Manual QA

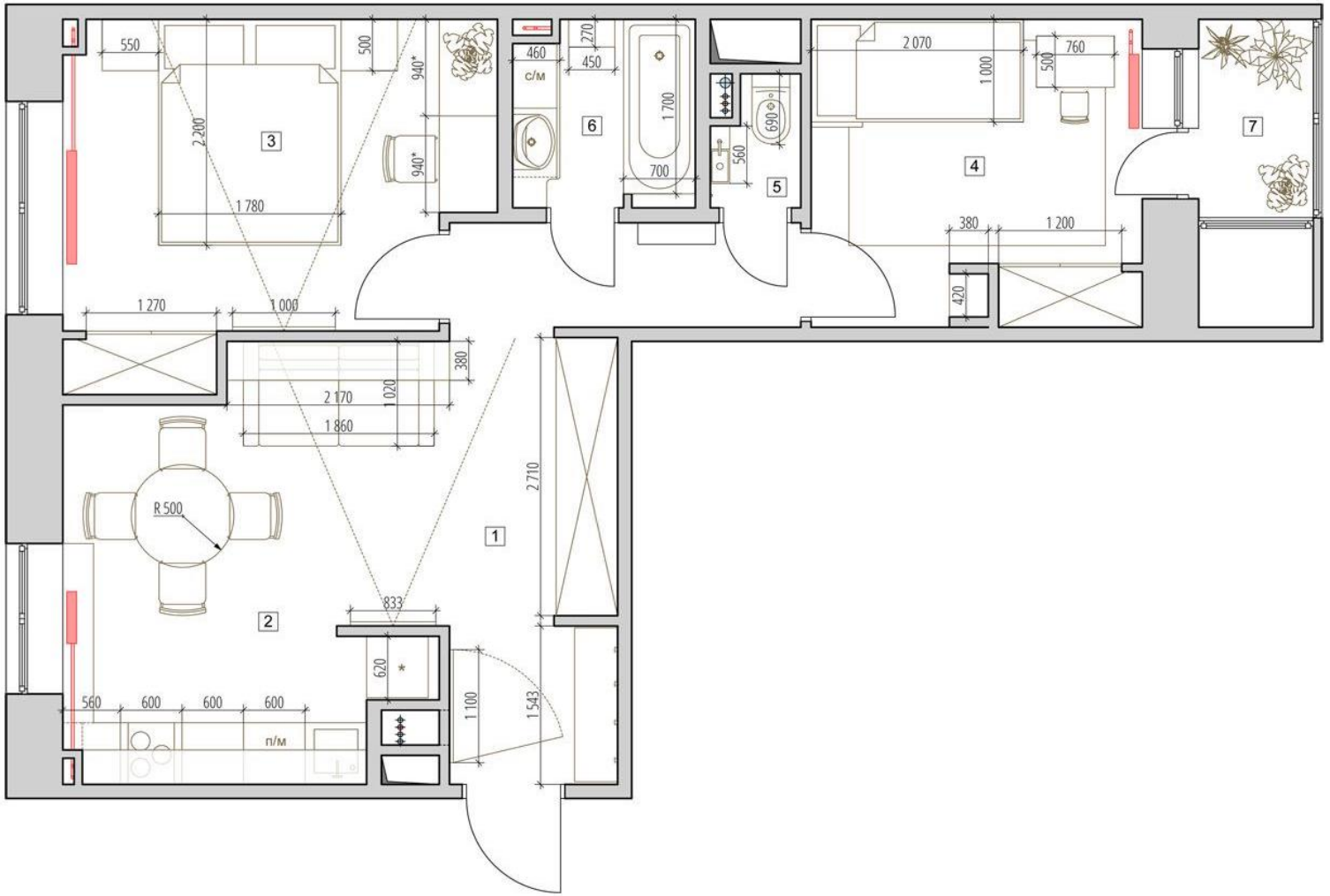
Архітектура Web Application

План уроку

- Основна теорія
- Монолітна архітектура
- Мікросервісна архітектура

Архітектура Web Application

Архітектура квартири

