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):
  - Архітектура мікросервісів, основні характеристики
  - Проблеми, які вирішує мікросервісна архітектура
  - Процес проєктування архітектури
    - Визначення компонентів
    - Визначення шаблонів зв’язку

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

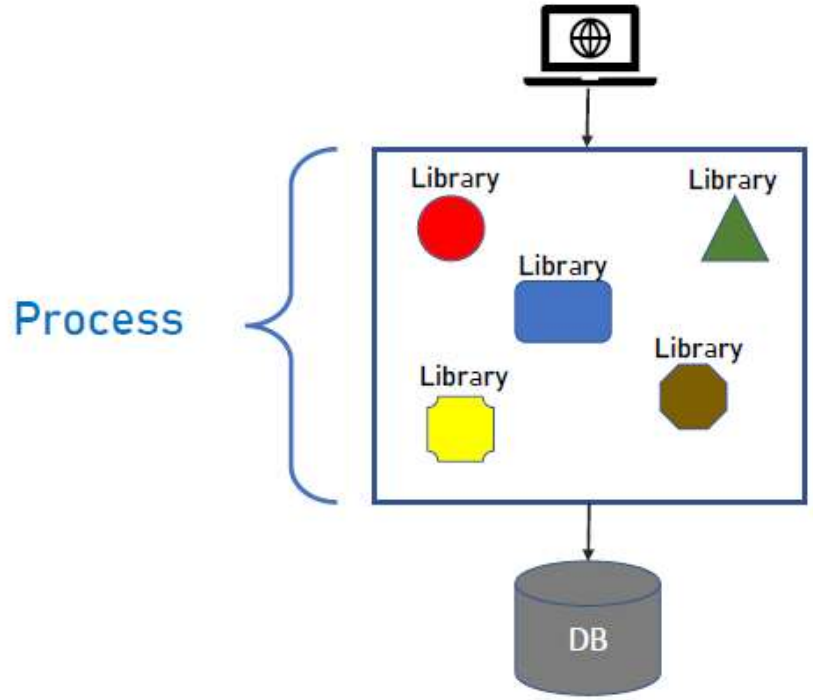
9 характеристик гарної архітектури мікросервісів:

