Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Інформаційні системи

Викладач: к.т.н., доц. Саяпіна Інна Олександрівна

План заняття:

- Мікросервісна архітектура (Microservice Architecture):
  - Історія мікросервісів
  - Проблеми з Monolith & SOA
  - Архітектура мікросервісів, основні характеристики
  - Проблеми, які вирішує мікросервісна архітектура

Навіщо вивчати мікросервіси?

- Компанії, що використовують мікросервіси

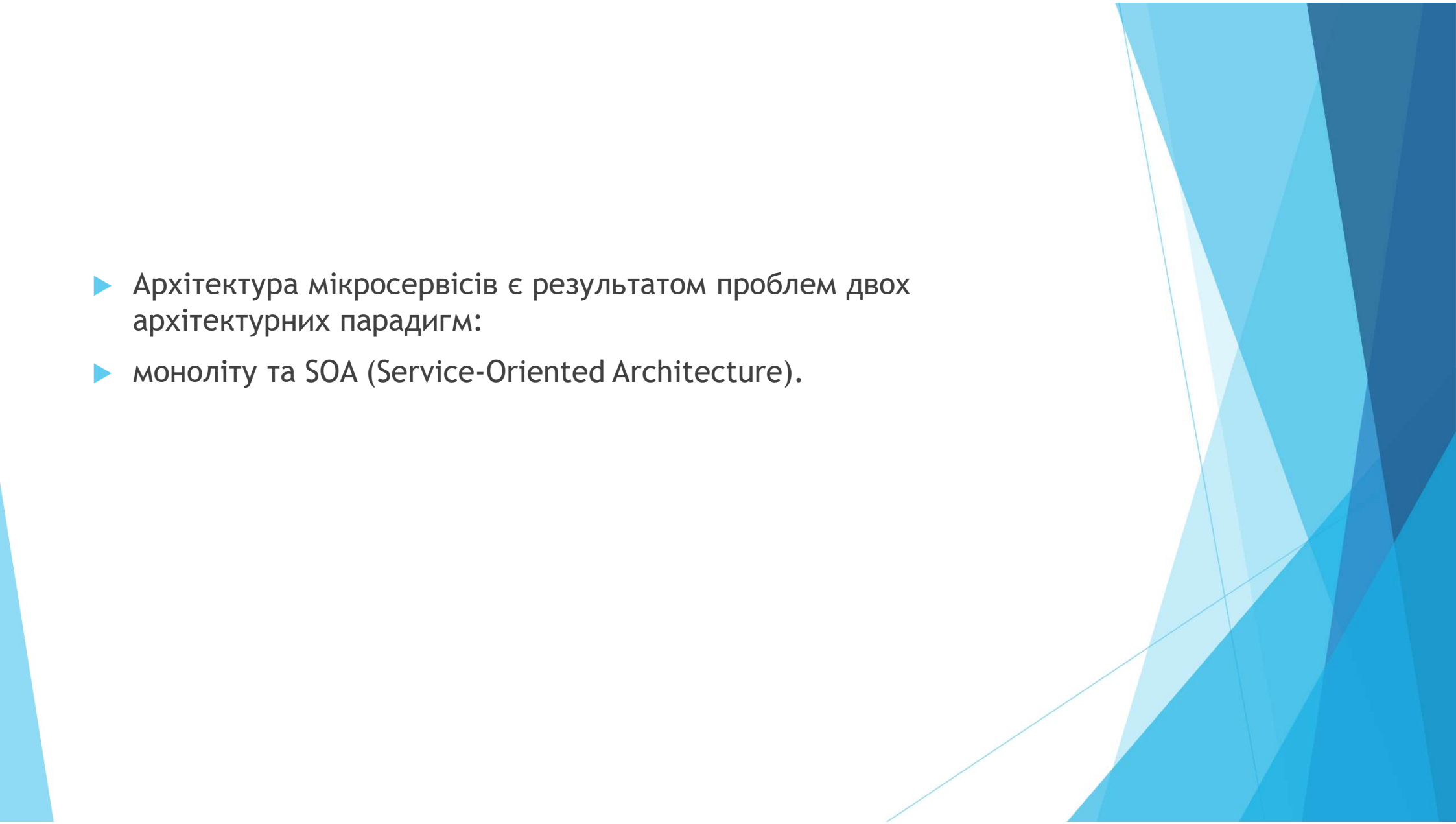
- Знаходить дуже широку популярність
- Є платформенно незалежною

- Дуже зручна у масштабуванні

Але…

- «Якщо мікросервіси реалізовані неправильно або використовуються як допоміжний засіб без усунення деяких основних недоліків у вашій системі, ви не зможете розробити новий продукт, оскільки потонете в складності».
  О. Нунан, інженер компанії Segment