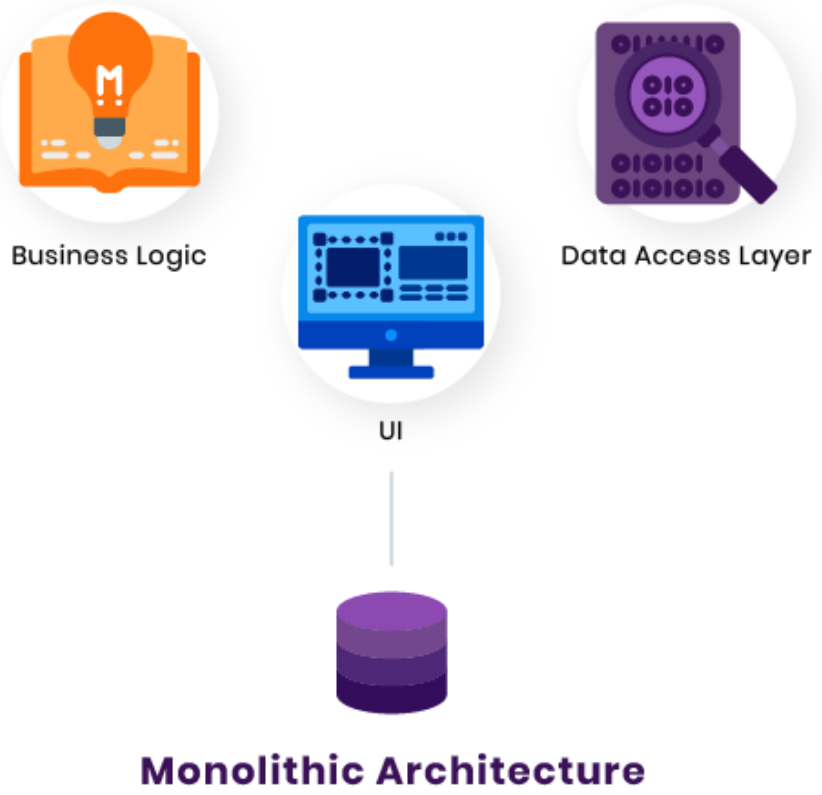


Архітектура ПО

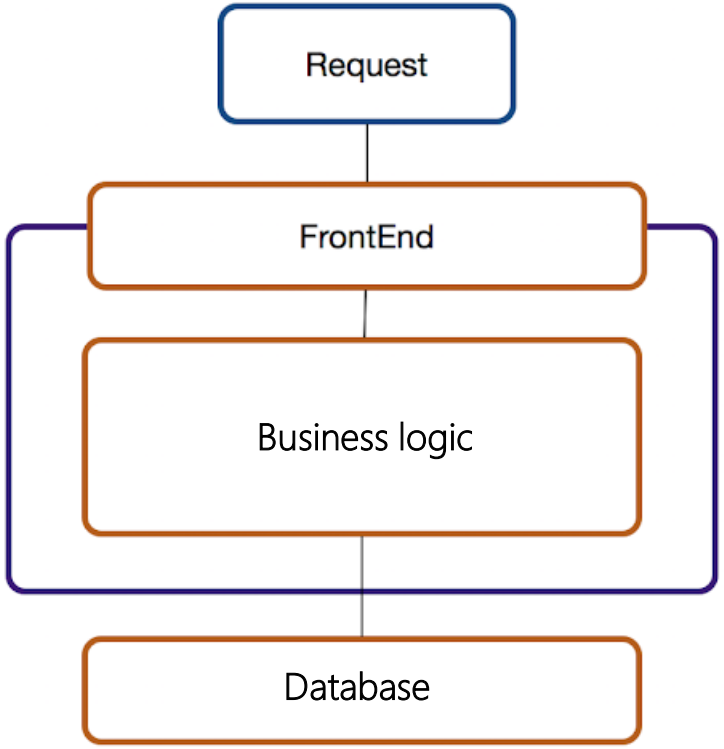
Архітектура програмного забезпечення – це організаційний дизайн всього програмного застосунку, включаючи всі підкомпоненти та зовнішні застосунки.