- 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

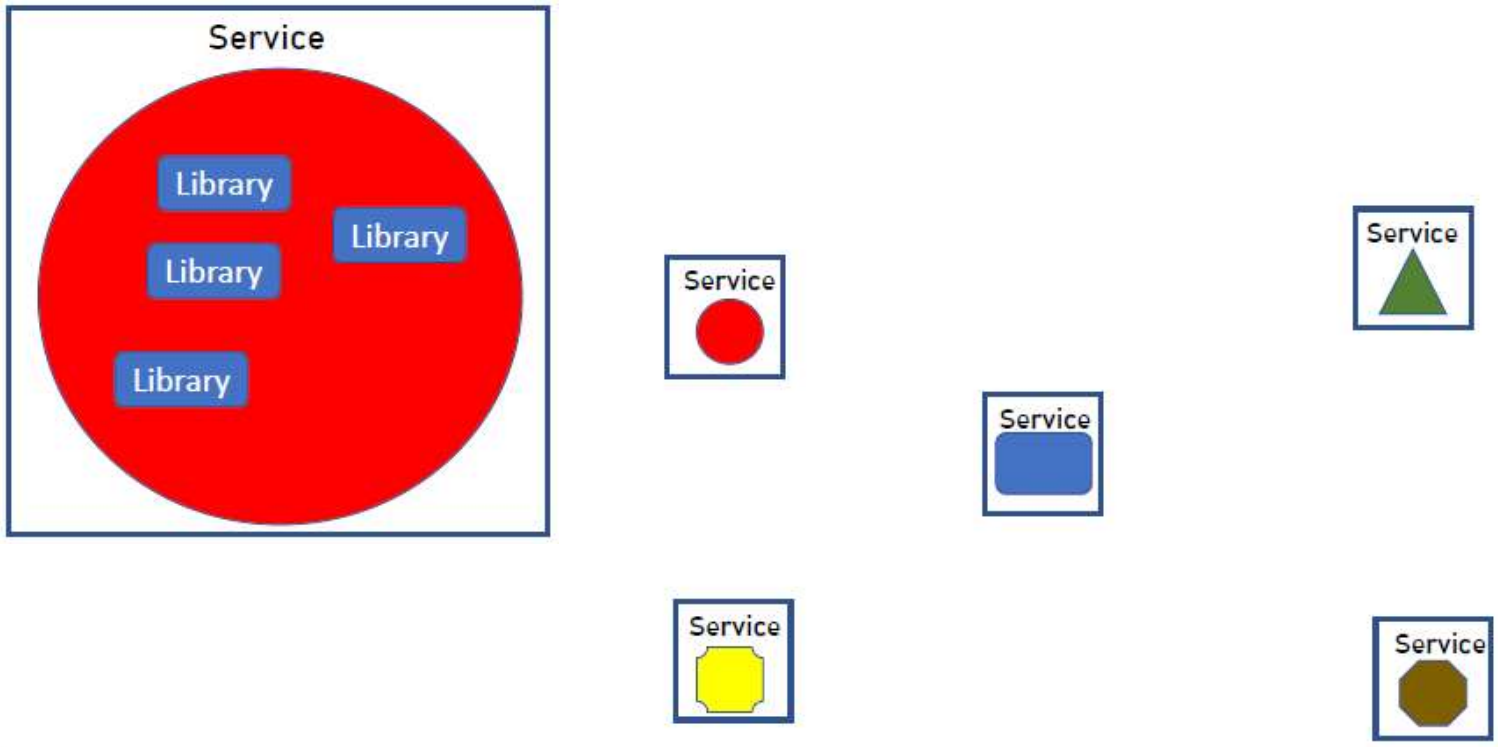
Компонентизація на основі сервісів

- 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

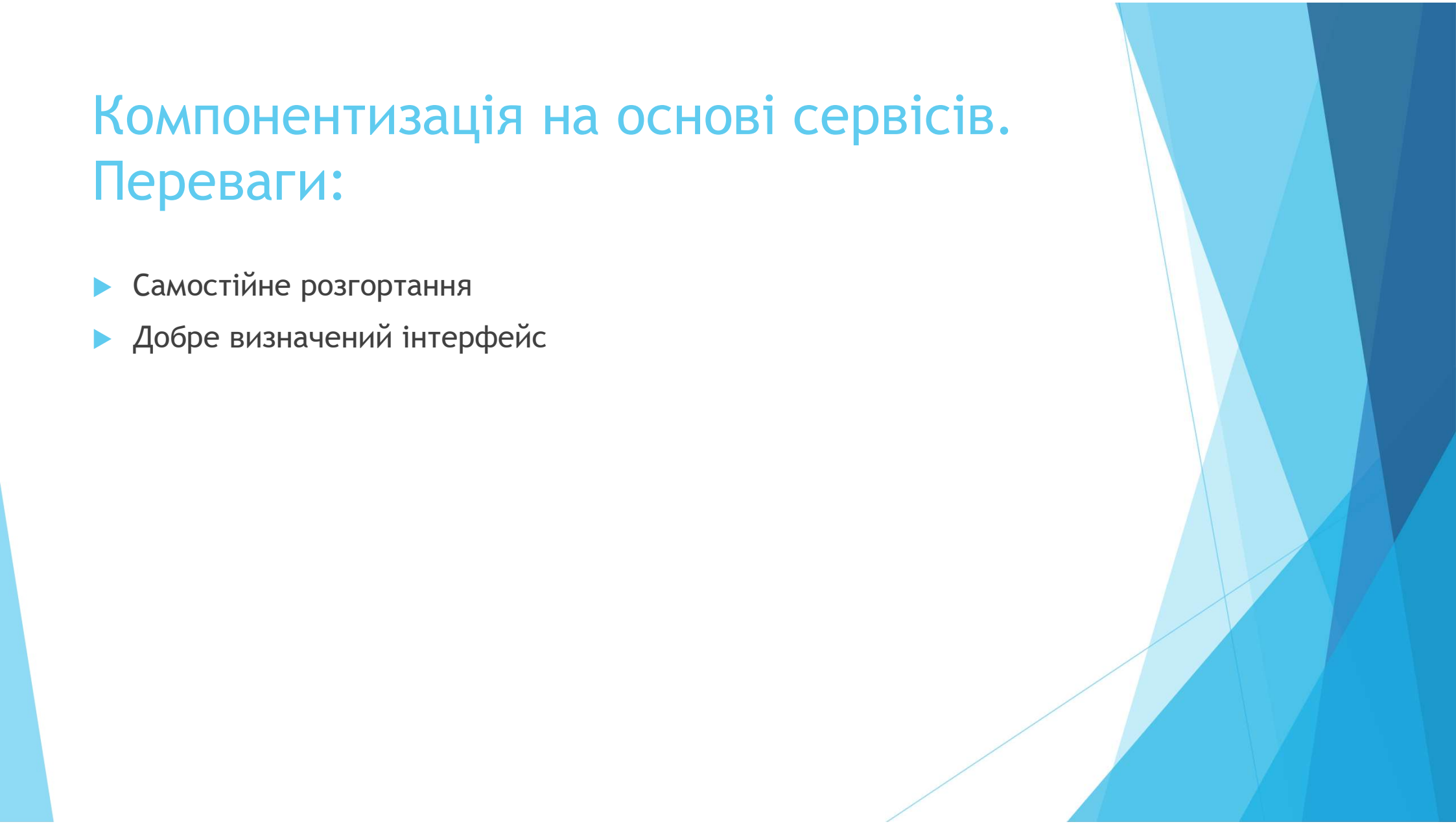


Компонентизація на основі сервісів



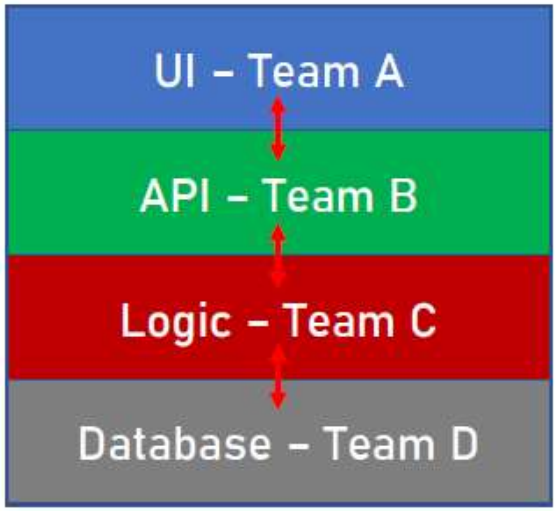
Компонентизація на основі сервісів. Переваги:

- Самостійне розгортання
- Добре визначений інтерфейс



Організація навколо бізнес-можливостей

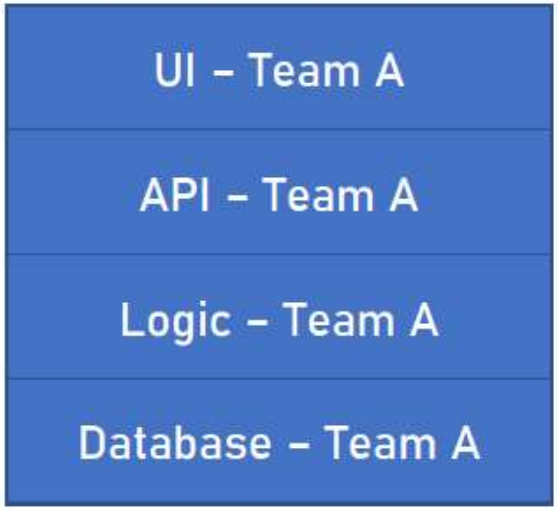
- Традиційні проекти мають команди з горизонтальними обов'язками: інтерфейс користувача UI, API, логіка, база даних тощо



