ЛАБОРАТОРНА РОБОТА № 6

Архітектурне моделювання. Елементи графічної нотації діаграми компоненті

Мета: Навчитися моделювати програмні системи за допомогою компонентних діаграм UML, розуміти структуру компонентів, їх взаємодію та залежності.
Час: 4 години.
Література:

1. Теоретичні відомості

Компонентна діаграма – теорія

Компонентна діаграма — це тип **UML-діаграми (Unified Modeling Language)**, який використовується для **візуалізації високорівневої структури системи**. Вона показує, як **програмні компоненти** взаємодіють та залежать один від одного.

- **Призначення**: Представити організацію та залежності компонентів у системі.
- **Компоненти**: Модульні частини системи (сервіси, модулі, бази даних).
- **З'єднання**: Залежності або взаємодія між компонентами.
- **Сфери використання**:
    - Архітектури мікросервісів та моноліти
    - Веб- та мобільні додатки
    - Інтеграція програмних систем

1. **Компонент (Component)**
    - Основний модуль системи, який виконує певну функціональність.
    - Може бути **апаратним, програмним або логічним** елементом.
    - На діаграмі позначається **прямокутником із заголовком** або прямокутником з маленьким значком компонента.
    - **Приклад**: `[User Service]`, `[Payment Module]`.
2. **Інтерфейс (Interface)**
    - Визначає набір операцій, які компонент надає (provided interface) або потребує (required interface).
    - Позначається кружком ("lollipop" для provided) або півкружком ("socket" для required).
    - **Приклад**: компонент `[Payment Service]` надає інтерфейс `IPaymentGateway`.
3. **Пакет (Package)**
    - Використовується для **групування пов'язаних компонентів**.
    - Допомагає структурувати великі системи і робить діаграму читабельнішою.
    - **Приклад**: `package "Microservices" { [Auth Service] [Order Service] }`.
4. **Залежність (Dependency)**
    - Вказує, що один компонент **потребує або використовує функціональність іншого**.
    - Позначається **пунктирною стрілкою**: `-->`.
    - **Приклад**: `[Order Service] --> [Payment Service]`.
5. **Комунікаційні зв'язки (Communication / Association)**
    - Вказує на **тесну взаємодію між компонентами**, частіше використовуються в деталізованих діаграмах.

    - Позначаються **суцільною лінією або стрілкою**.
6. **Артефакти (Artifacts)**
    - Реальні файли або ресурси, які компонент використовує або створює.
    - Наприклад: `.jar` файл, конфігураційний файл, база даних.
7. **Сервіси зовнішніх систем (External Services)**
    - Компоненти або сервіси, що не належать до системи, але використовуються нею.
    - Може бути зовнішній API, платіжний шлюз, служба повідомлень.

Основні елементи компонентної діаграми Plant UML

Компоненти: Auth, User, Order, Payment Service
Інтерфейси: IAuth, IPaymentGateway
Пакети: Клієнти, Мікросервіси, Бази даних/Артефакти, Зовнішні сервіси
Залежності: Стрілки показують потік викликів або використання
Артефакти: Наприклад, конфігураційний файл
Зовнішні сервіси: Payment Provider, Email Service

```
@startuml
skinparam componentStyle rectangle

' ========================
' Клієнти
' ========================
package "Клієнти" {
   [Web Browser]
   [Mobile App]
}

' ========================
' API Gateway
' ========================
[API Gateway]

' ========================
' Мікросервіси / Компоненти
' ========================
package "Мікросервіси" {
   [Auth Service]
   [User Service]
   [Order Service]
   [Payment Service]

   ' Інтерфейси
   interface "IAuth" as IAuth
   interface "IPaymentGateway" as IPayment
}

' ========================
' Бази даних / Артефакти
' ========================
package "Бази даних / Артефакти" {
   [Auth DB]
   [User DB]
   [Order DB]
```

```
    [Payment DB]
    artifact "Config File"
}

' ========================
' Зовнішні сервіси
' ========================
package "Зовнішні сервіси" {
    [Payment Provider]
    [Email Service]
}

' ========================
' Зв'язки
' ========================

' Клієнти -> API Gateway
[Web Browser] --> [API Gateway]
[Mobile App] --> [API Gateway]

' API Gateway -> Мікросервіси
[API Gateway] --> [Auth Service]
[API Gateway] --> [User Service]
[API Gateway] --> [Order Service]
[API Gateway] --> [Payment Service]

' Компоненти -> Інтерфейс
```