Монолітна архітектура

Монолітна архітектура



Монолітна архітектура



Монолітна архітектура

Переваги монолітної архітектури:

- Спрощена розробка та розгортання
- Менше проблем сумісності
- Найкраща продуктивність

Недоліки монолітної архітектури:

- Зі зростанням кодової бази зростає складність:
  - Внесення змін
  - Входження до проекту
- Обмежена гнучкість:
  - У технологіях
  - Архітектурні підходи
  - Масштабованості

Монолітна архітектура для тестування

Переваги:

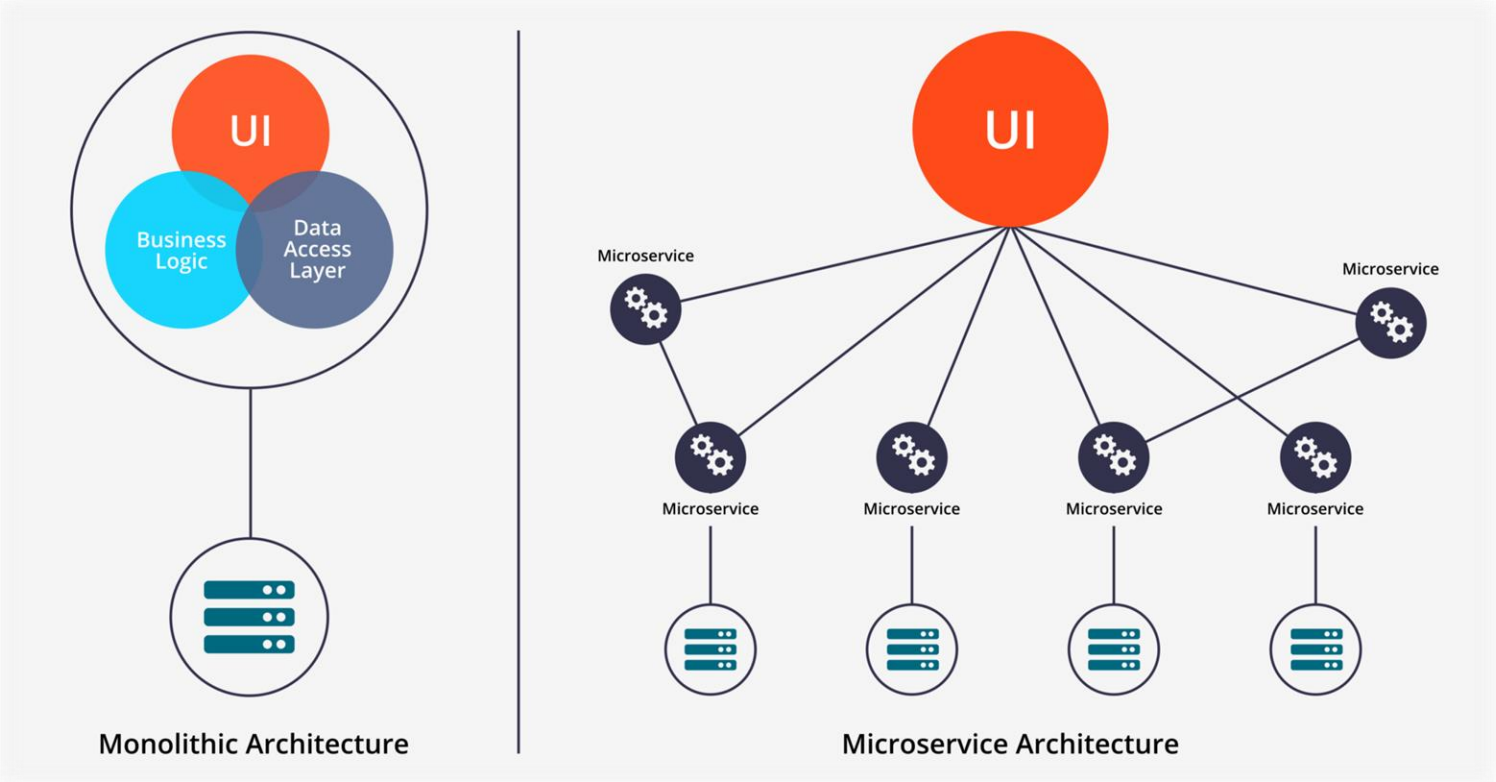
- Легко стежити та тестувати різні версії
- Легко створювати та стежити за тестовими даними