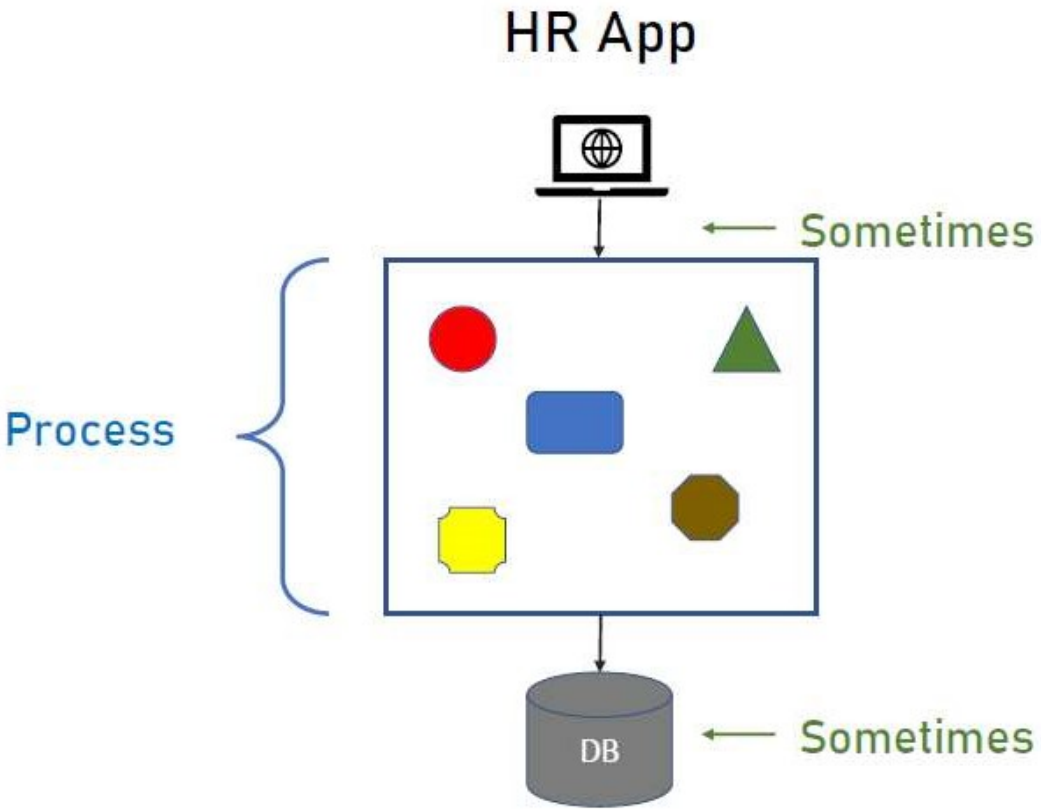
- Архітектура мікросервісів є результатом проблем двох архітектурних парадигм:
- моноліту та SOA (Service-Oriented Architecture).



