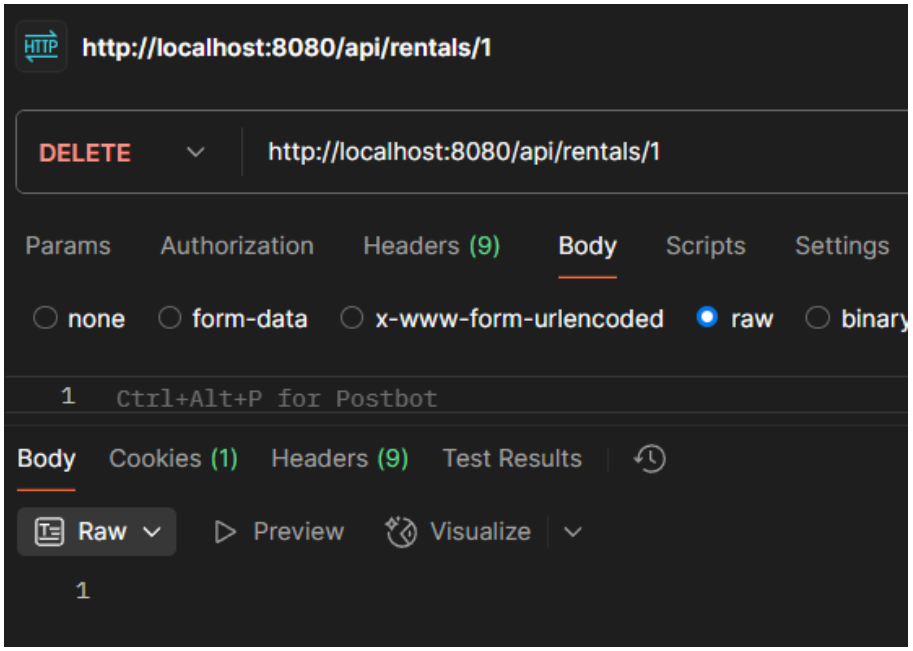

Рисунок 28 – Видалення запису оренди з бази даних

Висновок

У результаті виконання курсової роботи було розроблено повноцінну інформаційну систему для автоматизації процесів управління орендою техніки. Реалізований програмний продукт дозволяє ефективно керувати обладнанням, клієнтами, співробітниками, орендними угодами та записами технічного обслуговування.

Система підтримує базовий і розширений функціонал: створення, перегляд, оновлення та видалення сутностей, ведення історії оренд, облік повернень, генерацію статистики та звітності, включаючи загальний дохід і найпопулярніше обладнання. Передбачено також перевірку доступності техніки, облік прострочених оренд та механізми захисту доступу через авторизацію і автентифікацію з використанням JWT.

У ході роботи було реалізовано трирівневу архітектуру системи (контролери, сервіси, репозиторії) із використанням Spring Boot та Spring Data JPA. Забезпечено обробку помилок, валідацію введених даних, використання DTO для безпечного обміну інформацією між клієнтом і сервером.

Функціональність була протестована за допомогою інструмента Postman, що підтвердило стабільність і коректність роботи всіх основних API-ендпоінтів.

Таким чином, поставлені у вступі цілі були досягнуті: створено надійну, масштабовану та зручну систему, що здатна покращити якість обслуговування клієнтів, мінімізувати людський фактор і підвищити ефективність ведення бізнесу у сфері оренди техніки. Отримані знання з об'єктно-орієнтованого програмування, роботи з REST API, базами даних та безпекою дозволили поглибити професійні навички та закріпити теоретичні знання на практиці.

Список використаних джерел

1. DigitalOcean. URL: https://www.digitalocean.com/community/tutorials (дата звернення: 13.06.2025)

2. GeeksforGeeks. URL: https://www.geeksforgeeks.org/spring-boot/ (дата звернення: 13.06.2025)

3. Java Brains. URL: https://www.javabrains.io/ (дата звернення: 13.06.2025)

4. FreeCodeCamp. URL: https://www.freecodecamp.org/news/tag/spring-boot/ (дата звернення: 13.06.2025)

5. Okta Developer Blog. URL: https://developer.okta.com/blog/ (дата звернення: 13.06.2025)

6. YouTube – Amigoscode. URL: https://www.youtube.com/@amigoscode (дата звернення: 13.06.2025)

7. Spring Academy. URL: https://academy.spring.io/ (дата звернення: 13.06.2025)

8. JetBrains Blog. URL: https://blog.jetbrains.com/ (дата звернення: 13.06.2025)

9. CodeGym. URL: https://codegym.cc/ (дата звернення: 13.06.2025)

10. Javatpoint. URL: https://www.javatpoint.com/spring-boot-tutorial (дата звернення: 13.06.2025)