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

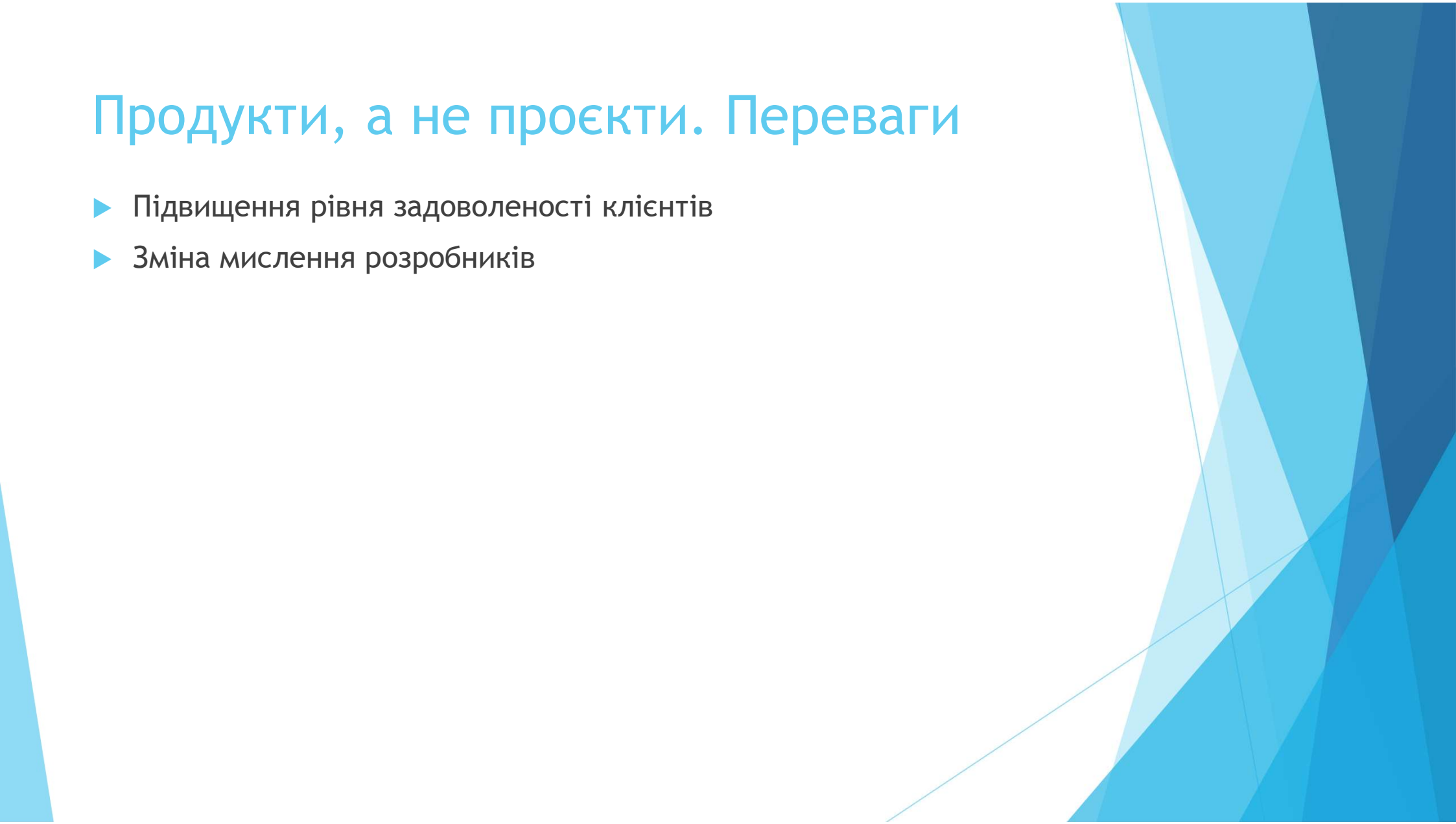
Продукти, а не проєкти

- У традиційних проєктах метою є надання робочого коду
- Відсутність тривалих відносин із замовником
- Часто немає знайомства із замовником
- Після доставки ПЗ команда переходить до наступного проекту

- У Мікросервісах:
- Метою є надання робочого продукту
- Продукт потребує постійної підтримки та тісних стосунків із замовником
- Команда також відповідає за мікросервіс після доставки

