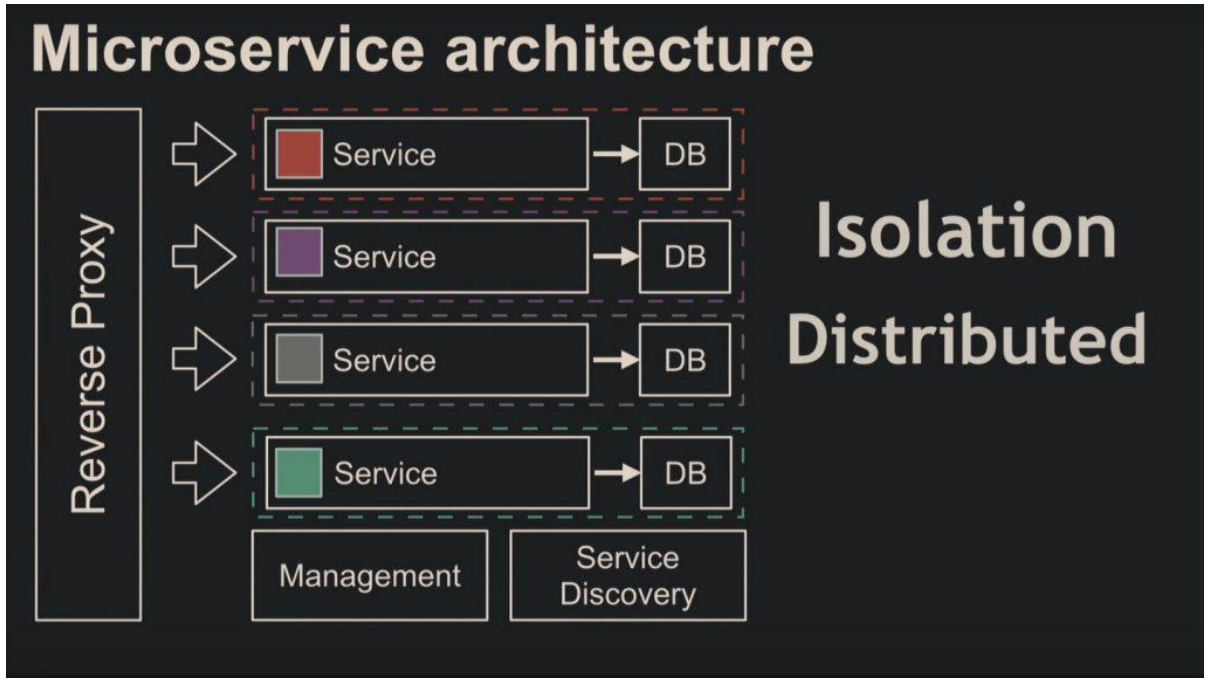
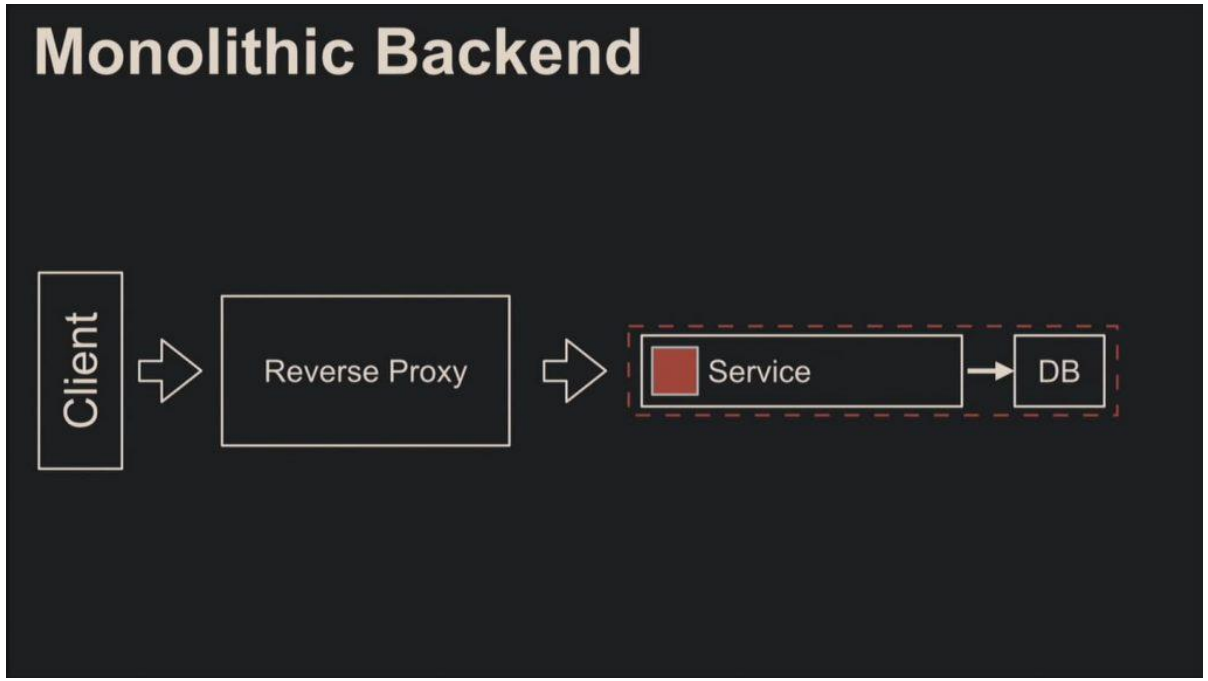
Мікросервісна архітектура MSA

MSA

MSA - принципова організація розподіленої системи на основі мікросервісів і їх взаємодії один з одним і з середовищем по мережі, а також принципів, що направляють проектування архітектури, її створення та еволюцію





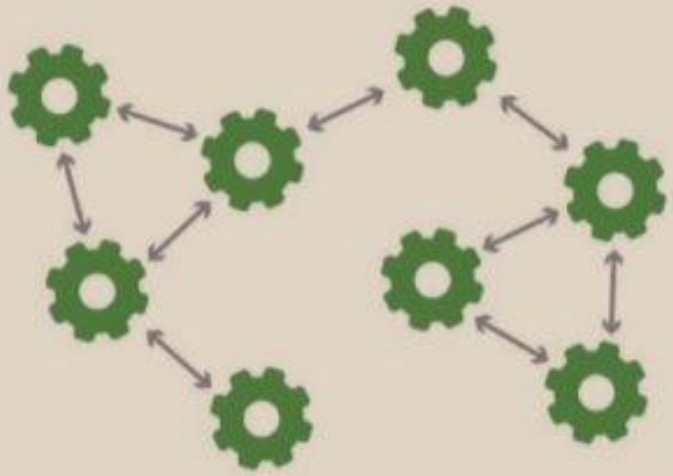
SOA vs MSA

2000's

SERVICE ORIENTED ARCHITECTURE

2010's

MICROSERVICES ARCHITECTURE



Признаки мікросервісу

- Мікросервіс відповідає за одну бізнес функцію(можливість).
- Мікросервіси можна розгортати окремо.
- Мікросервіс складається з одного або декількох процесів.
- У мікросервіса є власне сховище даних.
- Невелика команда розробників може супроводжувати декілька мікросервісов.
- Мікросервіс можна легко замінити.

Мікросервіс відповідає за одну бізнес функцію.

Мікросервіс відповідає за одну і тільки одну функціональну можливість з усією системи. Розділимо це твердження на дві складові частини.

- ❑ На мікросервіс покладається рівно один обов'язок.
- ❑ Цей обов'язок полягає в реалізації окремої можливості(функції).

Таким чином, мікросервіс повинен змінюватися тільки при зміні цієї можливості. Більш того, ви повинні постаратися реалізувати в мікросервісе цю можливість максимально повно, щоб при її зміні потрібно було міняти тільки один мікросервіс.

Існує два види можливостей в системі мікросервісов.

- ❑ Бізнес-можливість –дія, що виконується системой, яка сприяє реалізації призначення системи. Наприклад, це може бути відстеження кошиків замовлень користувачів або розрахунок цін.
- ❑ Технічна можливість - щось необхідне для кількох інших мікросервісов, наприклад для інтеграції з якоюсь сторонньою системою. Технічні можливості не можна назвати головними рушійними силами при розбитті системи на мікросервіси, вони виділяються лише в разі, коли виявляється, що кільком мікросервісам для їх бізнес-можливостей потрібна однакова технічна можливість.

Мікросервіси можна розгортати окремо

Щоб мати можливість розгортати окремий мікросервіс в той момент, коли інша система продовжує працювати, процес зборки повинен бути налаштований з огляду на таке.