Недоліки:

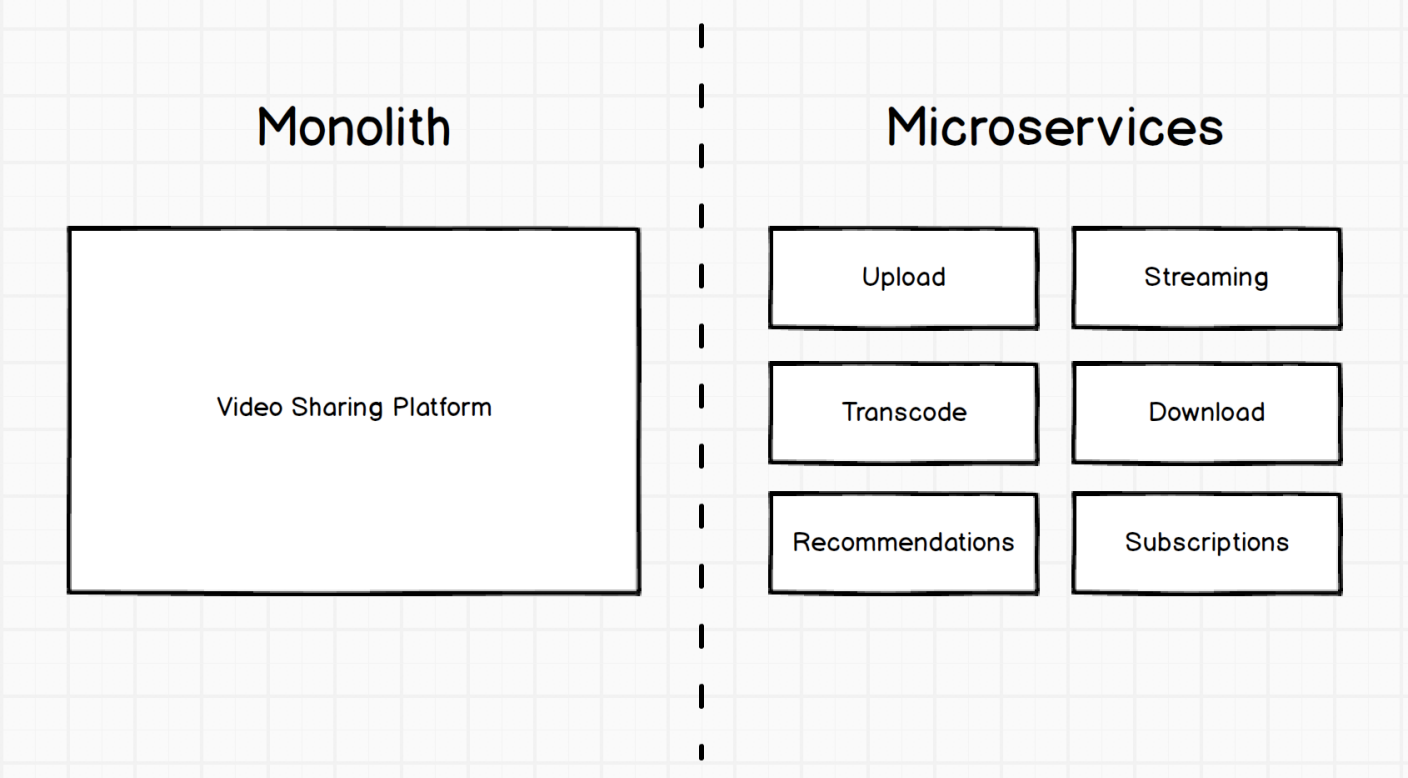
- Лише одна точка входу – UI
- Обмежує доступні види тестування
- Ускладнює вирахування тестового покриття
- Для тестування будь-якої частини програми потрібна вся програма