2. Завданн на лабораторну роботу

Завдання 1: Компонентна діаграма існуючої системи "Ordering"

1. Визначте основні компоненти системи:
    - Інтерфейс користувача (UI) – веб або мобільний додаток для оформлення замовлень.
    - Бізнес-логіка / менеджери – обробка замовлень, перевірка наявності товарів, керування статусом замовлень.
    - Модулі зберігання даних – база даних або репозиторії для зберігання інформації про замовлення, товари, клієнтів.
    - Зовнішні сервіси або пристрої – платіжні сервіси, служби доставки (якщо застосовується).

2. Опишіть взаємодію між компонентами та напрямок залежностей:
    - UI надсилає замовлення → Order Manager
    - Order Manager перевіряє Product Catalog
    - Order Manager взаємодіє з Payment Service для оплати
    - Після оплати Order Manager передає інформацію Delivery Service
    - Всі дані зберігаються у Database
3. Побудуйте компонентну діаграму UML:
    - Використовуйте будь-який UML-редактор (PlantUML, StarUML, Visio, Lucidchart).
    - Вкажіть компоненти, пакети та стрілки залежності.
4. Поясніть призначення компонентів та їх взаємодію:

- Наприклад, як UI працює з менеджерами, які дані зберігаються в базі, як викликаються зовнішні сервіси.

Завдання 2: Проектування компонентів для нової системи

1. Створіть проект програмної системи (наприклад, система керування приміщеннями або .NET Desktop Application).
2. Визначте компоненти для майбутньої системи:
   - User Interface (UI) – форми або вікна програми.
   - Room Manager / Sensor Manager / Device Manager – бізнес-логіка системи.
   - Repositories або Database – модулі для зберігання даних.
   - Зовнішні сервіси / пристрої – наприклад, сенсори, пристрої керування.
3. Визначте залежності між компонентами та напрямки взаємодії:
   - UI взаємодіє з менеджерами бізнес-логіки.
   - Менеджери працюють із репозиторіями та зовнішніми пристроями.
4. Побудуйте деталізовану компонентну діаграму UML для проектованої системи:
   - Використайте пакети компонентів для логічної групування.
   - Вкажіть інтерфейси для взаємодії між менеджерами та пристроями.
   - Показуйте взаємодію UI з бізнес-логікою та репозиторіями.
   - Поясніть, як компоненти взаємодіють у проекті:
   - Опишіть, як дані проходять через систему від UI до пристроїв і бази даних.