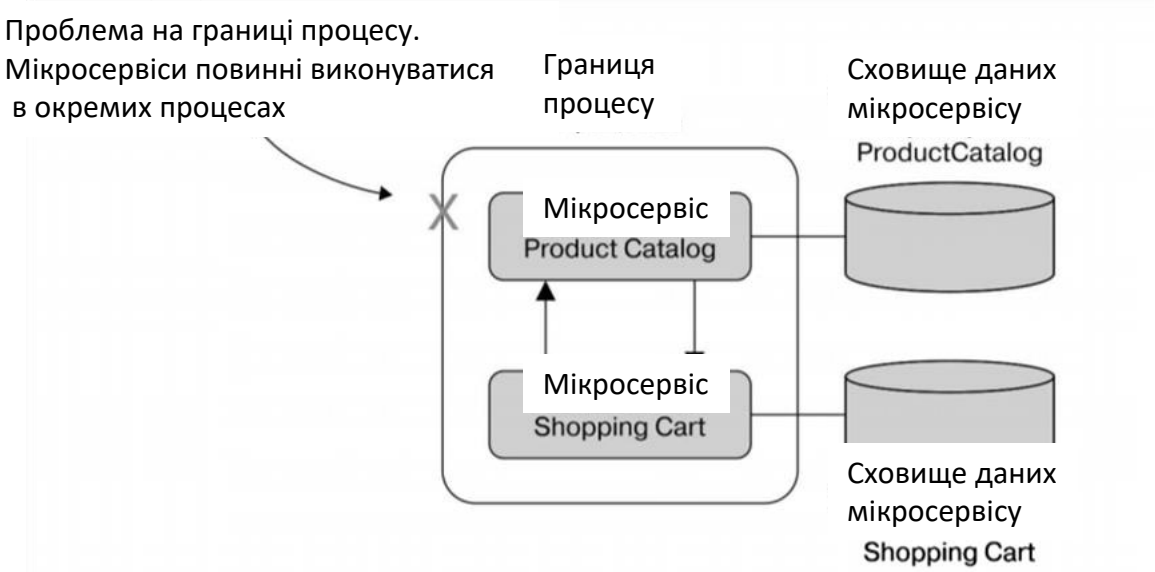
❑ Усі мікросервіси повинні бути вбудовані в окремі артефакти або пакети(компоненти).

❑ Процес розгортання повинен бути налаштований таким чином, щоб забезпечити підтримку індивідуального розгортання при продовженні функціонування інших мікросервісов. Наприклад, можна використовувати процес плаваючого розгортання, при якому мікросервіс розгортається на одному сервері за один раз, щоб знизити час простою.

Мікросервіс складається з одного або декількох процесів.

Мікросервіс повинен виконуватися в окремому процесі або окремих процесах, щоб залишатися якомога більш незалежним від інших мікросервісів тієї ж системи, а також щоб зберігати можливість окремого розгортання. Розбивши це положення на складові частини, отримуємо два пункти.

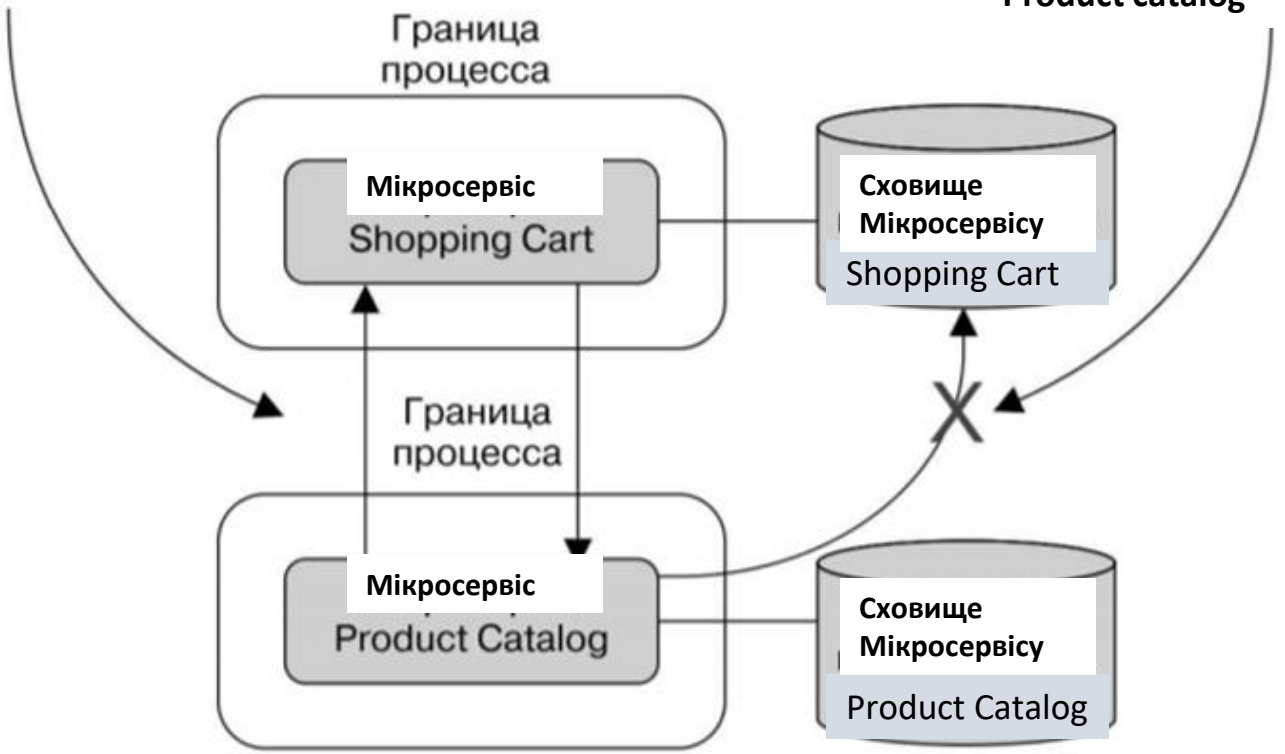
- ❑ Усі мікросервіси повинні працювати в окремих від інших мікросервісів процесах.
- ❑ У будь-якого мікросервіса може бути більш одного процесу.



У мікросервіса є власне сховище даних.

Уся взаємодія з мікросервісом Product catalog повинна відбуватися за допомогою загально доступних API

Пряме звернення до сховища мікросервісу Product catalog заборонено. Власник сховища Product catalog - мікросервіс Product catalog



Невелика команда розробників може супроводжувати декілька мікросервісов

- Один сервіс може розвивати одна команда не більше ніж з дюжини чоловік.
- Команда з півдюжини чоловік може розвивати півдюжини сервісів.
- Контекст (не тільки бізнесу, а й розробки) одного сервісу поміщається в голові однієї людини.
- Один сервіс може бути повністю переписаний однією командою за одну Agile-ітерацію

Мікросервіс можна легко замінити.

- Замінність мікросервіса означає можливість переписати його з нуля за розумний час. Іншими словами, що супроводжує мікросервіс команда повинна бути спосіб на замінити поточну реалізацію абсолютно нової, причому зробити це в звичайному робочому режимі. Цей відмінна ознака накладає ще одне обмеження на розмір мікросервіса: заміна занадто великого мікросервіса обійдеться не дешево, а замінити маленький цілком реально.