Мікросервісна архітектура

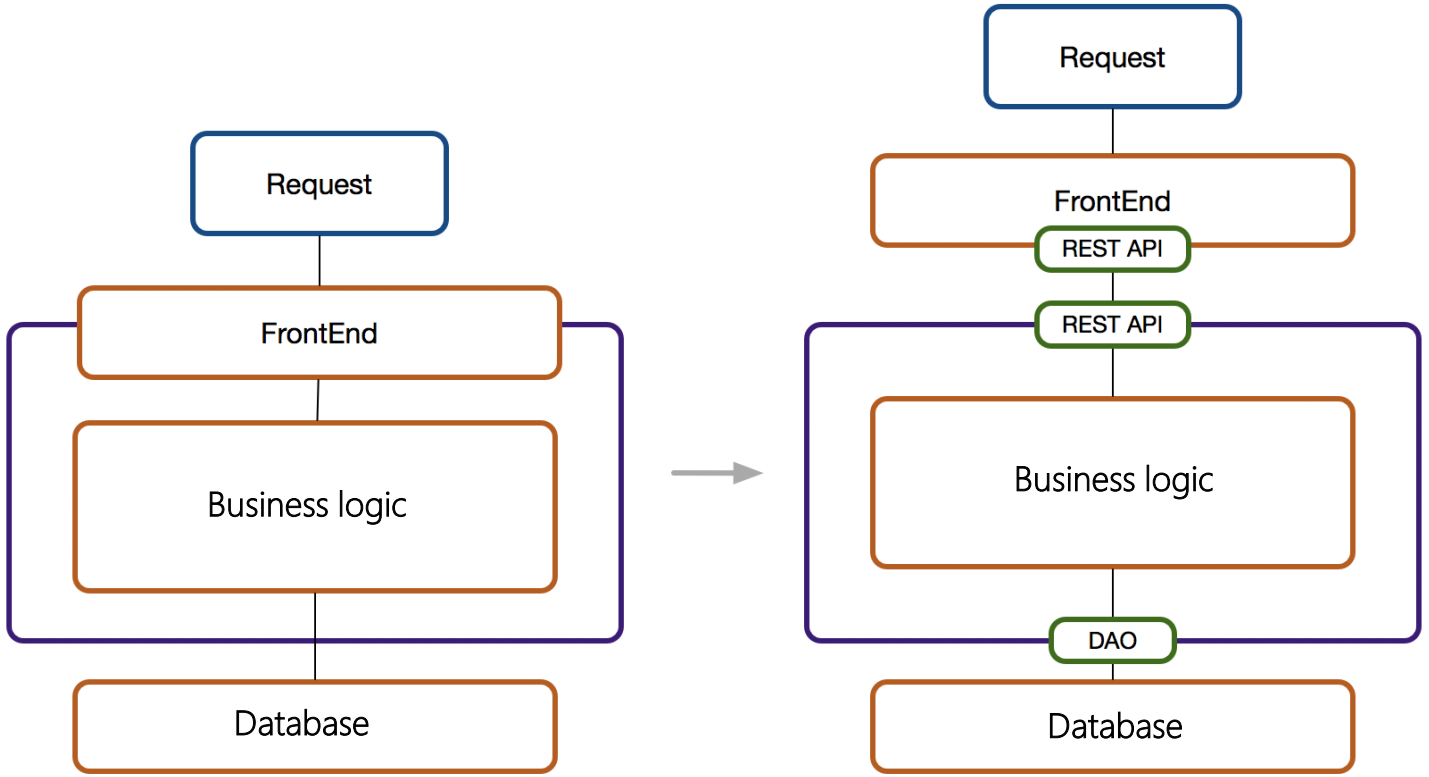
Мікросервіси та Моноліт



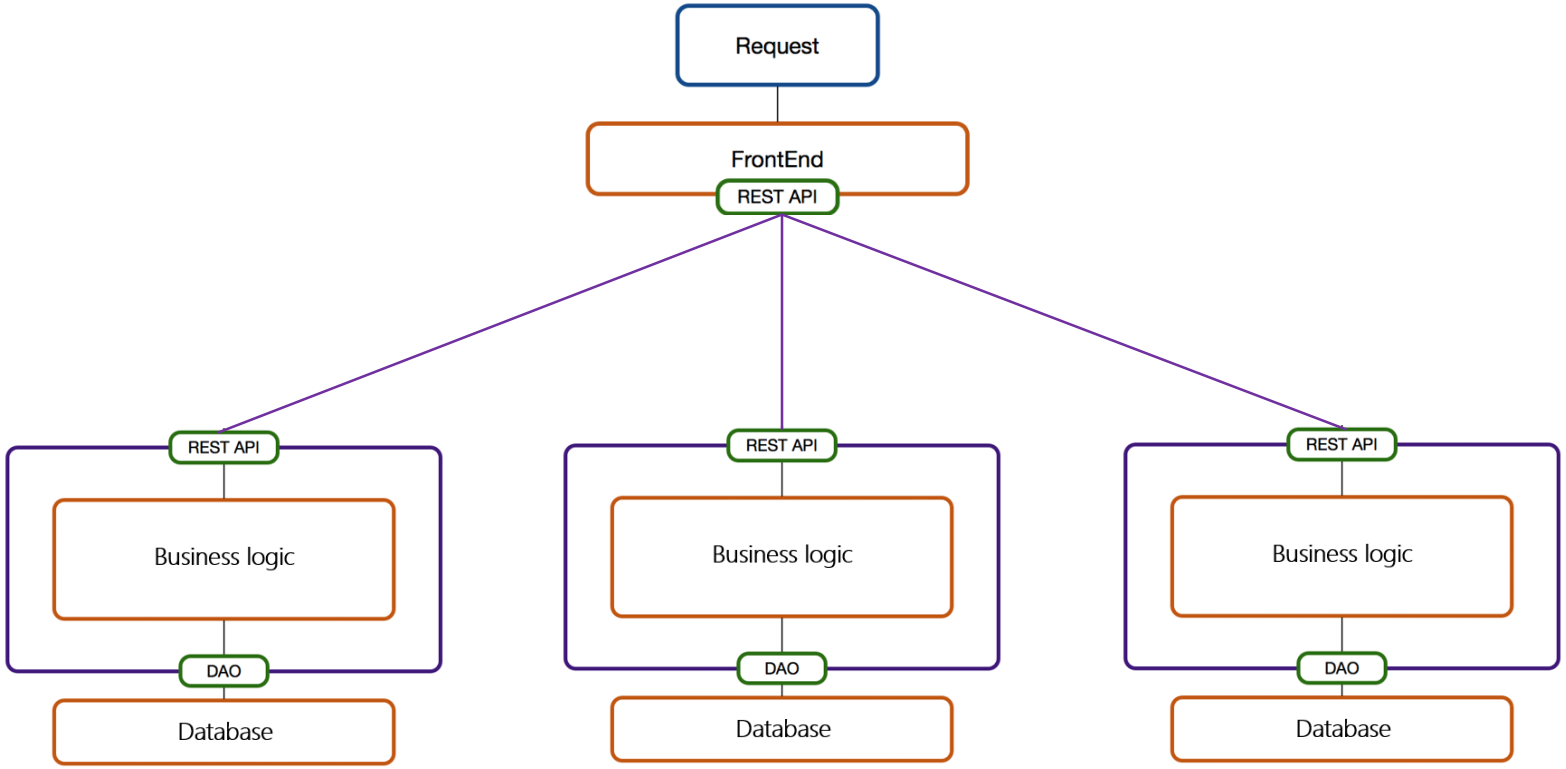
Мікросервісна архітектура на прикладі



Мікросервісна архітектура у деталях



Мікросервісна архітектура у деталях



Мікросервісна архітектура

Переваги архітектури:

- Висока відмовостійкість
- Гнучкість у технологіях та підходах
- Масштабованість
- Легка зміна коду будь-якого сервісу
- Легкість виведення написаного коду до робіту

Недоліки архітектури:

- Повідомлення між самими сервісами складне
- Накладні витрати на комунікації між компонентами
- Складність організації даних між сервісами
- Складне керування інфраструктурою – деплой, версіонування
- Важко організувати безпеку
- Підвищує необхідний рівень тех. компетенцій команди та керівництва