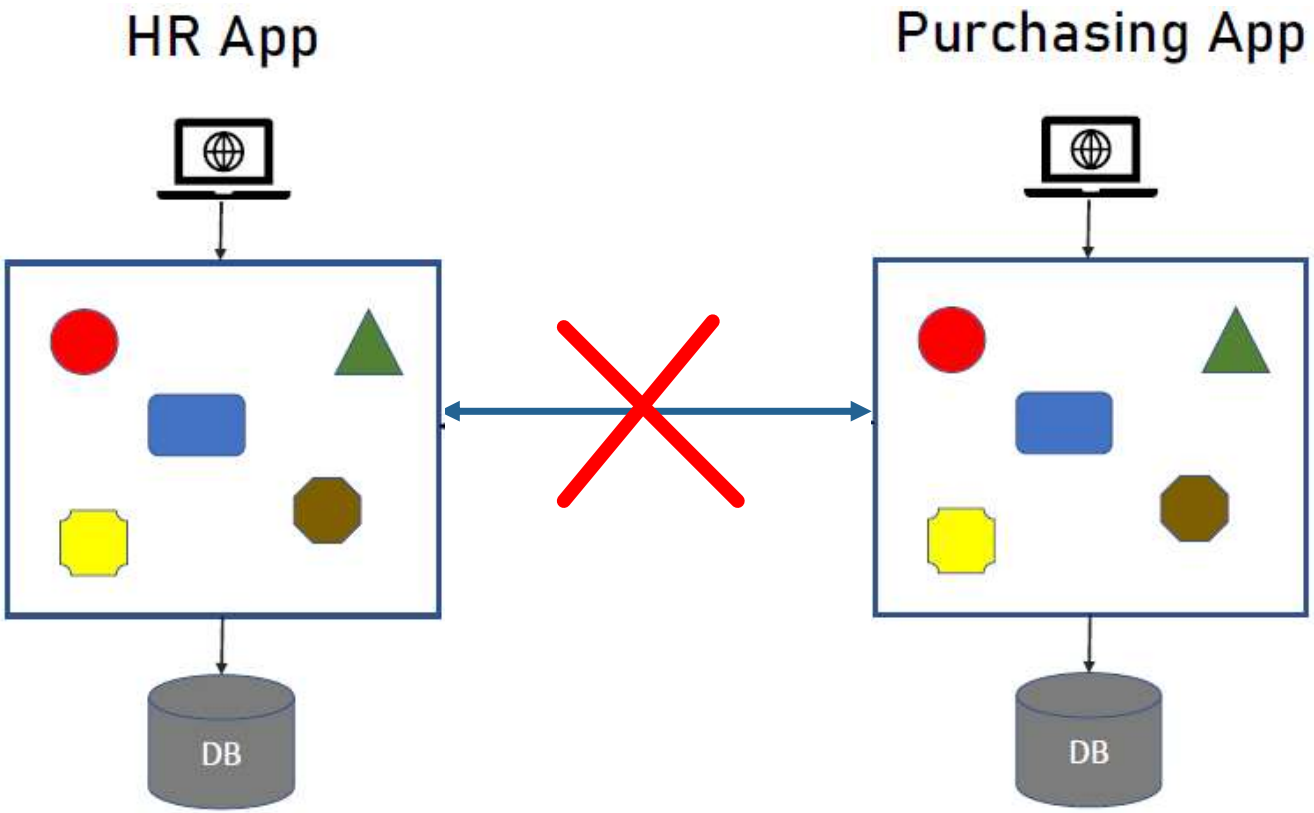
Монолітна архітектура

- Монолітна архітектура - предок усіх інших архітектур
- Усі компоненти програмного забезпечення виконуються в одному процесі
- Сильний зв'язок між усіма класами
- Зазвичай реалізується як silo

Приклад монолітної архітектури



Приклад монолітної архітектури



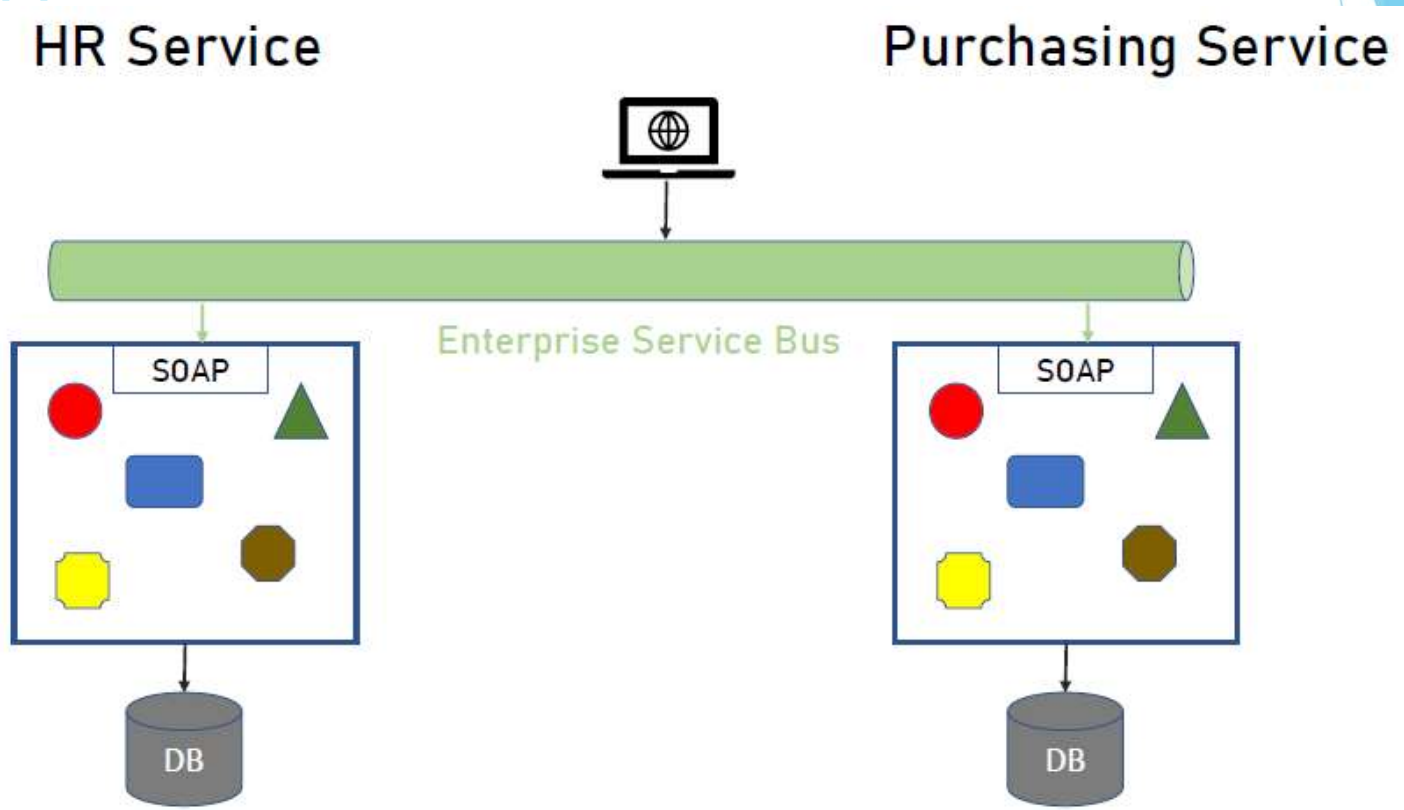
Переваги монолітної архітектури

- Легкість проєктування
- Продуктивність

Сервіс-орієнтована архітектура (SOA)