Продукти, а не проєкти. Переваги

- Підвищення рівня задоволеності клієнтів
- Зміна мислення розробників

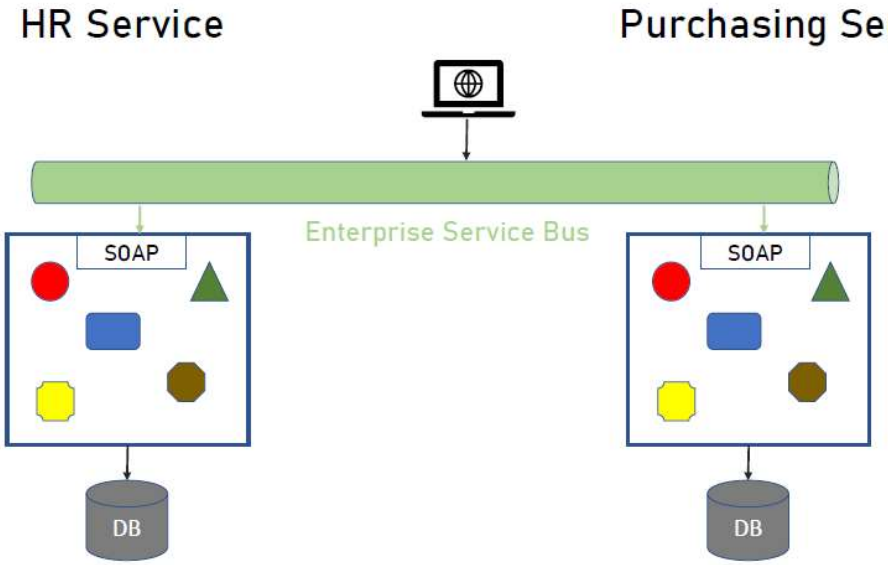


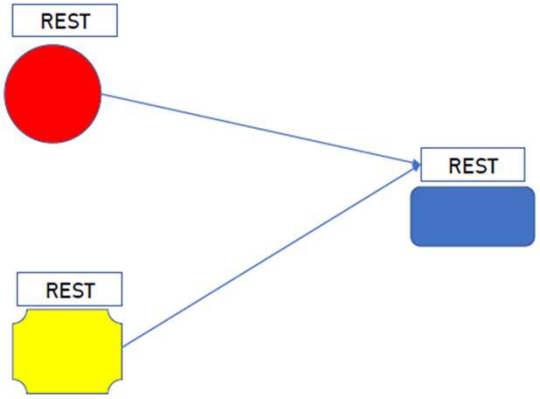
Smart Endpoints та Dumb Pipes

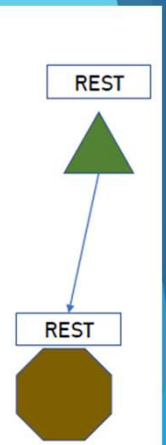
- Традиційні проекти SOA використовували два складні механізми:
  - ESB
  - WS --* протоколи
- Складність підтримки та ускладнена комунікація між сервісами

- У мікросервісах:
  - Використання простих протоколів для забезпечення «Dumb Pipes»
  - Прагнення використовувати те, що вже пропонує Інтернет
  - Зазвичай REST API, найпростіший існуючий API

Smart Endpoints та Dumb Pipes





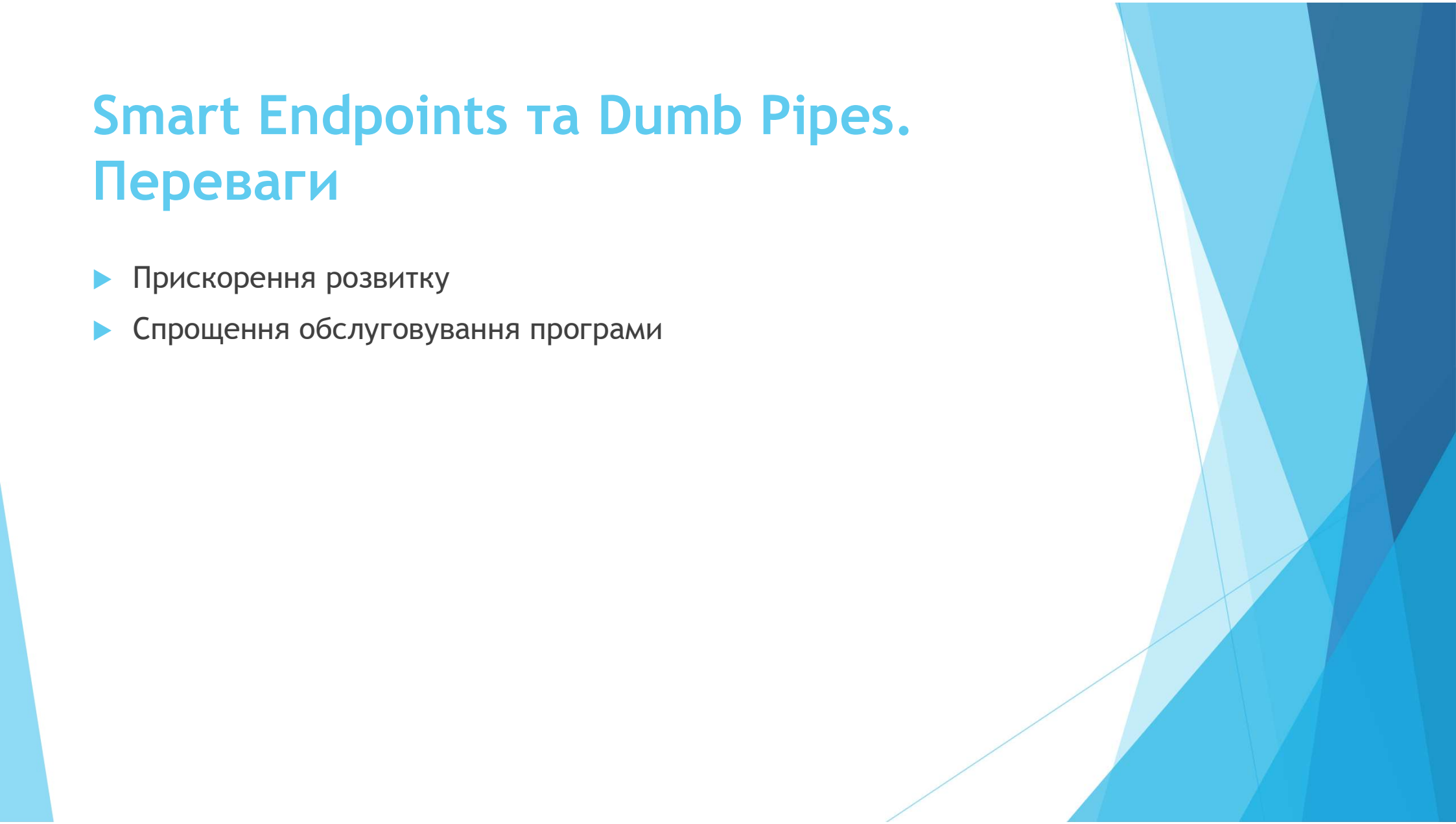


Smart Endpoints та Dumb Pipes

- Важливі зауваження:
  - Прямі зв’язки між службами не є гарною ідеєю
  - Краще скористайтеся службою виявлення(discovery service) або шлюзом (gateway)
  - В останні роки було введено більше протоколів (GraphQL ,gRPC), деякі з них досить складні

Smart Endpoints та Dumb Pipes. Переваги

- Прискорення розвитку
- Спрощення обслуговування програми



Децентралізоване управління

- У традиційних проєктах є стандарт практично для всього:
  - Яку платформу для розробників використовувати
  - Яку базу даних використовувати
  - Як створюються журнали і т.п.

  - З мікросервісами кожна команда приймає власні рішення:
    - Яку платформу для розробників використовувати
    - Яку базу даних використовувати
    - Як створюються журнали і т.д.