Мікросервісна архітектура для тестування

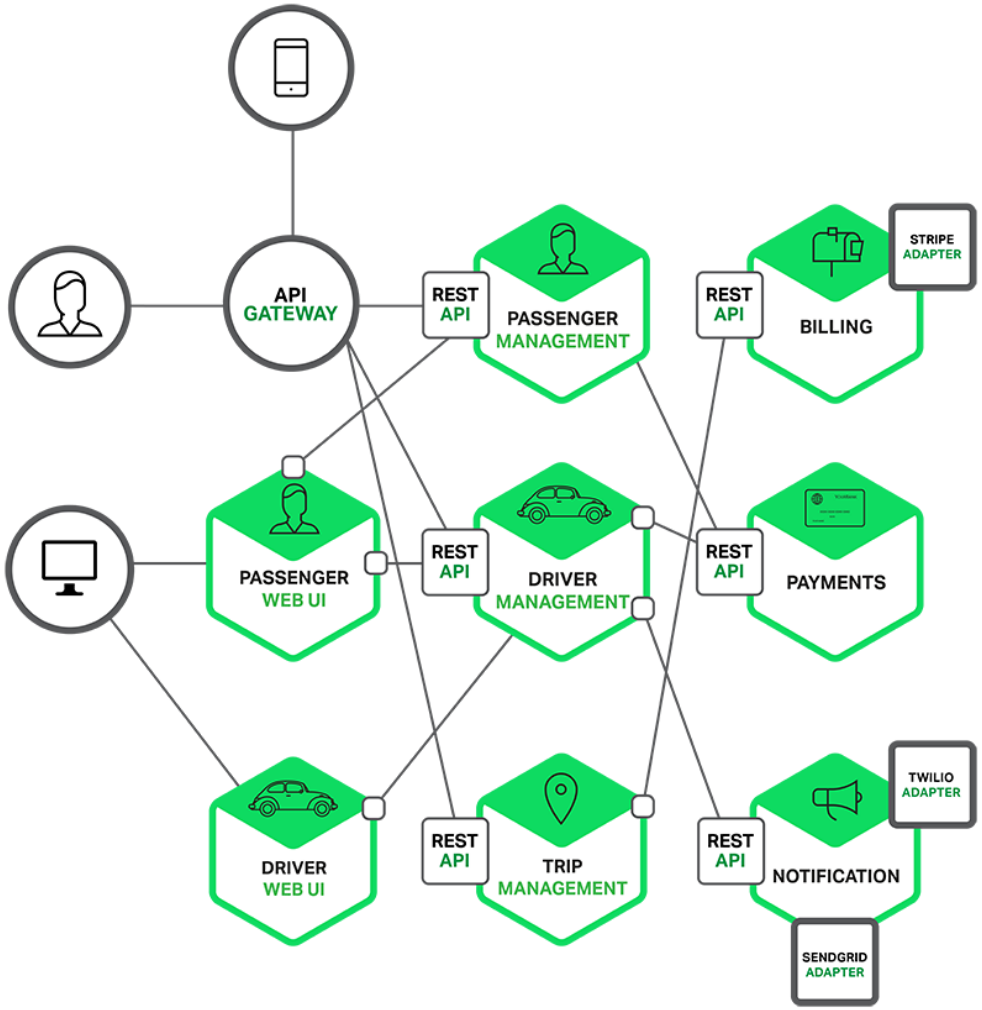
Переваги:

- Можливості прямого тестування будь-якого компонента та їх зв'язок
- Тестування будь-якого компонента можна організувати незалежно
- API дозволяє оптимізувати тестове покриття