3. Методичні вказівки до виконання

Завдання 1. Проаналізувати приклади діаграм компонентів наведені в роботі

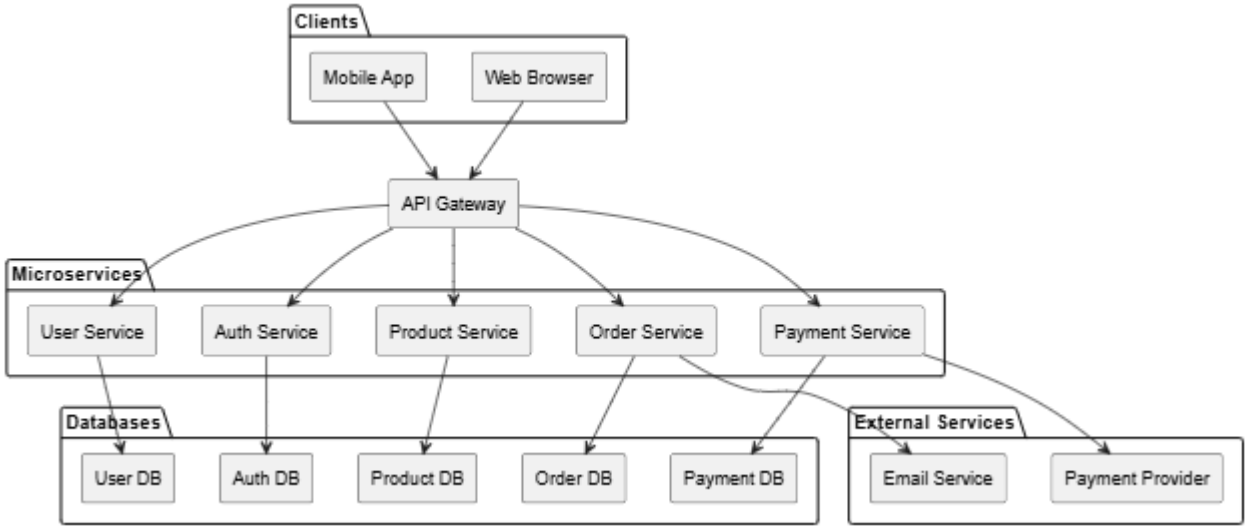


Рис.1. Узагальненя діаграма компонентів системи “Ordering” на основі мікросервісної архітектури

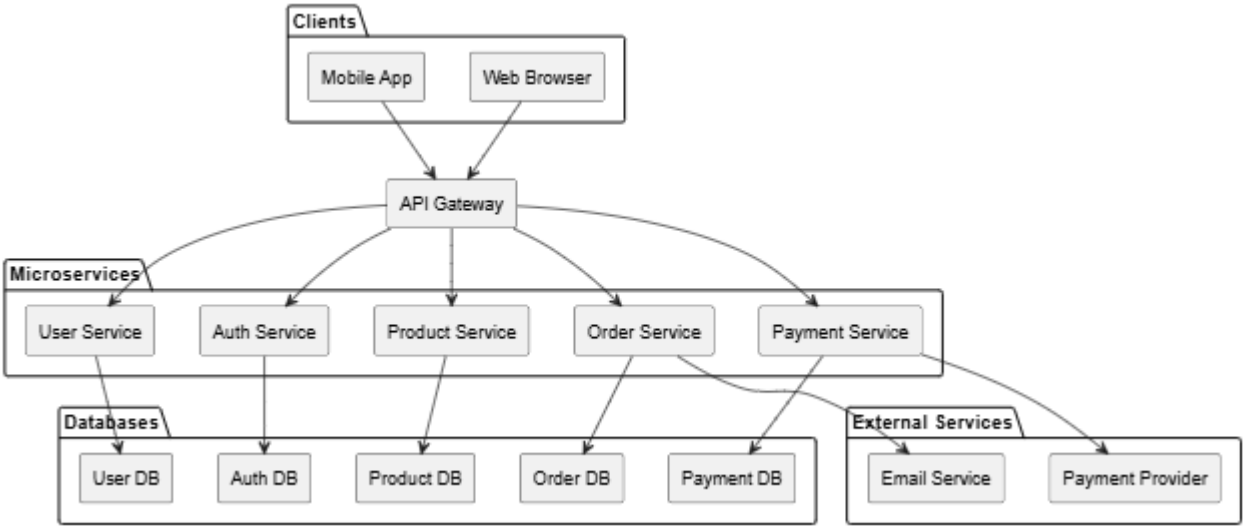


Рис.2. Узагальненя діаграма компонентів системи “Ordering” на основі монолітної архітектури