Децентралізоване управління

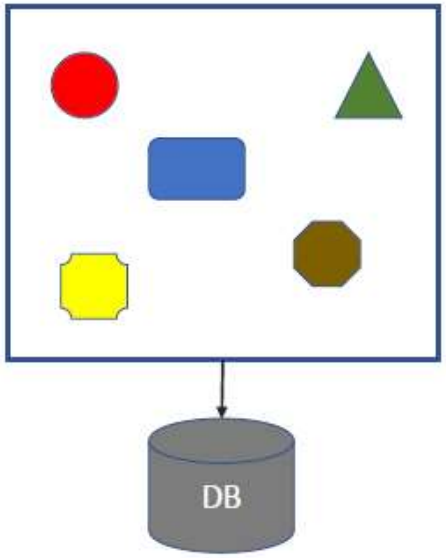
- Кожна команда несе повну відповідальність за своє обслуговування і тому прийматиме оптимальні рішення.
- Це забезпечується завдяки слабозв’язаній природі мікросервісів
- Polyglot – використання декількох платформ розробки в одній системі, що забезпечує платформенну незалежність

Децентралізоване управління. Перевага

- Дозволяє приймати оптимальні технологічні рішення для конкретного сервісу

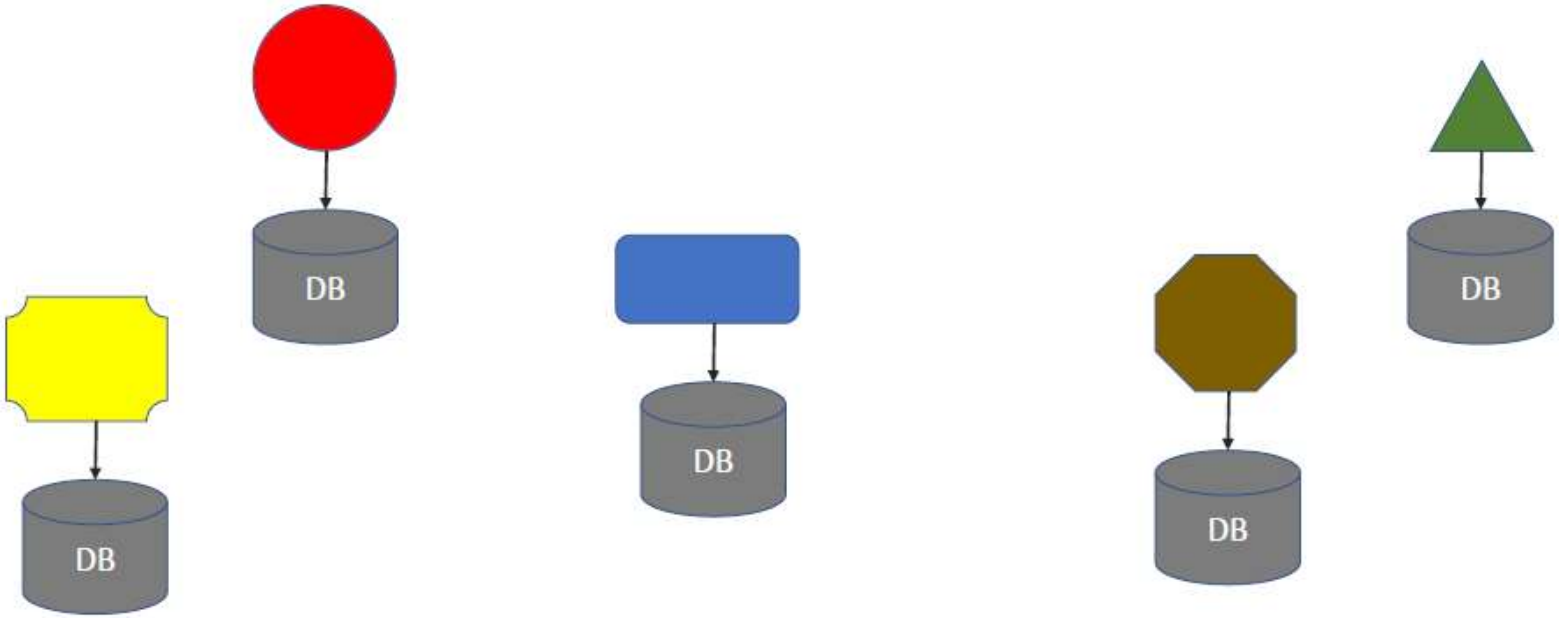
Децентралізоване керування даними

- Традиційні системи мають єдину базу даних
- Вона зберігає усі дані системи з усіх компонентів



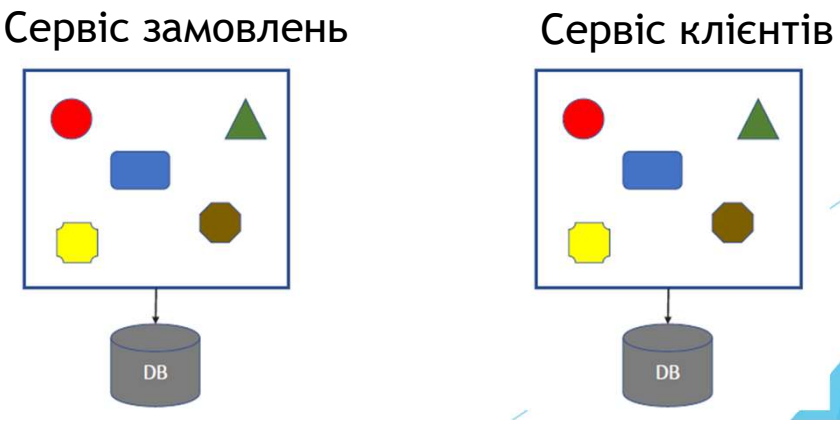
Децентралізоване керування даними

- У мікросервісах кожна служба має власну базу даних.



Децентралізоване керування даними

- Важливі зауваження:
  - Це найбільш суперечливий атрибут мікросервісів
  - Це не завжди можливо реалізувати
  - Викликає такі проблеми, як розподілені транзакції, дублювання даних та інше
  - Не треба наполягати на цьому. У кожному випадку має вирішуватись окремо.