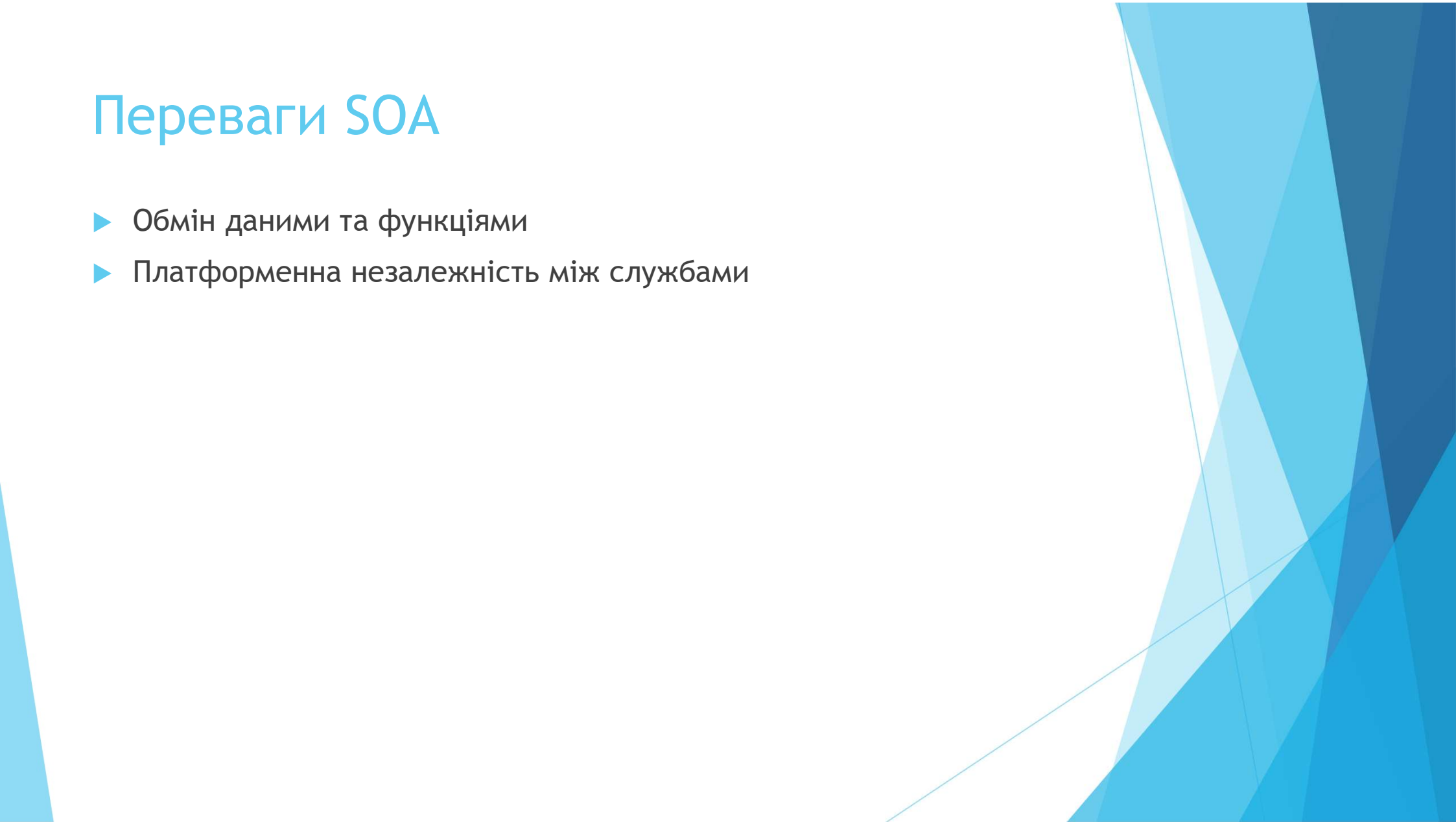
- Вперше згадується у 1998 році
- Програми — це сервіси, що надають певні функціональні можливості зовнішньому світу
- Сервіси надають метадані для оголошення своєї функціональності
- Зазвичай реалізується за допомогою SOAP і WSDL
- Зазвичай реалізується за допомогою ESB (Enterprise Service Bus)

Приклад SOA


HR Service
Purchasing Service
Enterprise Service Bus
SOAP
SOAP
DB
DB

Переваги SOA

- Обмін даними та функціями
- Платформенна незалежність між службами



Характеристики SOA

Architecture characteristic	Star rating
Partitioning type	Technical
Number of quanta	1
Deployability	★
Elasticity	★★★
Evolutionary	★
Fault tolerance	★★★
Modularity	★★★
Overall cost	★
Performance	★★
Reliability	★★
Scalability	★★★★
Simplicity	★
Testability	★