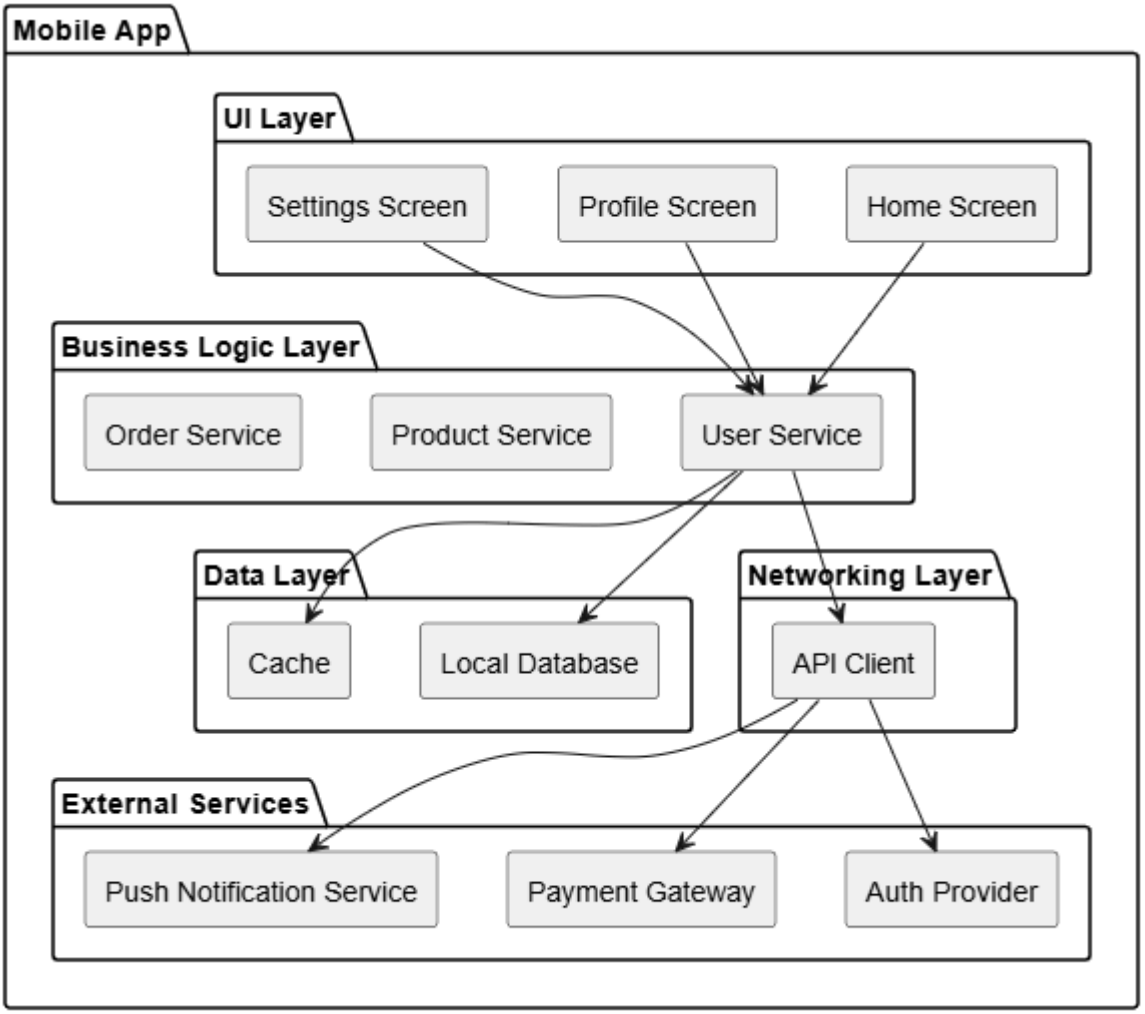


Рис3. Узагальненя діаграма компонентів системи “Ordering” мобільна версія

Завдання 2. Побудувати діаграму компоненетів системи для індивідуального завдання.

Розробити **компонентну діаграму** для системи керування приміщеннями.