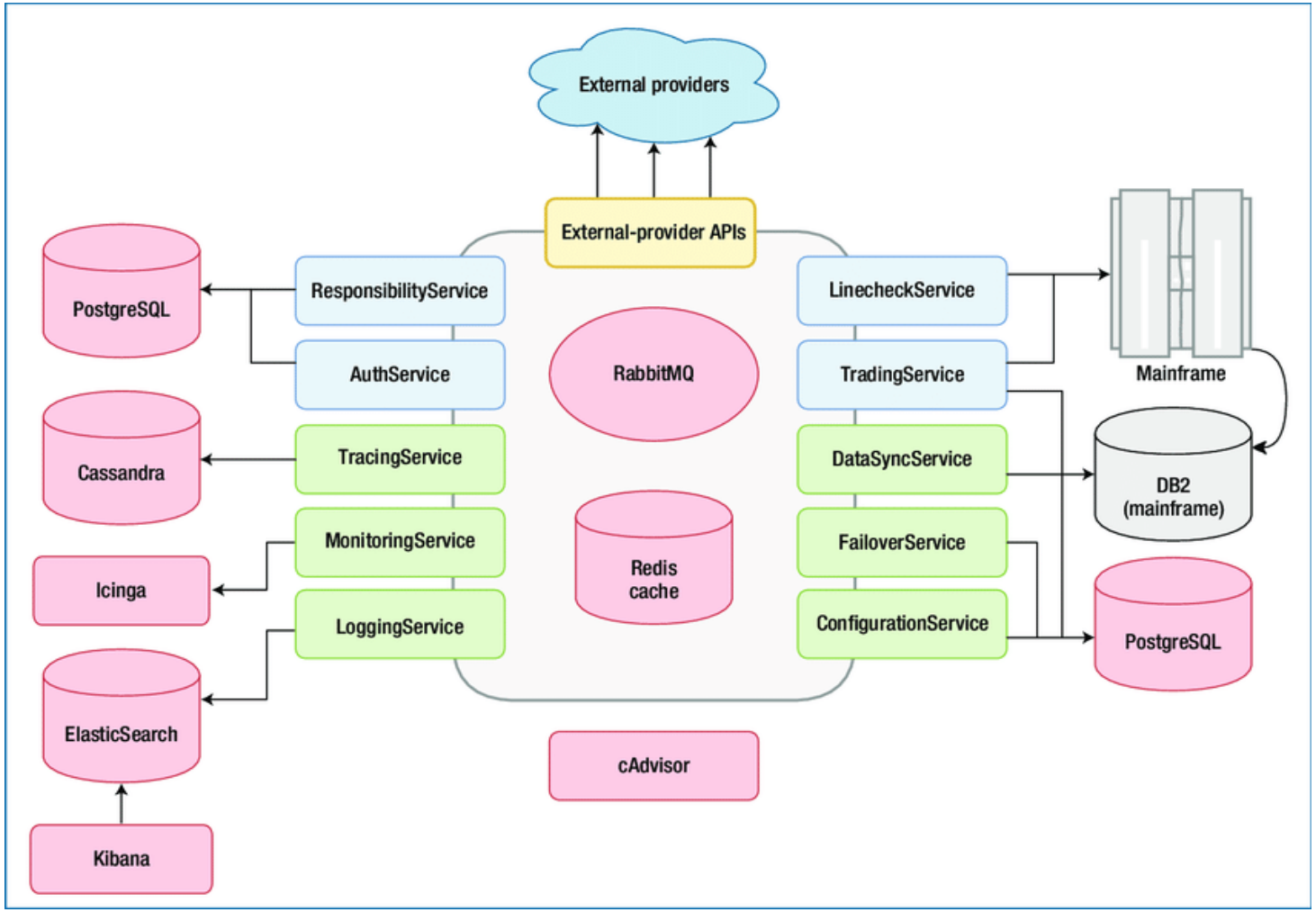
Недоліки:

- Важко тестувати дані та особливо транзакції
- Важко знайти першопричину помилки
- Необхідність синхронізації проходу тестів з деплом усіх компонентів, що беруть участь.
- Складність тестування зростає з різноманітністю інтерфейсів сервісів
- Високі вимоги до планування тестування та QA менеджменту

Мікросервісна архітектура. Приклад



Мікросервісна архітектура. Приклад



Архітектура Web Application. Висновки

Моноліт:

- Легший як у розробці так і в тестуванні
- Обмежує види тестування
- Можливі області, які неможливо тестувати

Мікросервіси:

- Збільшують складність розробки та тестування надаючи великі можливості
- Більш вимогливі до технічного рівня команди та менеджменту

Що ми сьогодні вивчали

План уроку

- Основна теорія
- Монолітна архітектура
- Мікросервісна архітектура

Підсумки

Що одне, найголовніше, ви дізнались сьогодні?

Дякую за увагу! До нових зустрічей!

TestProvider Pearson CyberBionic systematics Microsoft Partner Windows Azure EBA