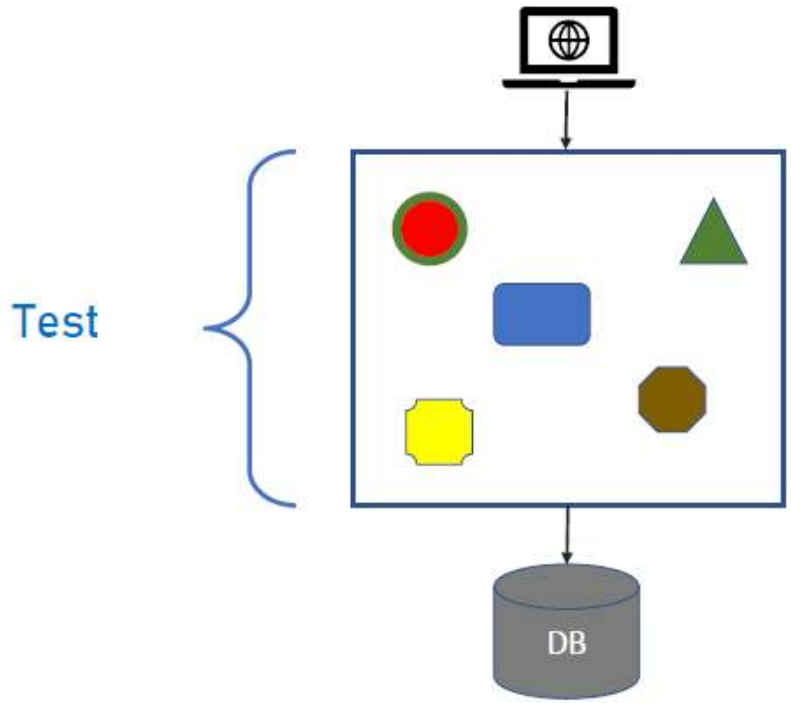
Недоліки моноліту

- **Єдина технологічна платформа:**
  - З монолітом усі компоненти повинні бути розроблені за допомогою однакової платформи розробки
  - Ця платформа не завжди найкраща для даного завдання
  - Не можна використовувати іншу зручнішу платформу платформу для певних функцій
  - Подальше оновлення — це проблема, яка потребує оновлення всього додатка

Недоліки моноліту

- **Негнучке розгортання:**
  - З монолітом нове розгортання завжди стосується всієї програми
  - Неможливо розгорнути лише частину програми
  - Навіть при оновленні лише одного компонента розгортається вся кодова база
  - Вимагає ретельного тестування для кожного розгортання
  - Потребує тривалих циклів розвитку

Недоліки моноліту. Приклад



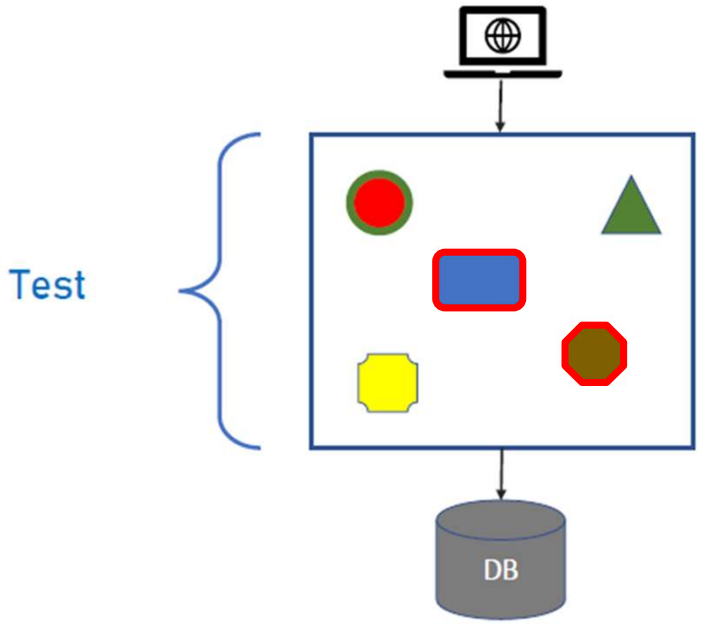
Недоліки моноліту.

- **Неефективне використання ресурсів:**
  - У моноліті обчислювальні ресурси (ЦП і оперативна пам’ять) розподіляються між усіма компонентами
  - Якщо певний компонент потребує більше ресурсів, це неможливо зробити
  - Дуже неефективний