Завдання:

1. **Визначте основні компоненти системи:**
    - `User Interface (UI)` – форми та елементи інтерфейсу користувача.
    - `Room Management` – компонент для керування приміщеннями та їх атрибутами.
    - `Sensor Manager` – компонент для роботи з датчиками (температура, вологість).
    - `Device Manager` – компонент для керування пристроями (обігрівач, кондиціонер).
    - `Database / Storage` – компонент для зберігання даних про приміщення, датчики та пристрої.
2. **Визначте взаємодію між компонентами:**
    - Вкажіть, які компоненти викликають інші компоненти та отримують від них дані.
    - Наприклад:
        - UI звертається до Room Management, Sensor Manager та Device Manager.
        - Менеджери взаємодіють з базою даних та фізичними пристроями.
3. **Позначте роль кожного компонента:**
    - Фронтенд (UI)
    - Бізнес-логіка (Room Management, Sensor Manager, Device Manager
    - Зберігання даних (Database / Storage)
    - Зовнішні пристрої (датчики, обігрівач, кондиціонер)
4. **Позначте залежності та напрямки взаємодії:**
    - Використовуйте стрілки для показу, який компонент використовує інший.
5. **Опис проекту:**
    - Коротко поясніть, як компоненти взаємодіють між собою у вашій системі.
    - Наприклад:
      Користувач через UI задає параметри приміщення, UI передає команди менеджерам, менеджери зберігають дані в базі та керують сенсорами й пристроями.
6. **Використати UML-редактор для побудови діаграми компонентів:**
    - PlantUML, Visual Paradigm, StarUML, MS Visio або інший інструмент.
    - Діаграма повинна містити: компоненти, пакети (за бажанням), стрілки залежності та короткі пояснення.

Опціонально:

- Можна додати підкомпоненти для Sensor Manager або Device Manager (наприклад, TemperatureSensor, HumiditySensor, Heater, AirConditioner).

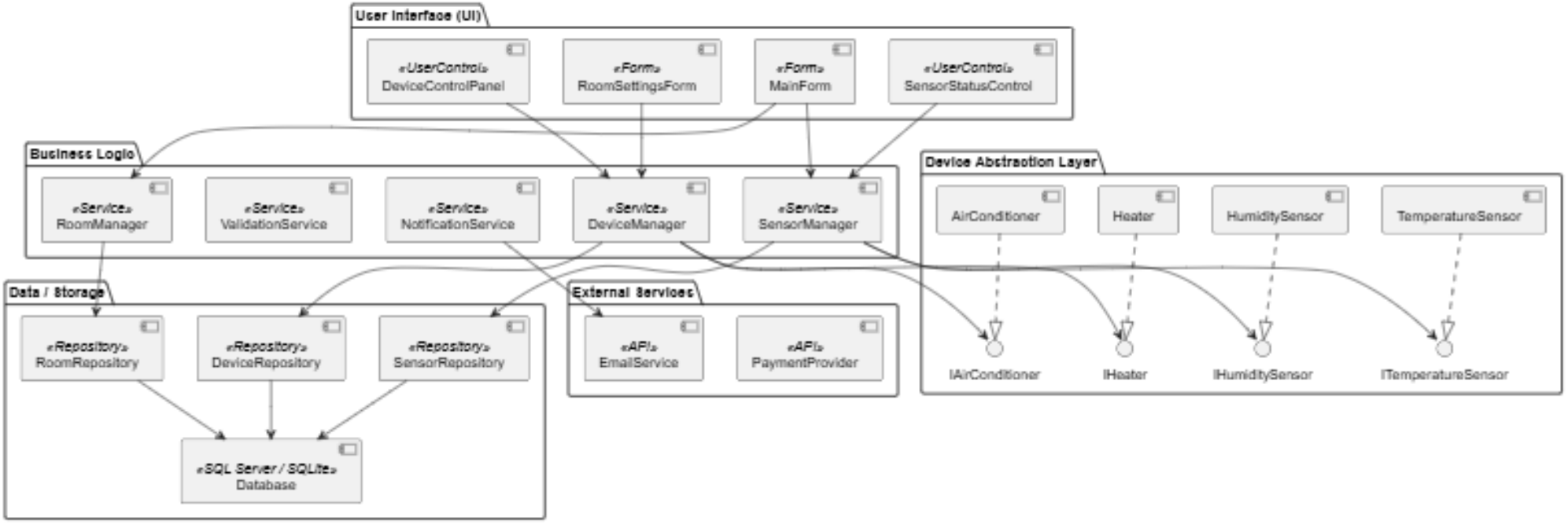
- Вказати інтерфейси між менеджерами та зовнішніми пристроями.