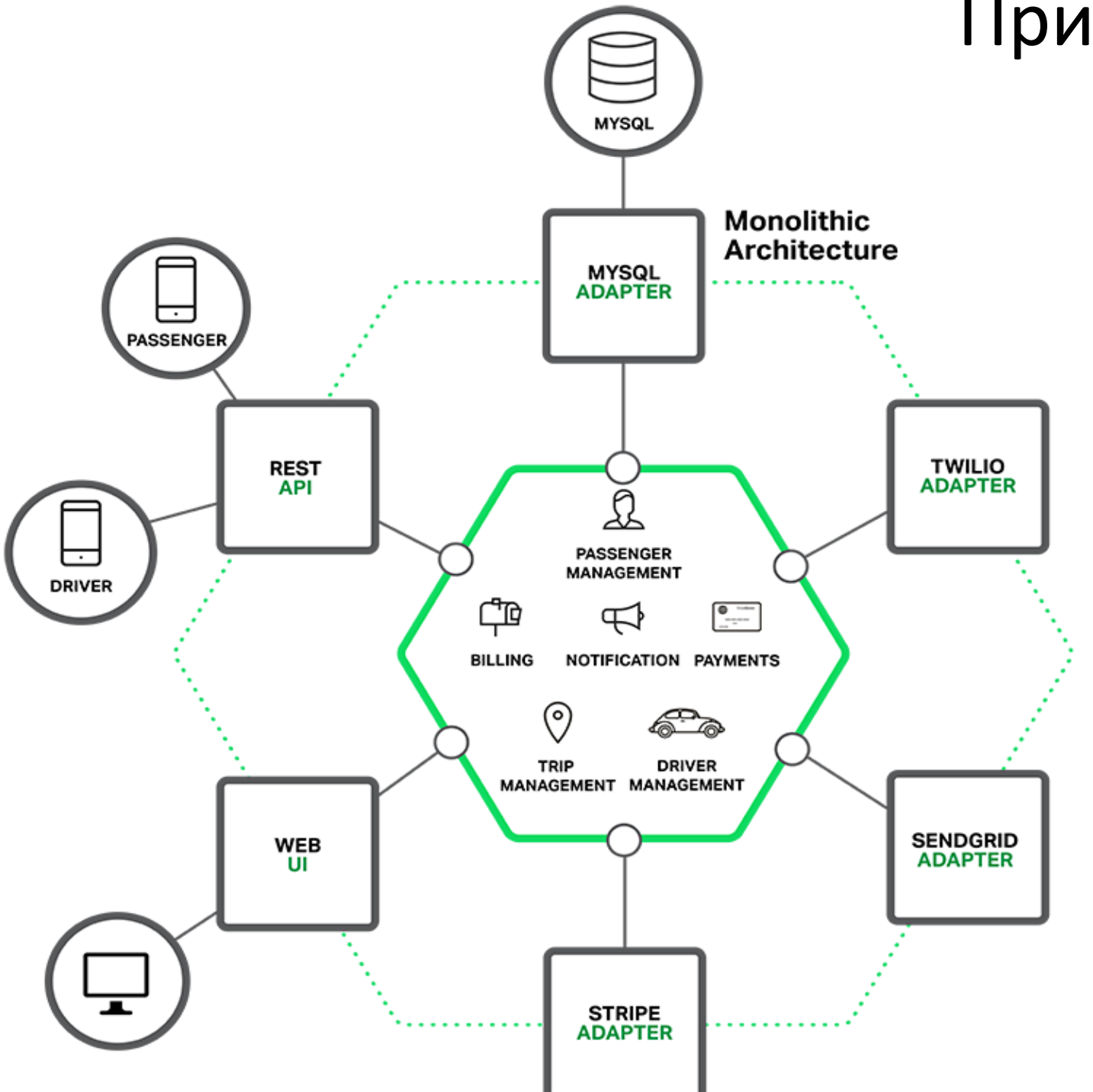
Переваги мікросервісів

- ❑ забезпечують можливість безперервної доставки ПЗ;
- ❑ забезпечують ефективність процесу розробки в силу надзвичайної легкості їх супроводу;
- ❑ спочатку виконані стійкими до помилок;
- ❑ масштабуються в бік збільшення або зменшення незалежно один від одного.

Недоліки мікросервісів

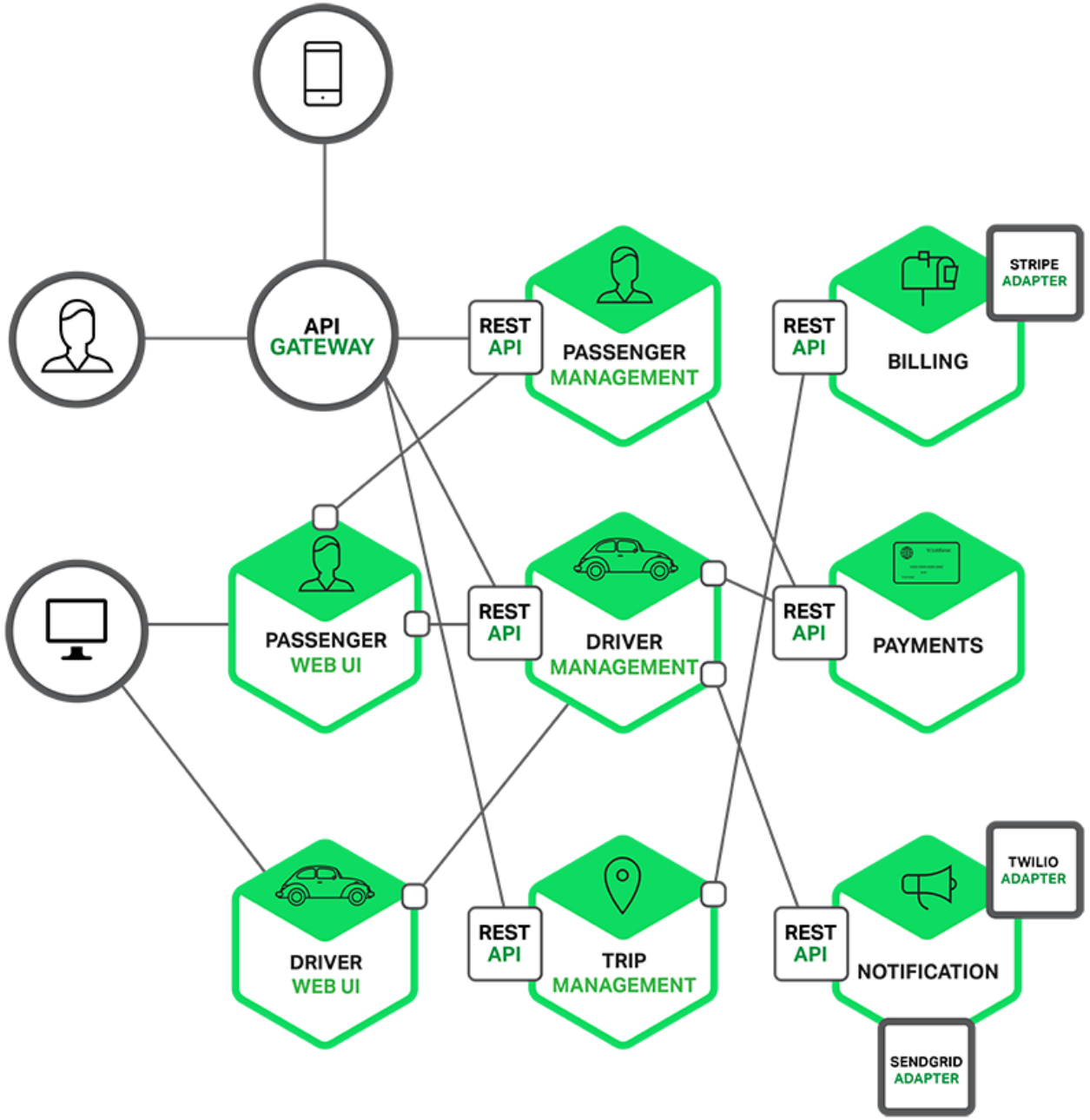
- Системи мікросервісов - розподілені системи. Пов'язані з вибором розподіленої архітектури витрати добре відомі. Такі системи складніше для осмислення і тестування, ніж монолітні, і обмін повідомленнями між процесами або відвідай відбувається на порядки повільніше, ніж виклики методів всередині процесу.
- Системи мікросервісов складаються з безлічі мікросервісов, і необхідно розробити і розгорнути кожен з них, а також забезпечити управління ним при експлуатації у виробничому середовищі. Це тягне за собою велику кількість розгортання і складну схему установки на виробництві.
- У кожного мікросервіса - власна база коду. Отже, рефакторинг з переміщенням коду з одного мікросервіса в інший вимагає великих зусиль. Необхідно заздалегідь подбати про правильне визначення області дії всіх мікросервісов.

Приклад

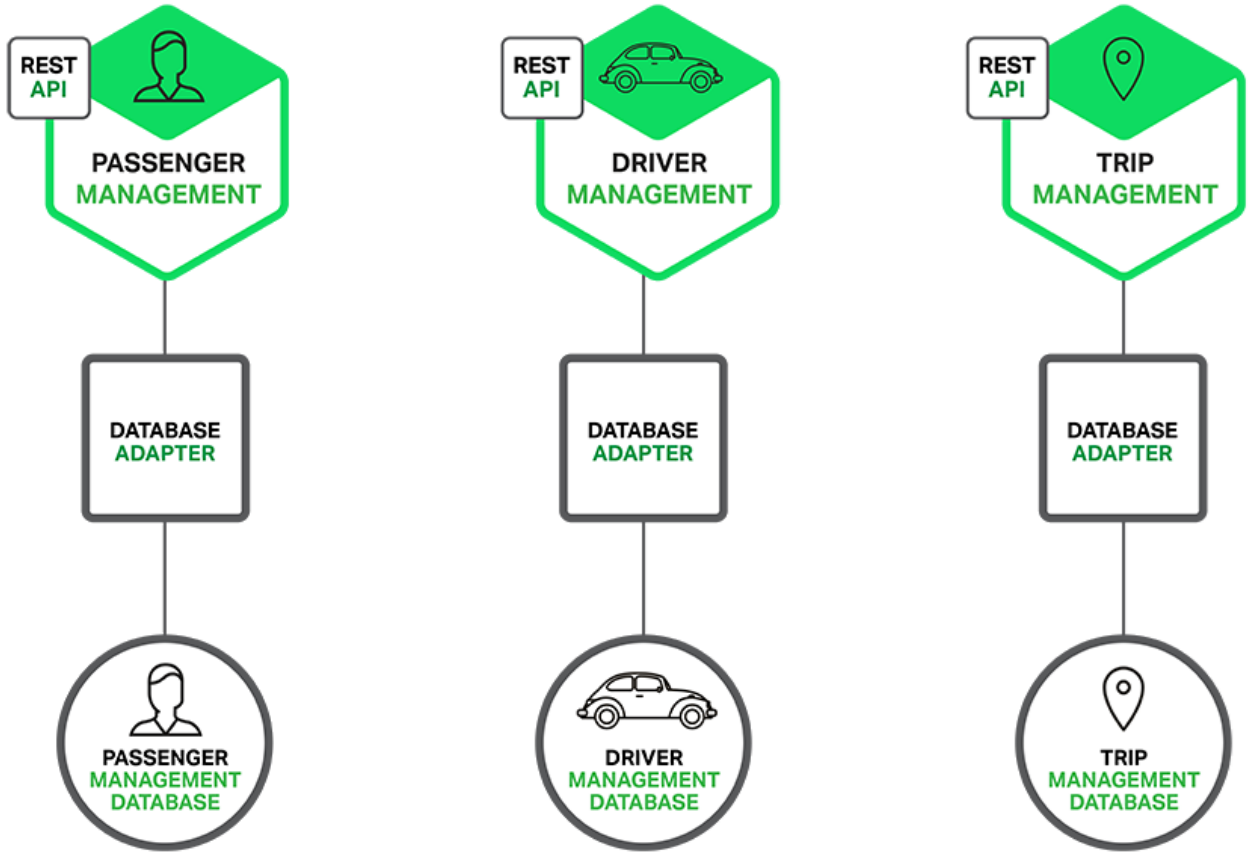


Необхідно написати абсолютно новий додаток для виклику таксі, яке буде конкурувати я такими додатками як Uber. Після кількох попередніх зустрічей та збору вимог, ви можете створити новий проект, або вручну, або за допомогою генератора, який поставляється з Rails, Spring Boot, Play, або Maven. Це новий додаток буде мати модульну гексагональну архітектуру