Недоліки моноліту. Приклад

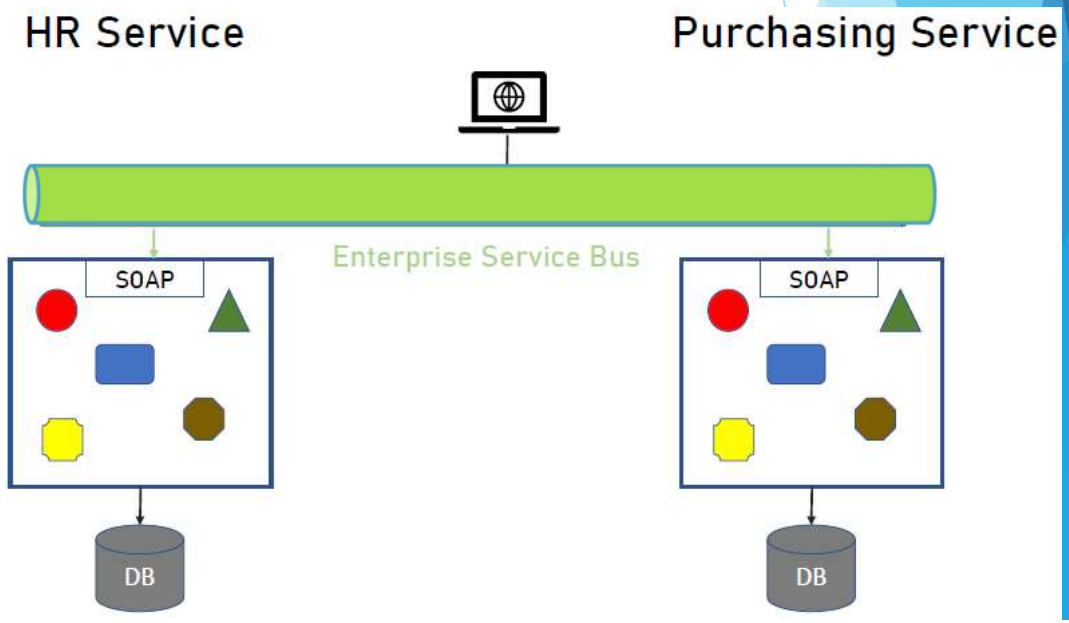
Недоліки моноліту

- **Складність та великий обсяг коду:**
  - З монолітом кодова база є великою та складною
  - Кожна маленька зміна може вплинути на інші компоненти
  - Тестування не завжди виявляє всі помилки
  - Дуже важко підтримувати
  - Може зробити систему застарілою



Недоліки SOA

- **Складна та дорога ESB (Enterprise Service Bus):**
  - У SOA ESB є одним із основних компонентів
  - Може швидко стати роздутим і дорогим
  - Намагається зробити все
  - Дуже важко підтримувати



Недоліки SOA

- **Недостатність інструментарію**
  - Щоб SOA була ефективною, потрібні були короткі цикли розробки
  - Не було забезпечено швидке тестування та розгортання
  - Не існувало інструментів для забезпечення цього
  - Економії часу не досягнуто

Поява Мікросервісів

- Проблеми з монолітом і SOA призвели до нової парадигми
- Має бути модульним із простим API
- Термін «мікросервіси» вперше з'явився в 2011 році
- У 2014 році Мартін Фаулер і Джеймс Льюїс опублікували свою статтю «Мікросервіси». https://martinfowler.com/articles/microservices.html
- Став стандартом де-факто для визначення мікросервісів

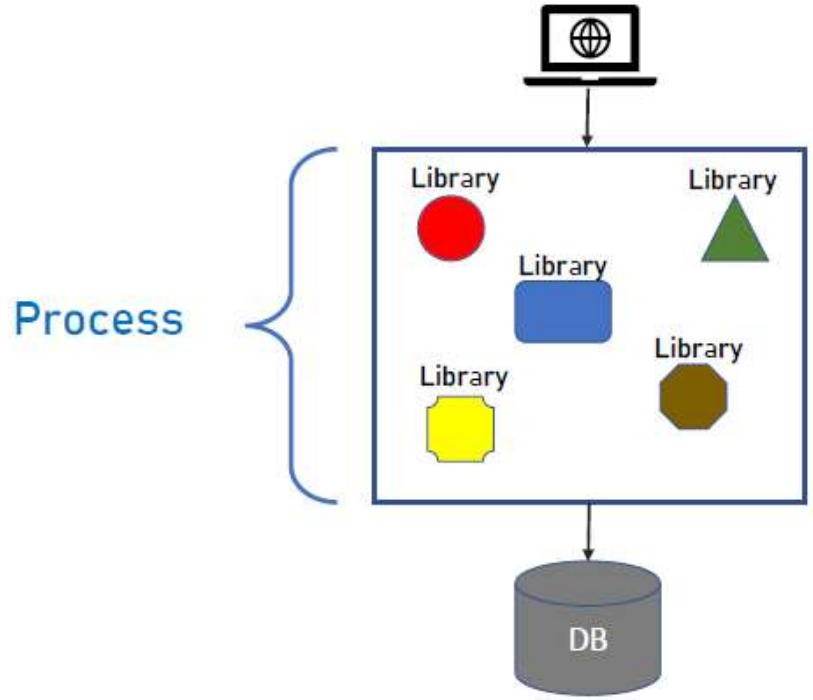
9 характеристик гарної архітектури мікросервісів:

- Компонентизація на основі сервісів
- Організація навколо бізнес-можливостей
- Продукти, а не проєкти
- Smart End points та Dumb Pipes
- Децентралізоване управління
- Децентралізоване керування даними
- Автоматизація інфраструктури
- Дизайн на випадок відмов
- Еволюційний дизайн