Приклад .NET Desktop Application (Windows Forms або WPF)

```
@startuml
' ========================
' UI Layer
' ========================
package "User Interface (UI)" {
    [MainForm] <<Form>>
    [RoomSettingsForm] <<Form>>
    [SensorStatusControl] <<UserControl>>
    [DeviceControlPanel] <<UserControl>>
}

' ========================
' Business Logic Layer
```

```
' ========================
package "Business Logic" {
    [RoomManager] <<Service>>
    [SensorManager] <<Service>>
    [DeviceManager] <<Service>>
    [ValidationService] <<Service>>
    [NotificationService] <<Service>>
}

' ========================
' Device Interfaces & Abstractions
' ========================
package "Device Abstraction Layer" {
    interface ITemperatureSensor
    interface IHumiditySensor
    interface IHeater
    interface IAirConditioner

    [TemperatureSensor] ..|> ITemperatureSensor
    [HumiditySensor] ..|> IHumiditySensor
    [Heater] ..|> IHeater
    [AirConditioner] ..|> IAirConditioner
}

' ========================
' Data Layer
' ========================
package "Data / Storage" {
    [RoomRepository] <<Repository>>
    [SensorRepository] <<Repository>>
    [DeviceRepository] <<Repository>>
    [Database] <<SQL Server / SQLite>>
}

' ========================
' External Services
' ========================
package "External Services" {
    [PaymentProvider] <<API>>
    [EmailService] <<API>>
}

' ========================
' Connections
' ========================

' UI -> Business Logic
[MainForm] --> [RoomManager]
[MainForm] --> [SensorManager]
[RoomSettingsForm] --> [DeviceManager]
[SensorStatusControl] --> [SensorManager]
[DeviceControlPanel] --> [DeviceManager]

' Business Logic -> Device Abstraction
[SensorManager] --> ITemperatureSensor
[SensorManager] --> IHumiditySensor
[DeviceManager] --> IHeater
[DeviceManager] --> IAirConditioner

' Business Logic -> Data Layer
[RoomManager] --> [RoomRepository]
```

```
[SensorManager] --> [SensorRepository]
[DeviceManager] --> [DeviceRepository]

' Repositories -> Database
[RoomRepository] --> [Database]
[SensorRepository] --> [Database]
[DeviceRepository] --> [Database]

' Optional External Services
[NotificationService] --> [EmailService]

@enduml
```


User Interface (UI)
«UserControl»
DeviceControlPanel
«Form»
RoomSettingsForm
«Form»
MainForm
«UserControl»
SensorStatusControl
Business Logic
«Service»
RoomManager
«Service»
ValidationService
«Service»
NotificationService
«Service»
DeviceManager
«Service»
SensorManager
Device Abstraction Layer
AirConditioner
Heater
HumiditySensor
TemperatureSensor
IAirConditioner
IHeater
IHumiditySensor
ITemperatureSensor
Data / Storage
«Repository»
RoomRepository
«Repository»
DeviceRepository
«Repository»
SensorRepository
«SQL Server / SQLite»
Database
External Services
«API»
EmailService
«API»
PaymentProvider