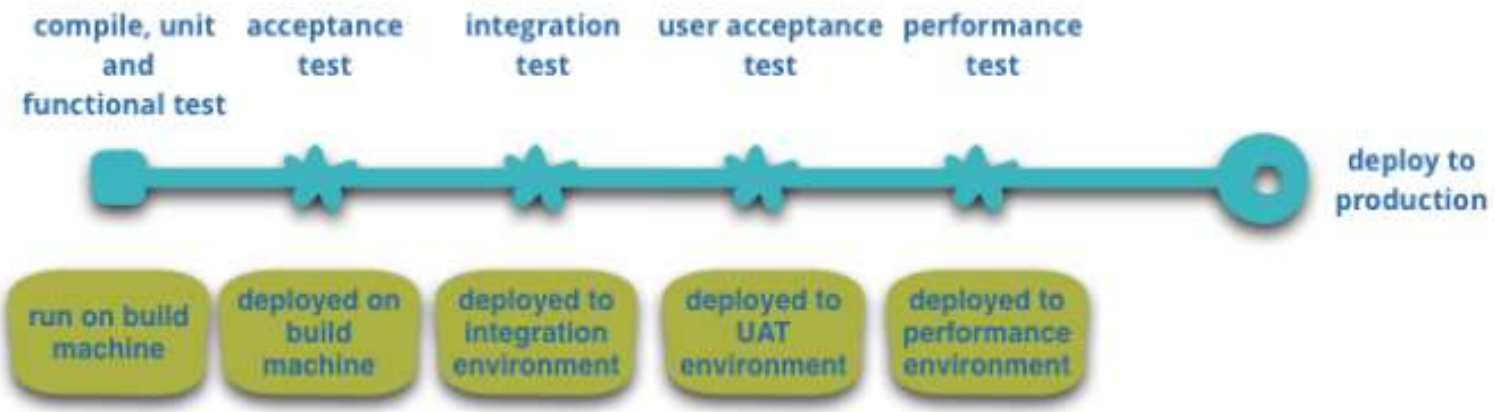
Децентралізоване керування даними. Переваги

- Застосування оптимальних інструментів баз даних для відповідного завдання
- Сприяє ізоляції сервісу

Автоматизація інфраструктури

- Парадигма SOA страждала від нестачі інструментів
- Інструменти значно допомагають у розгортанні за допомогою:
  - Автоматизованого тестування
  - Автоматичного розгортання

Автоматизація інфраструктури



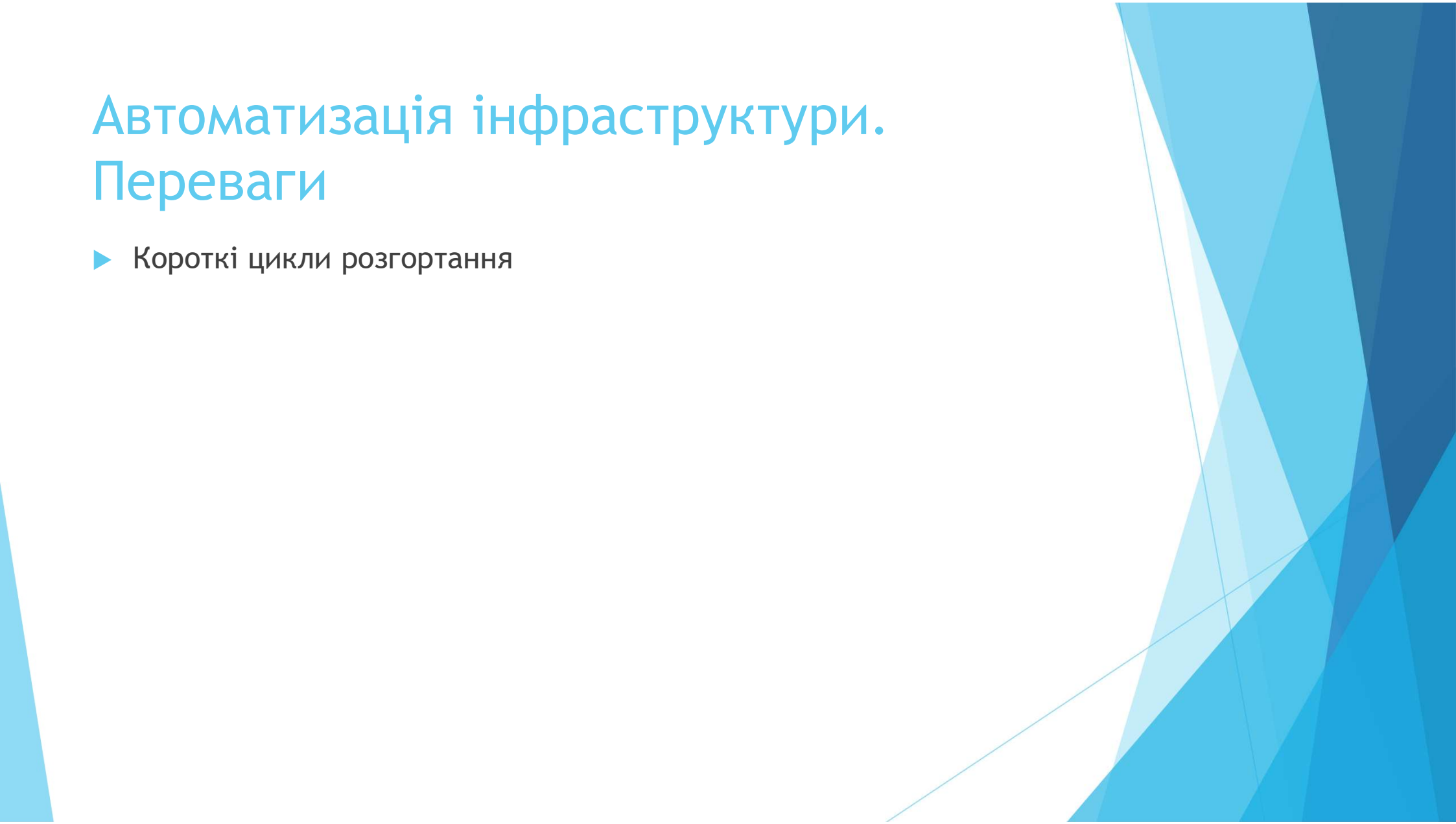
- Джерело: https://martinfowler.com/articles/microservices.html