- В основі програми є бізнес-логіка, яка реалізується за допомогою додаткових модулів, які визначають послуги, доменних об'єктів і подій. Навколо сердечника адаптери, які взаємодіють із зовнішнім світом. Приклади адаптерів включають в себе компоненти доступу до бази даних, компоненти обміну повідомленнями, які виробляють і споживають повідомлення, а також веб-компоненти, які або виставляють API, або реалізують користувальницький інтерфейс.
- Незважаючи на наявність логічно модульну архітектуру, додаток запакований і розгортається як моноліт. Фактичний формат залежить від мови додатку і рамок.
- Програми, написані в цьому стилі надзвичайно поширені. Вони прості в розробці. Такі додатки також просто перевірити. Ви можете реалізувати від початку до кінця тестування просто запустити додаток і тестувати призначеного для користувача інтерфейсу з Selenium. Монолітні додатки також прості в розгортанні. Ви просто повинні скопіювати упакований додаток на сервер. Ви можете також масштабувати додаток, запустивши декілька копій. На ранніх стадіях проекту він працює добре.

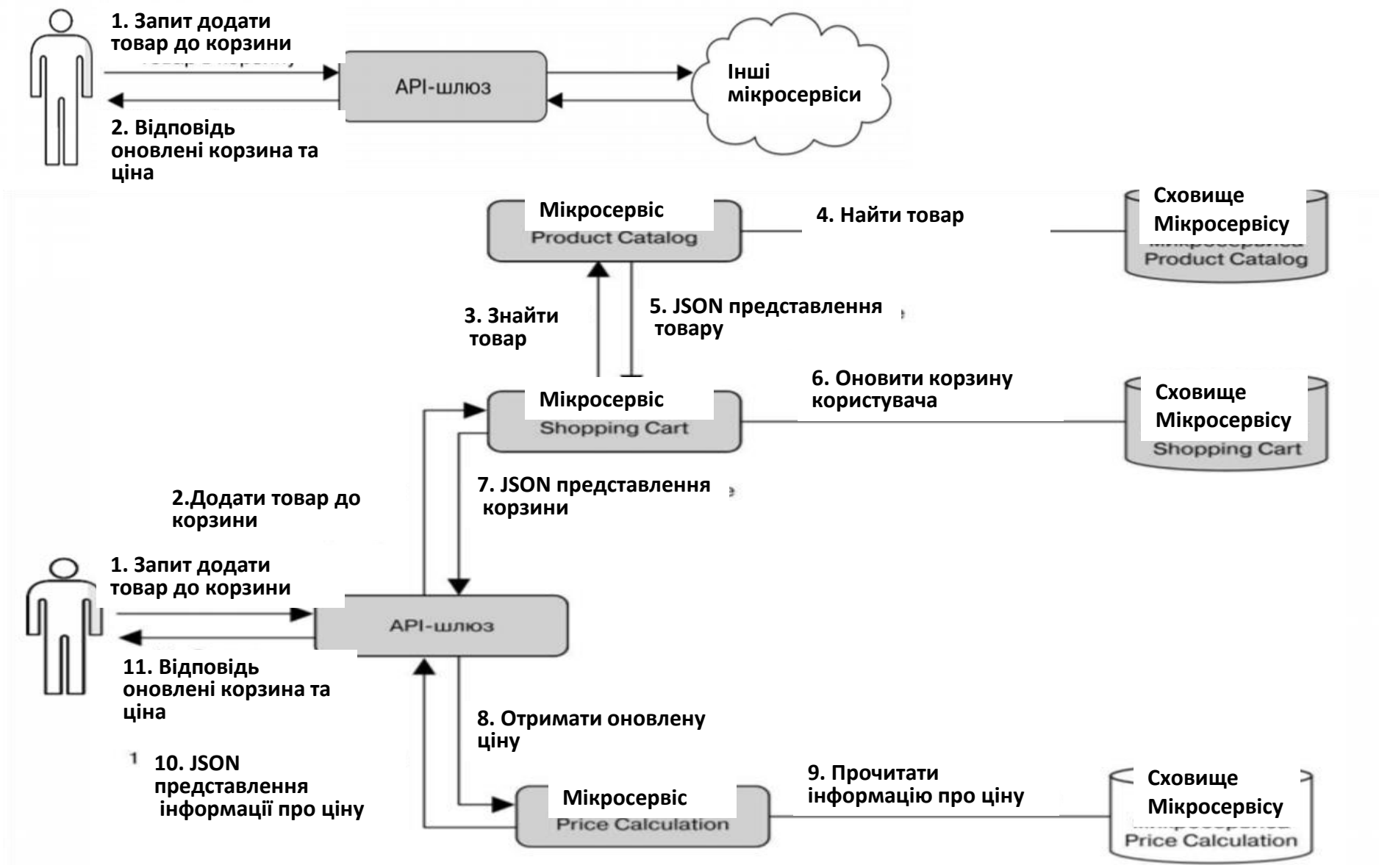


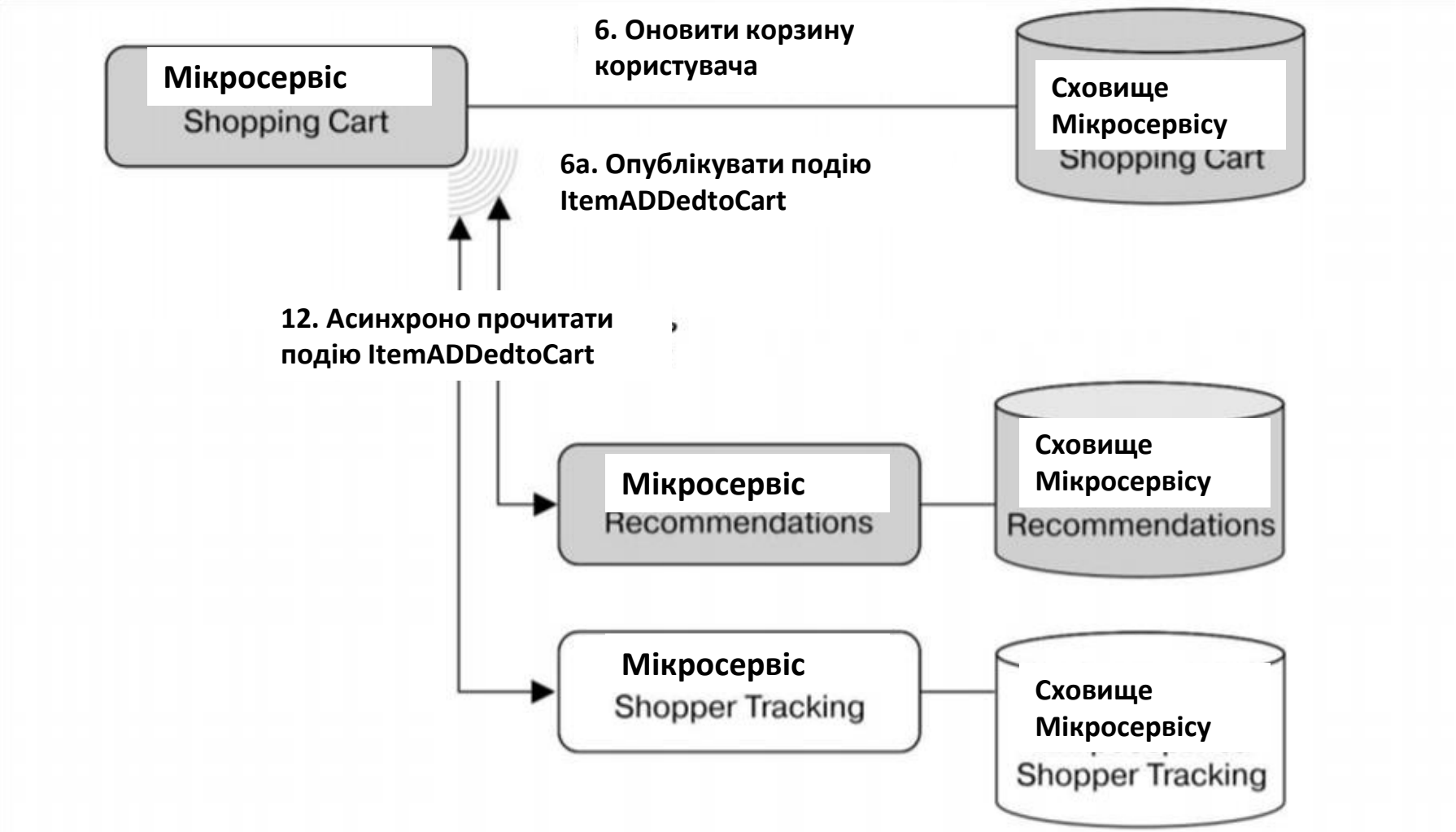
Сервіс, як правило, реалізує безліч різних функцій або функціональних можливостей, таких як управління замовленнями, управління клієнтами і т.д. Кожен мікросервіс є міні-додатком, який має свою власну гексагональную архітектуру, що складається з бізнес-логіки, а також різних перехідників.