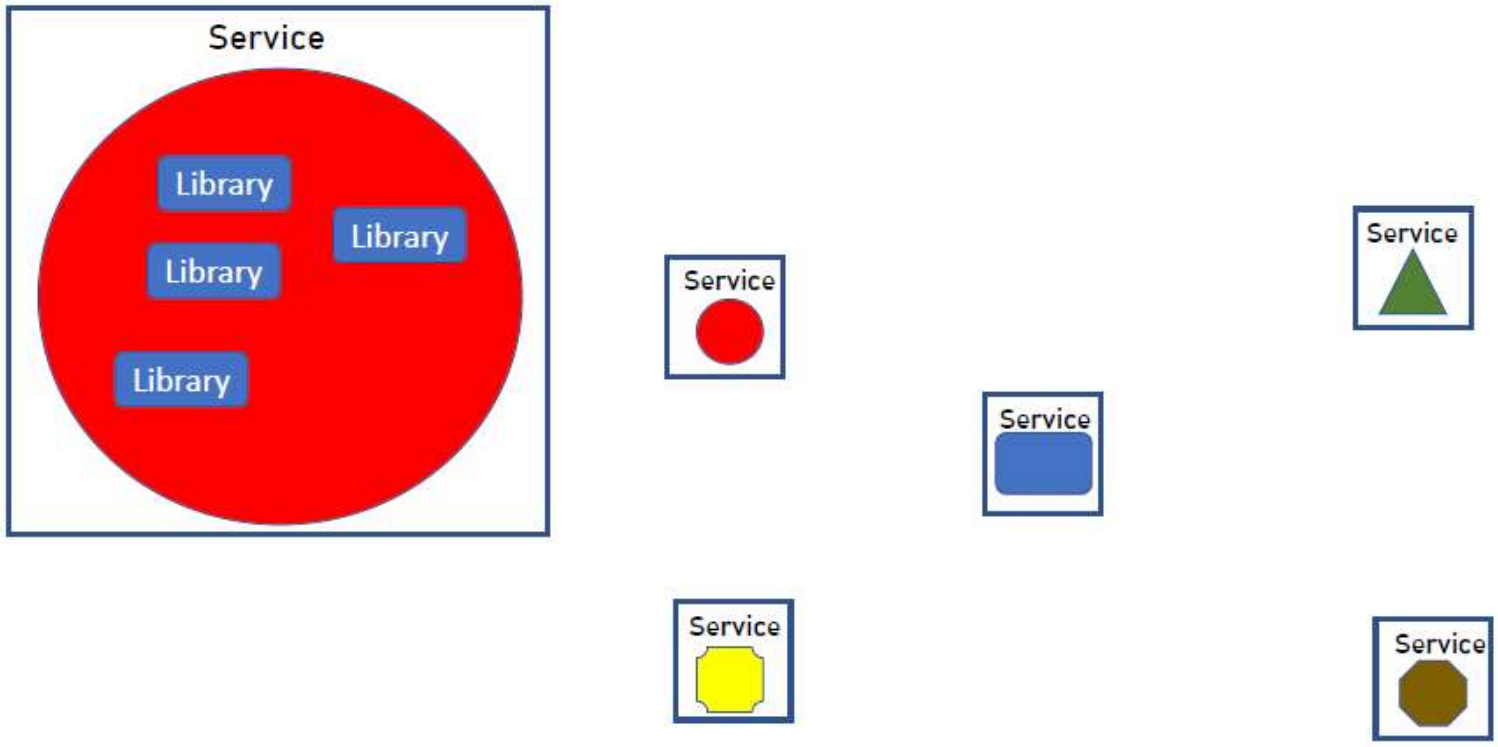
Компонентизація на основі сервісів

- Модульний дизайн – це завжди гарна ідея
- Компоненти — це назви частин, які разом складають програмне забезпечення
- Модульності можна досягти за допомогою:
  - Бібліотеки, що викликаються безпосередньо в процесі. Це зовнішні файли коду, які використовуються у вашому коді, як правило, після їх оголошення за допомогою ключових слів import, require або using.
  - Служби, викликані позапроцесним механізмом (веб-API, RPC)
- У мікросервісах перевага надається використанню компонентизації на основі сервісів
- Бібліотеки можна використовувати всередині сервісу