Автоматизація інфраструктури

- Для мікросервісів важлива автоматизація
- Короткі цикли розгортання є критично важливими для архітектури мікросервісів, оскільки вони дозволяють організаціям бути більш гнучкими та реагувати на потреби бізнесу та клієнтів, мінімізуючи при цьому ризик виникнення проблем у системі.
- Розгортання та тестування не можна виконувати лише вручну
- Існує багато засобів автоматизації, наприклад:

Автоматизація інфраструктури. Переваги

- Короткі цикли розгортання

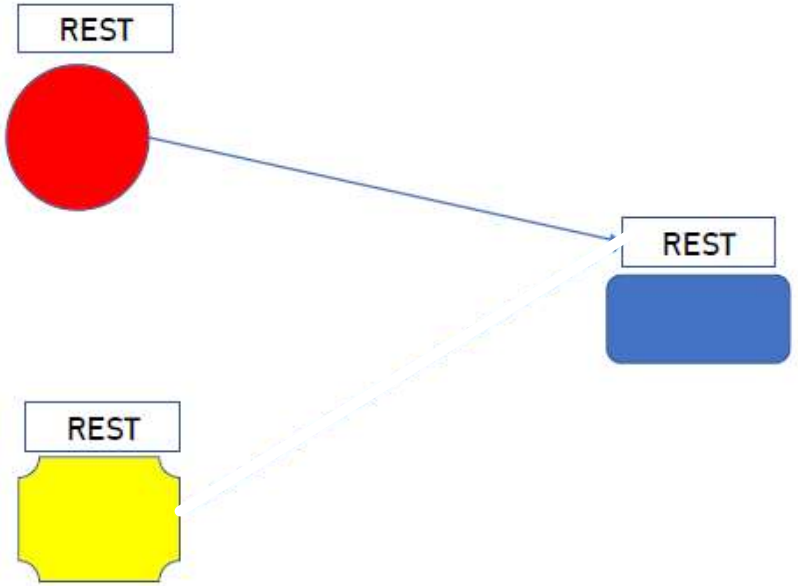


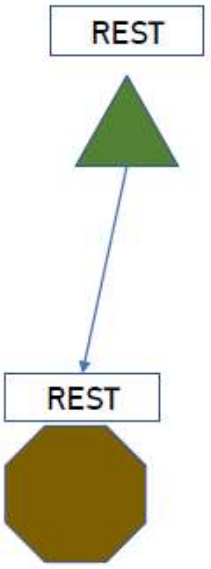
Дизайн на випадок відмов

- У мікросервісах є багато процесів і багато мережевого трафіку
- Багато що може піти не так
- Код повинен враховувати, що може статися збій, і грамотно його обробляти
- Повинне бути організоване розширене логування та моніторинг