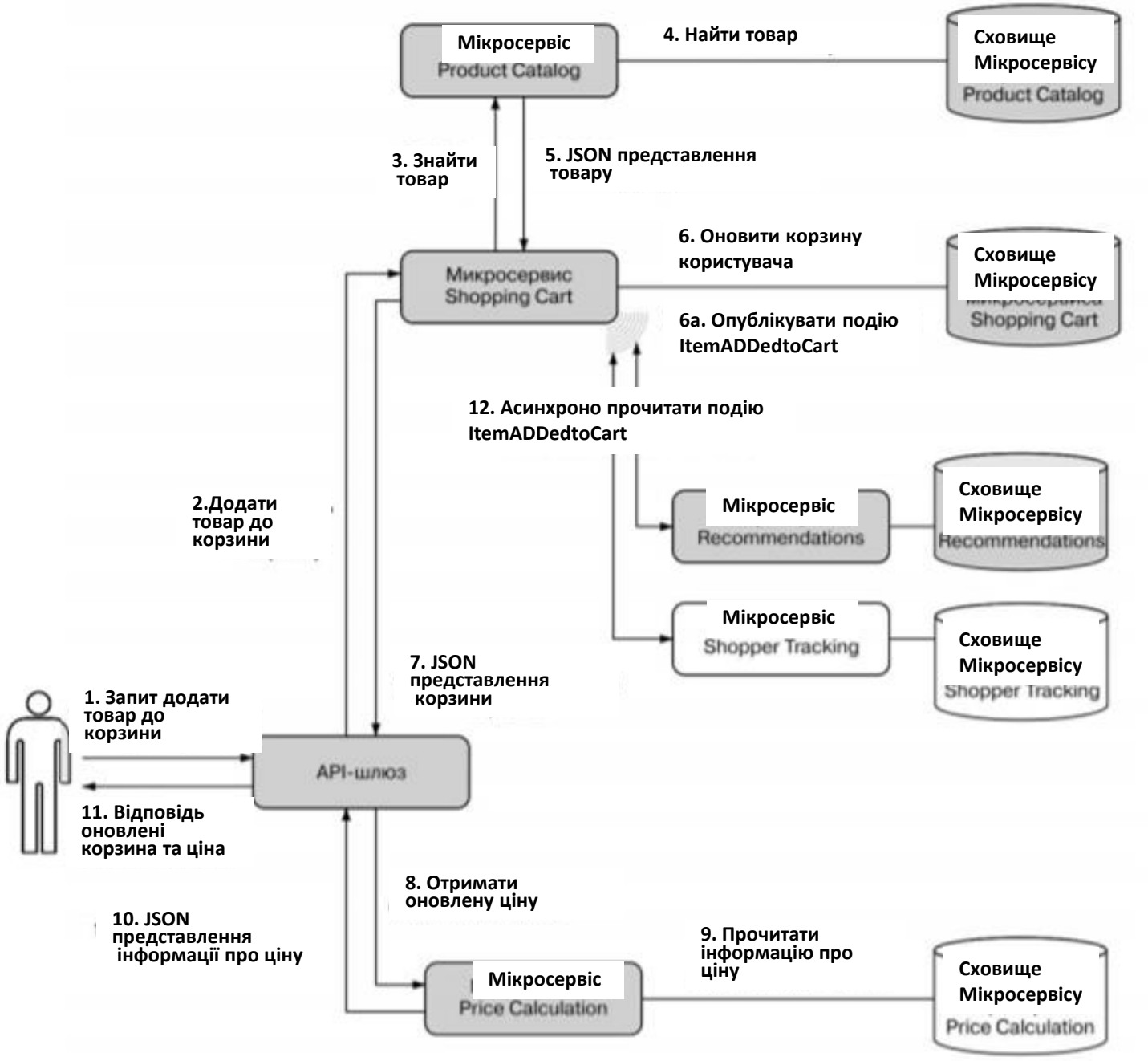
Кожен із сервісів має свою власну базу даних. Крім того, сервіс може використовувати тип бази даних, яка найкраще підходить для його потреб. Наприклад, сервіс водія, який знаходить пропозиції, близькі до потенційного пасажиру, необхідно використовувати базу даних, яка підтримує ефективні гео-запити.

Приклад. Відвідувач інтернет-магазину додає товар в корзину замовлень.