Компонентизація на основі бібліотек

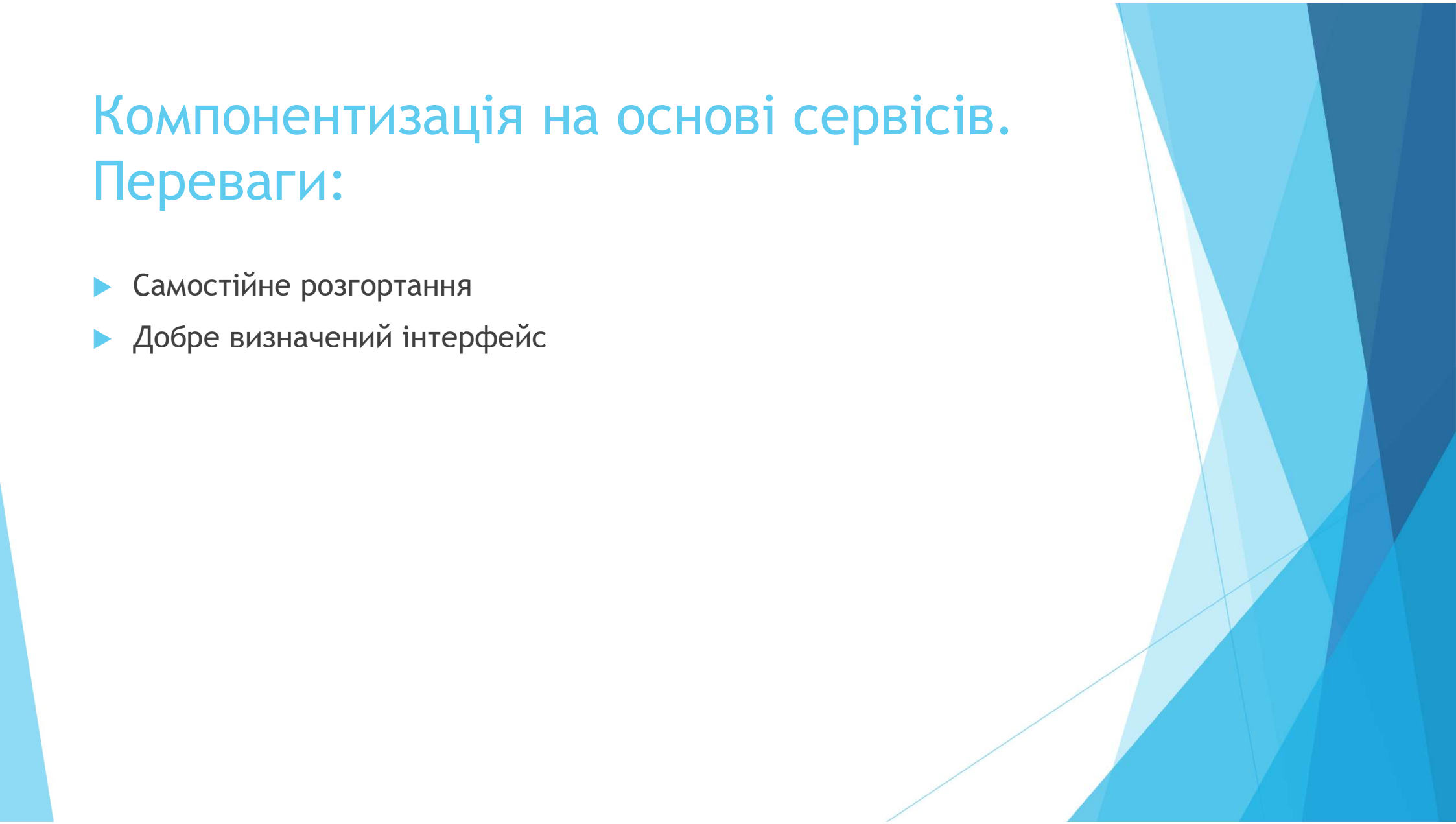


Компонентизація на основі сервісів



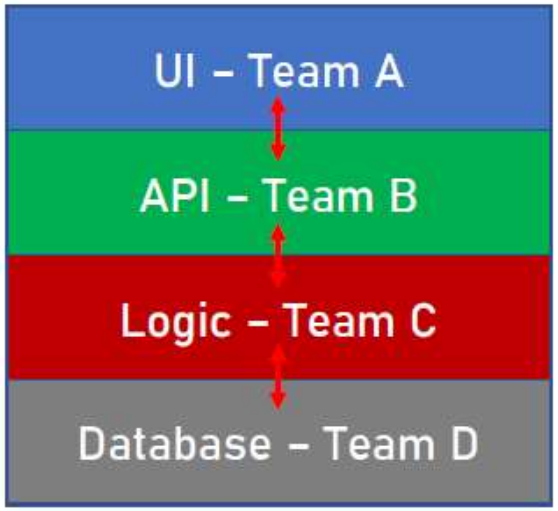
Компонентизація на основі сервісів. Переваги:

- Самостійне розгортання
- Добре визначений інтерфейс



Організація навколо бізнес-можливостей

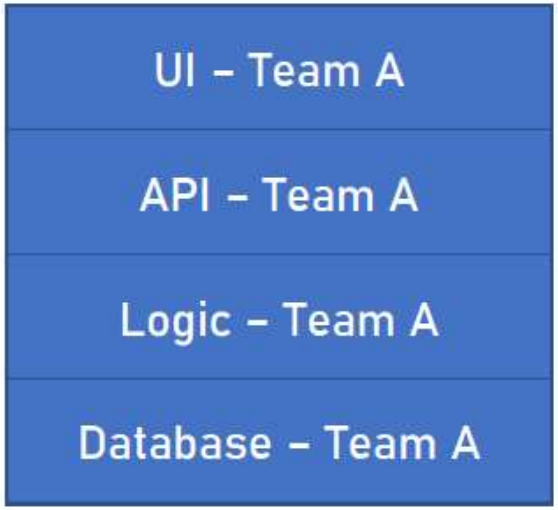
- Традиційні проекти мають команди з горизонтальними обов'язками: інтерфейс користувача UI, API, логіка, база даних тощо



повільна, громіздка міжгрупова взаємодія

Організація навколо бізнес-можливостей

- У мікросервісах кожна послуга обробляється однією командою, яка відповідає за всі аспекти



Організація навколо бізнес-можливостей

- У мікросервісах кожна служба обробляє чітко визначені бізнес-можливості.

Організація навколо бізнес-можливостей. Переваги

- Швидка розробка
- Чітко визначені межі сервісу