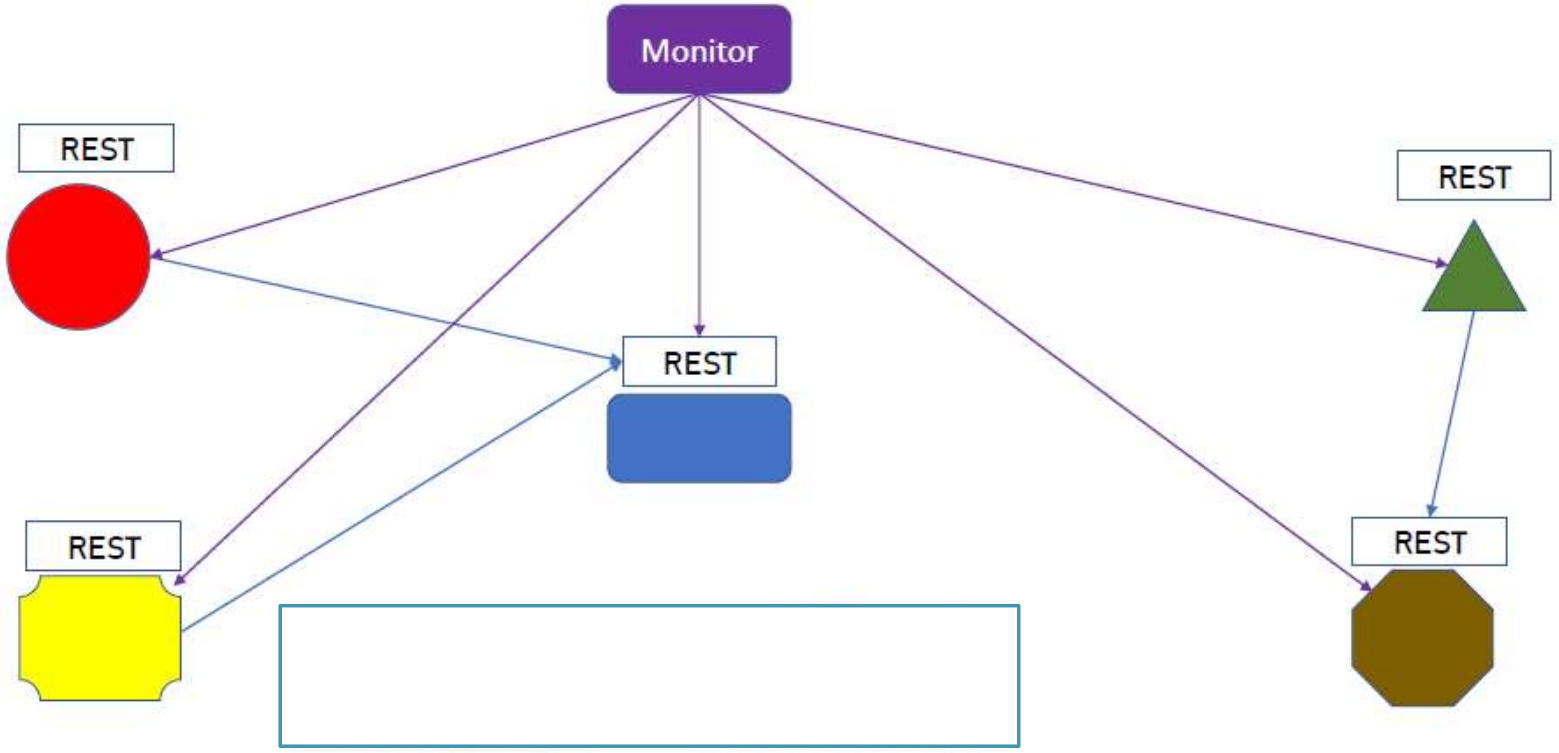
Дизайн на випадок відмов

- Передбачити та обробити виняток (наприклад, у блоці try-catch). Виняток — це помилка або несподівана поведінка, яка виникає під час виконання програми. Якщо її не обробляти належним чином, це може призвести до збою програми
- Спробувати відновити з’єднання
- Залогувати виняток





Дизайн на випадок відмов



Дизайн на випадок відмов. Переваги

- Підвищення надійності системи
- Покращення користувацького досвіду

Еволюційне проєктування

- Перехід до мікросервісів має бути поступовим
- Слід починати з малого та вдосконалювати кожну частину окремо. Поступова зміна дозволяє виявити, чи ми нічого не зруйнували

Висновки

- Це рекомендації, а не обов'язкові інструкції
- Приймайте те, що працює для вас
- Світ мікросервісів швидко змінюється
- Слідкуйте за новими інструментами API, моніторингу, хмарними сервісами тощо.

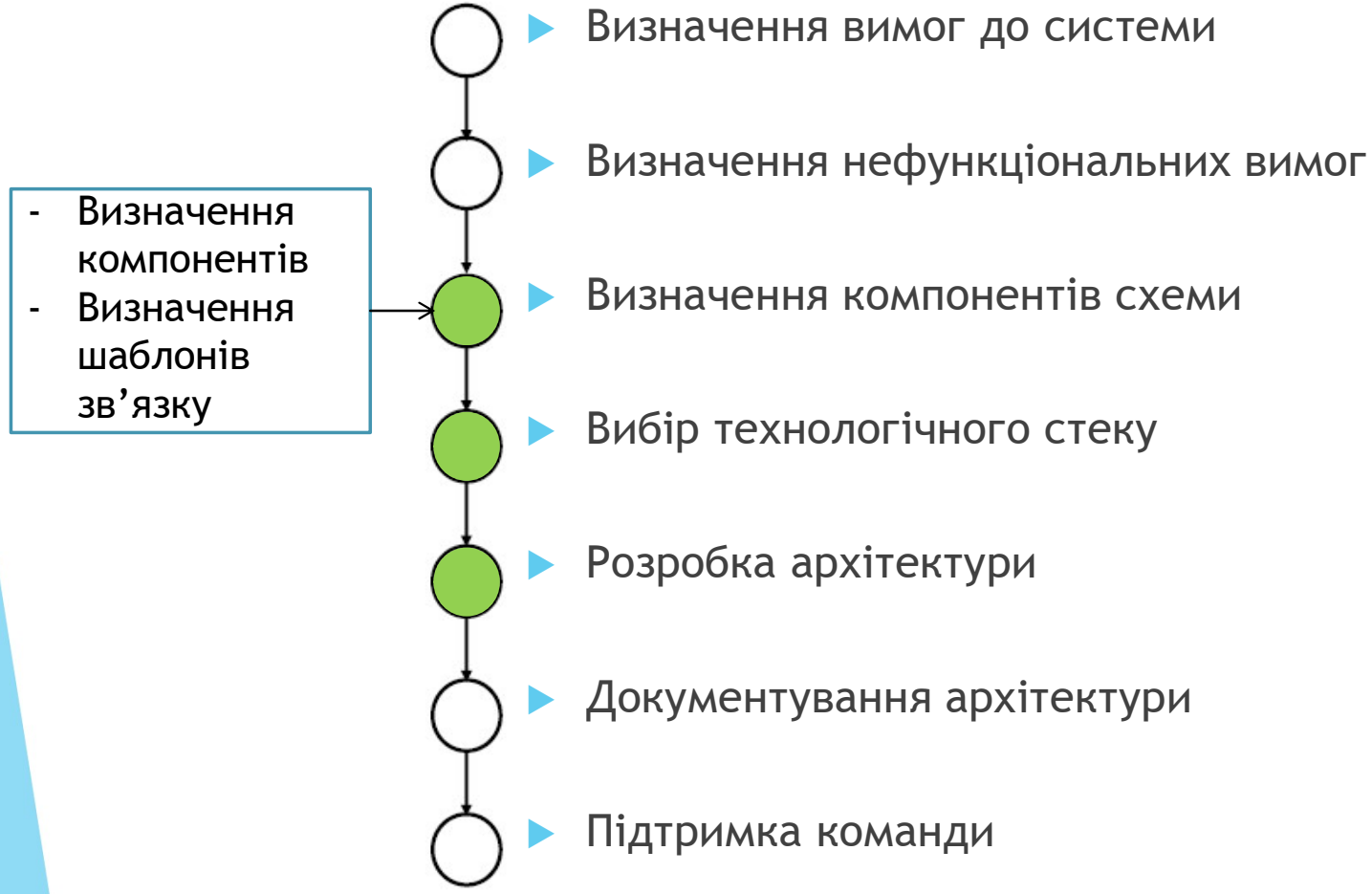
Висновки

- Найважливіші атрибути:
  - Компонентація навколо сервісів
  - Організація навколо бізнес-можливостей
  - Децентралізоване управління
  - Децентралізоване керування даними (якщо можливо)
  - Автоматизація інфраструктури

Процес проєктування архітектури

- Проектування архітектури мікросервісів має бути методичним
- Не розробляйте поспіхом
- «Більше планування, менше кодування»
- Вирішальне значення для успіху системи

Архітектурний процес



- Визначення вимог до системи
- Визначення нефункціональних вимог
- Визначення компонентів схеми
- Вибір технологічного стеку
- Розробка архітектури
- Документування архітектури
- Підтримка команди