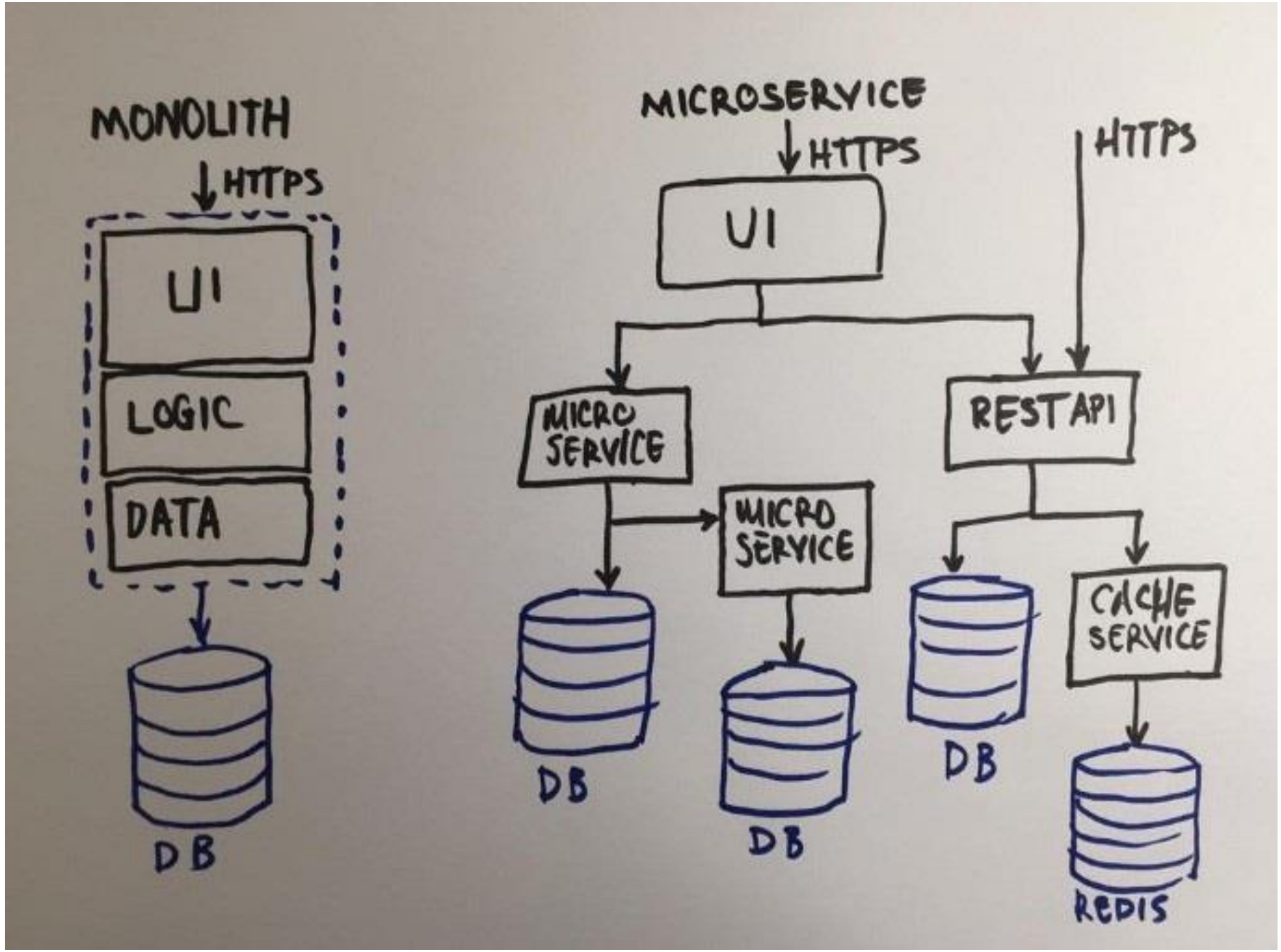


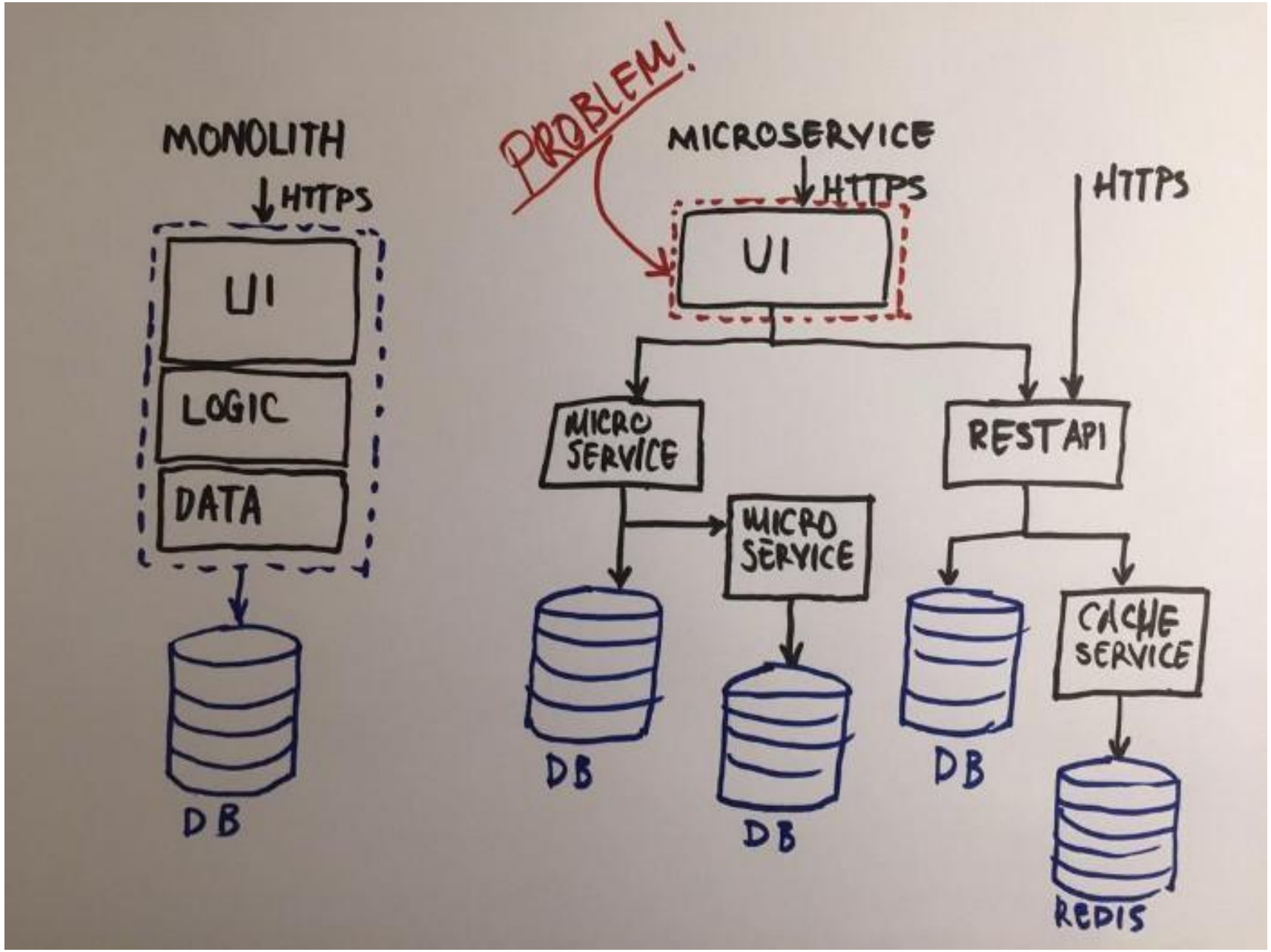


Побічні дії запитів користувача

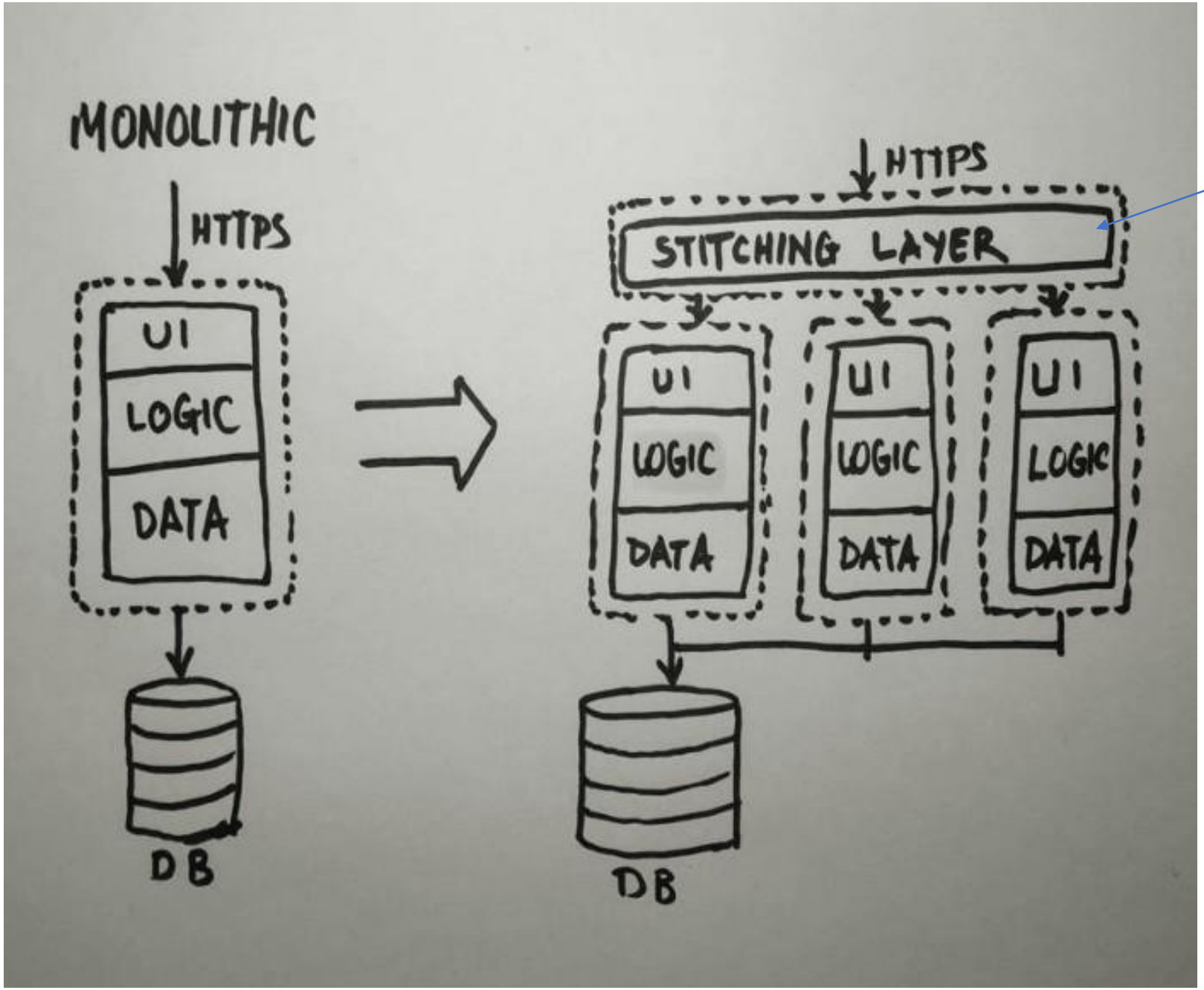

Мікросервіс
Product Catalog
4. Найти товар
Сховище Мікросервісу
Product Catalog
3. Знайти товар
5. JSON представлення товару
Микросервис
Shopping Cart
6. Оновити корзину користувача
Сховище Мікросервісу
Shopping Cart
6a. Опублікувати подію ItemADDedtoCart
12. Асинхроно прочитати подію ItemADDedtoCart
2.Додати товар до корзини
Мікросервіс
Recommendations
Сховище Мікросервісу
Recommendations
Мікросервіс
Shopper Tracking
Сховище Мікросервісу
Shopper Tracking
7. JSON представлення корзини
1. Запит додати товар до корзини
API-шлюз
11. Відповідь оновлені корзина та ціна
8. Отримати оновлену ціну
10. JSON представлення інформації про ціну
Мікросервіс
Price Calculation
9. Прочитати інформацію про ціну
Сховище Мікросервісу
Price Calculation

Мікро-фронтенди


MONOLITH
HTTPS
UI
LOGIC
DATA
DB
MICROSERVICE
HTTPS
UI
HTTPS
MICRO SERVICE
MICRO SERVICE
REST API
CACHE SERVICE
DB
DB
DB
REDIS



Принцип мікро-фронтендів: представлення веб-сайту або веб-додатку як набору функцій, за які відповідають незалежні команди. У кожної з команд є своя місія, своє поле роботи, на якому вона спеціалізується. Команда крос-функціональна і розробляє
весь цикл - від бази даних до призначеного для користувача інтерфейсу



Спеціальний додаток для взаємодії з користувачем та зв’язування мікро-додатків (мікросервісів)

Проблеми реалізації

Проблема №1: досягти цілісної і погодженої поведінки від UI, коли у нас кілька абсолютно автономних мікрозастосунків.

Проблема №2: переконатися, що одна команда не переписує CSS іншої команди

Проблема №3: зробити глобальну інформацію загальної для різних мікрозастосунків.

Проблема №4: якщо все мікрозастосунки автономні, як проводити маршрутизацію на стороні клієнта?

…

Проблема №N: оркеструвати сторону клієнта, щоб кожного разу не перезавантажувати сторінку

Сторона клієнта

- оркестрації
- маршрутизація
- ізоляція мікрозастосунків
- взаємодія додатків
- єдність UI мікрозастосунків

Сторона сервера

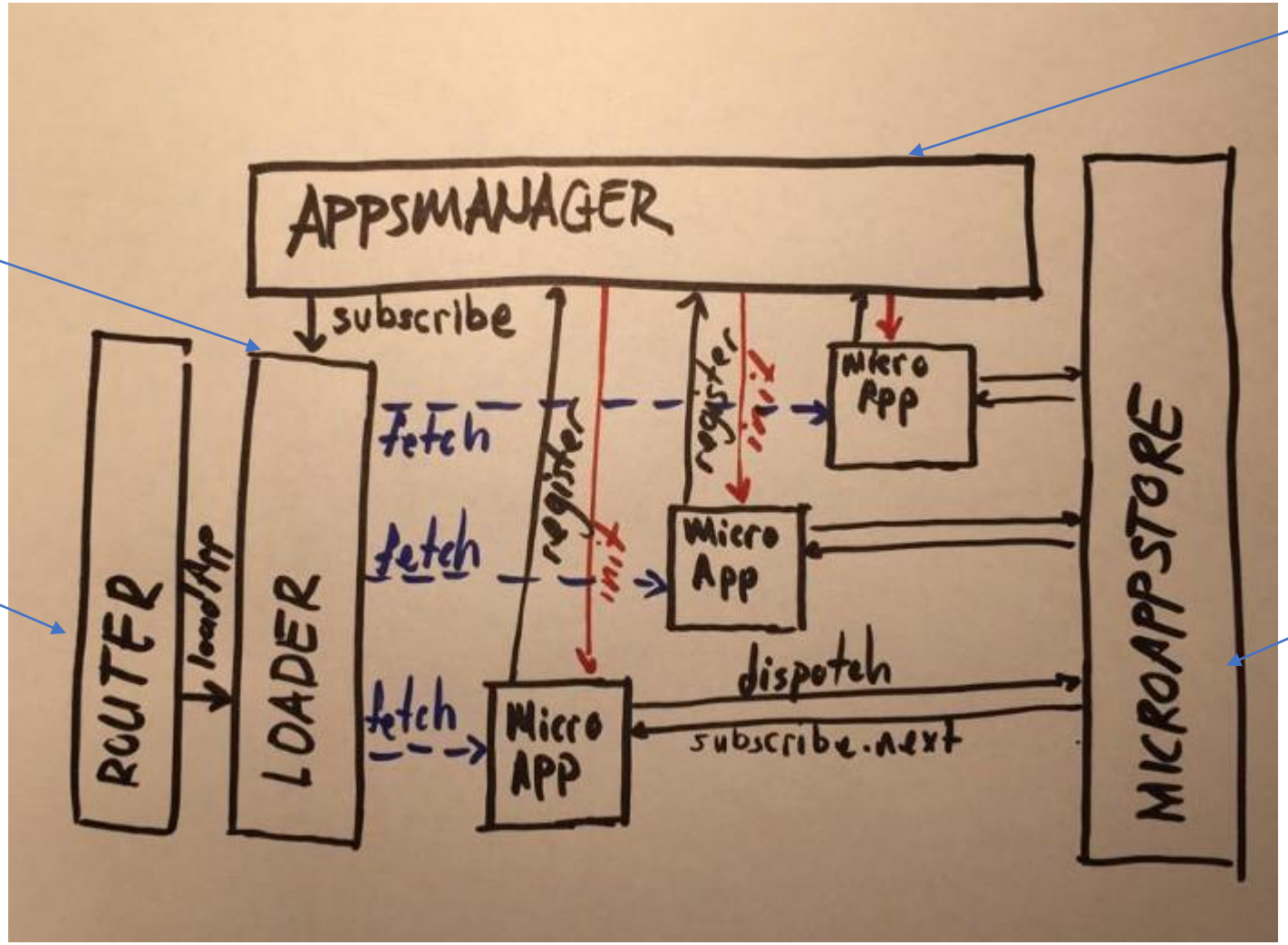
- серверний рендеринг
- маршрутизація
- управління залежностями

Сторона клієнту

AppsManager - ядро оркестрації мікро-додатків на стороні клієнта. Основне завдання AppsManager - створення дерева залежностей. Як тільки все залежності дозволені, AppsManager запускає мікрододаток.

Loader. Він відповідає за завантаження додатків для клієнтської сторони.

Router. Для виконання маршрутизації на стороні клієнта.



MicroAppStore. Для вирішення клієнтського взаємодії між мікрозастосунками.

Сторона серверу

- MicroAppServer

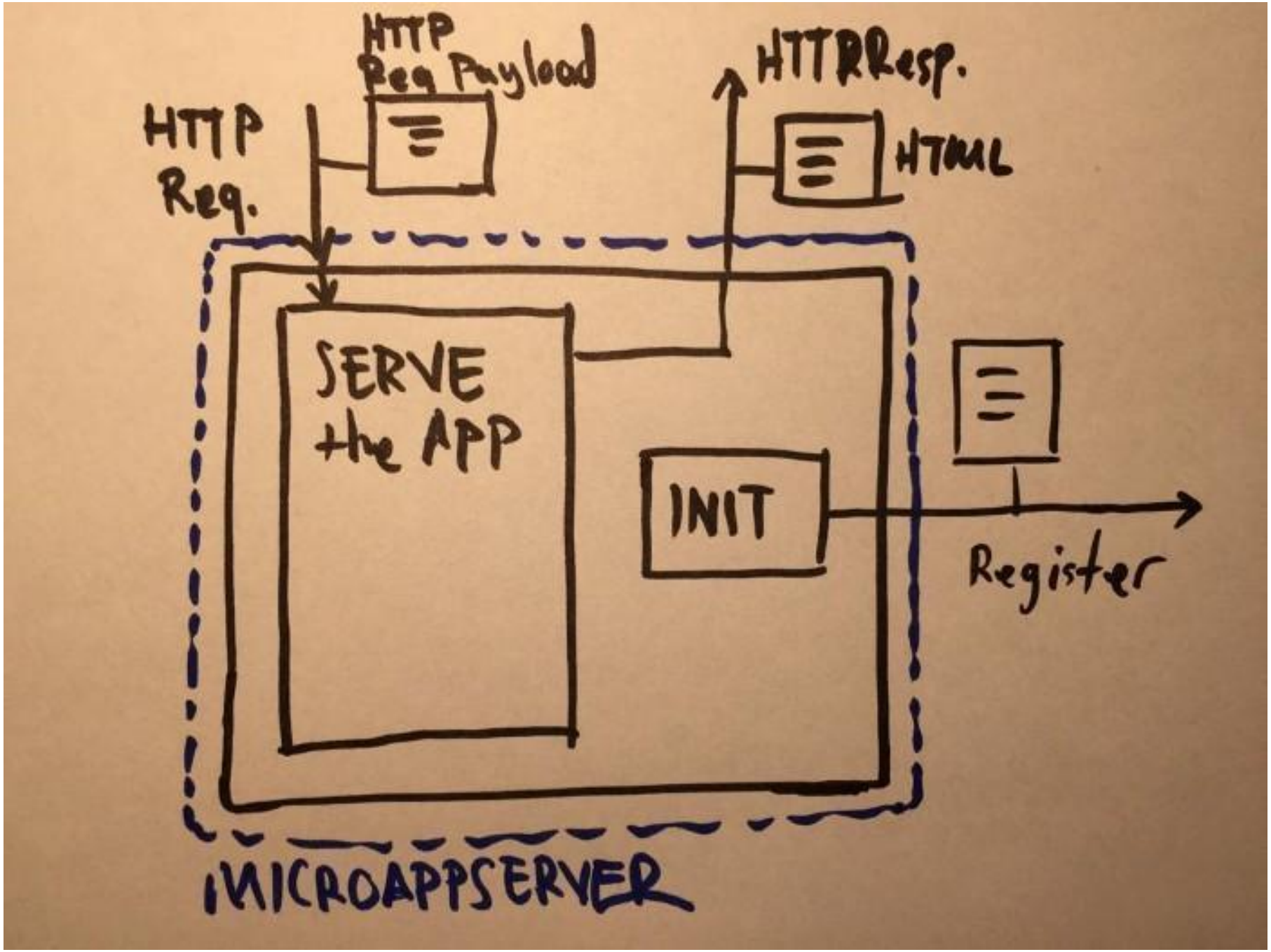
Мінімально можливий функціонал MicroAppServer можна висловити так: init і serve. Коли MicroAppServer завантажується, перше що він повинен робити - це викликати SticthingServer і зареєструвати ендпоінт з оголошеним мікро-додатком. Воно визначає залежності, типи і URL схеми MicroAppServer

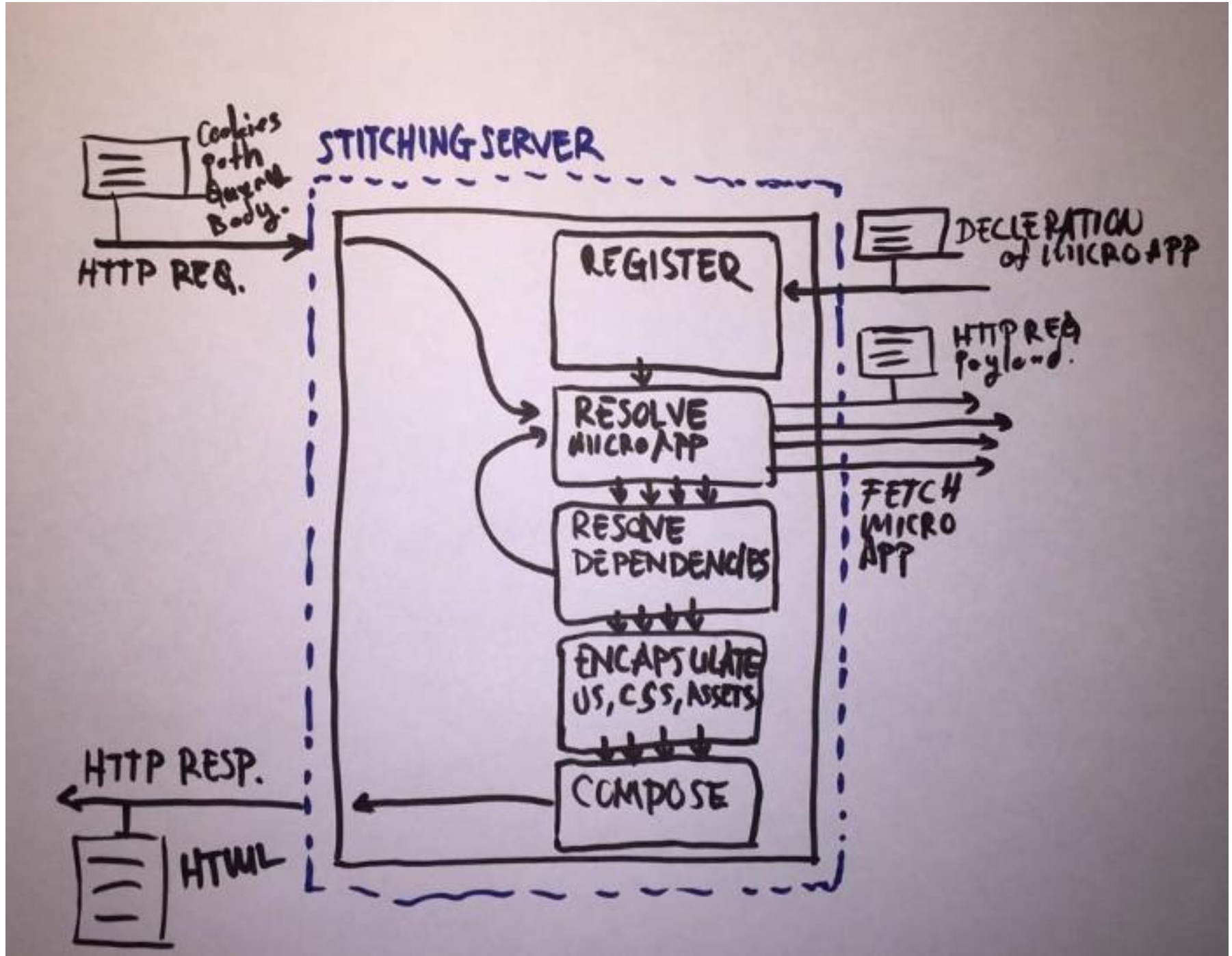
- StitchingServer

StitchingServer дозволяє зареєструвати endpoint в MicroAppServers.

Коли MicroAppServer реєструється в StichingServer, StichingServer записує оголошення MicroAppServer. Дозволивши MicroAppServer і все його залежності, в назвах усіх відповідних шляхів в CSS, JS і HTML з'явиться відповідний публічний URL. Додатковий крок - додавання до CSS-селекторів унікального префіксу MicroAppServer для запобігання конфлікту між мікрозастосунками на стороні клієнта.

Потім на сцену виходить головна задача StitchingServer: компоновка всіх отриманих частин і повернення цільної HTML-сторінки.

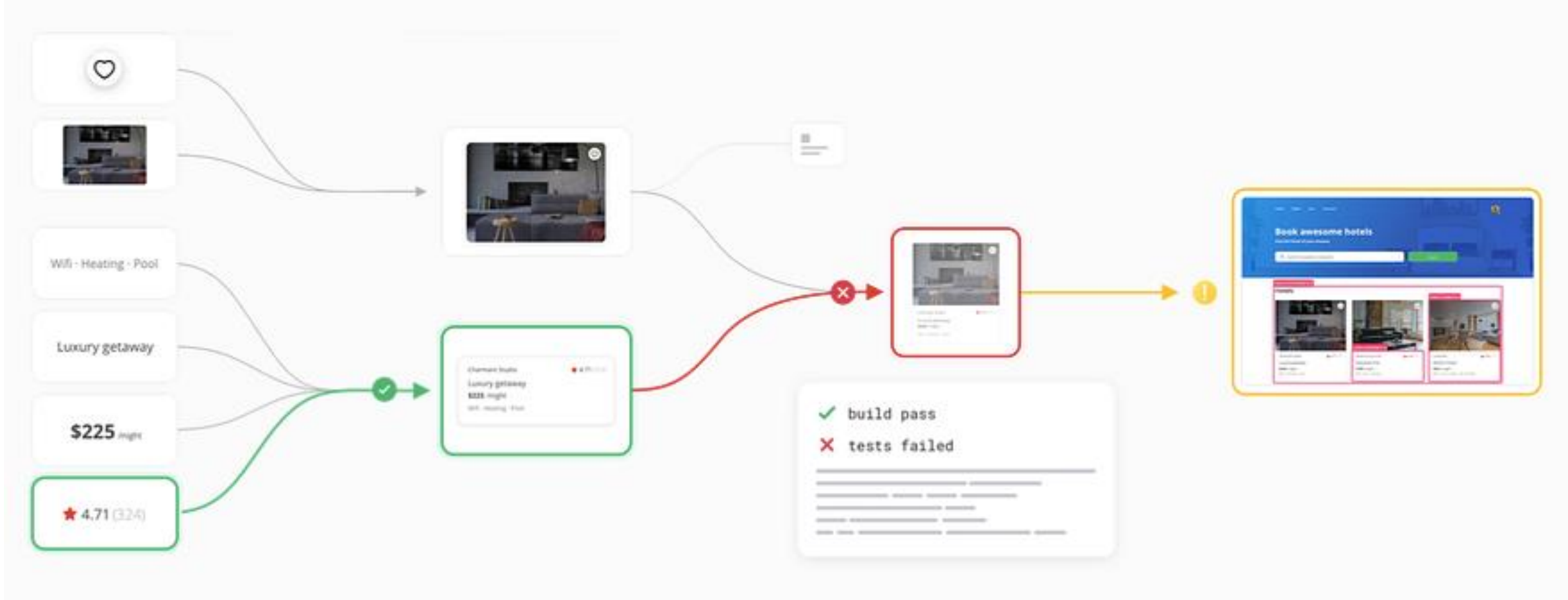

HTTP
Req Payload
HTTP
Req.
HTTPResp.
HTML
SERVE
the APP
INIT
Register
MICROAPPSERVER


Cookies
Path
Query
Body
HTTP REQ.
STITCHING SERVER
REGISTER
DECLERATION of MICRO APP
HTTP REQ Payload
RESOLVE MICRO APP
FETCH MICRO APP
RESOLVE DEPENDENCIES
ENCAPSULATE JS, CSS, ASSETS
COMPOSE
HTTP RESP.
HTML

MFE Composition

Інтеграція micro frontend може здійснюватися кількома способами

- **Композиція під час збірки**: у файлі package.json ви можете вказати micro frontend як залежності NPM і написати просту оболонку програми для маршрутизації та композиції компонентів під час збірки.
- **Композиція на стороні сервера**: micro frontend компонуються на сервері, що означає, що клієнт отримує повністю зібрану сторінку, що прискорює завантаження.
- **Композиція під час виконання**: композиція під час виконання передбачає отримання ресурсів із URL-адрес, які були розгорнуті незалежно. Micro frontend та їхні версії не потребують явного жорсткого кодування в залежності проекту.


Wifi · Heating · Pool
Luxury getaway
$225 /night
4.71 (324)
Book awesome hotels
build pass
tests failed

Кожен MicroFrontEnds(MFE)

- має власний життєвий цикл збірки
- може самостійно розгортати оновлення
- Має невеликий розмір в комплекті
- може бути відкладено завантажений
- повинен мати мінімальний зв’язок з іншими MFE
- може керуватися автономною командою

Who is using micro-frontends

The Model Store

basket: 0 item(s)

Tractor Porsche-Diesel Master 419



buy for 66,00 €

Related Products



Team Product

Team Checkout

Team Inspire

Приклад

https://serzn1.github.io/micro-frontends/1-composition-client-only/

3

https://micro-frontends.org/