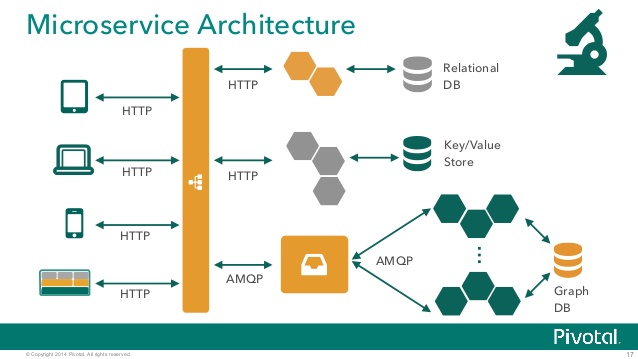
**Завдання до проекту з побудови Мікросервісів**

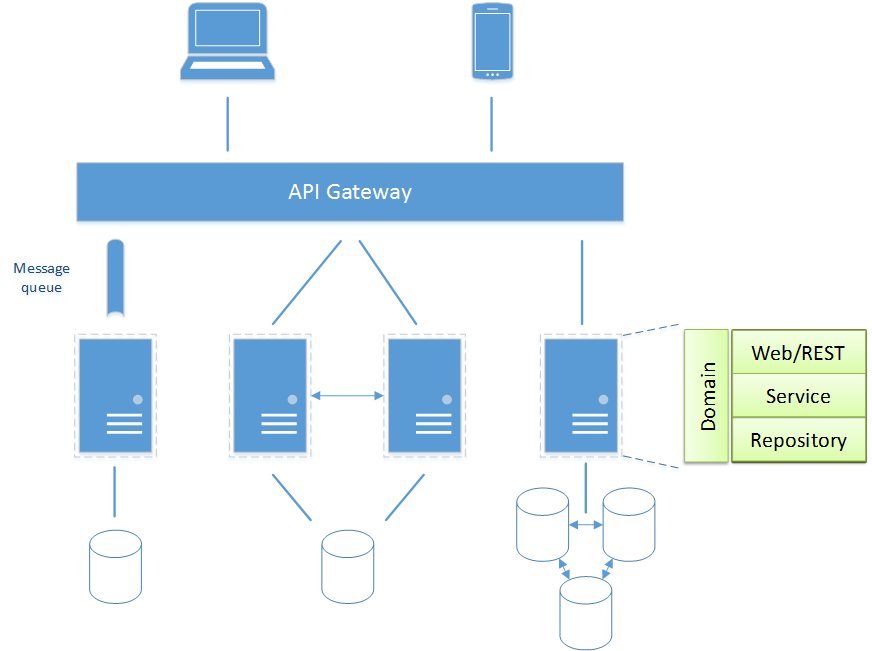
**Загальна характеристика і вимоги до проекту**

* Проект може виконуватись командою з 2х учасників
* Має бути схема з описом мікросервісів та їх взаємодією
* Середовище для централізованої розробки:
  + Source code repository (GitHub, BitBucket, ..) - обов'язково
  + Continuous integration (СI) - бажано
  + Kubernetes / Docker Compose - бажано
* до заліку розробка повинна бути закінчена



**Вимоги до системи**

* система повинна бути побудована на основі мікросервісной архітектури
* кожен з мікросервісов являє собою трьох-рівневий додаток, що складається з шару Веб (Web layer) або Native client, шару бізнес-логіки (Service + Domain layer) на сервері додатків / Веб-сервері, шару доступу до даних (Repository / Persistence layer)
* для кожного мікросервіса повинна використовуватися окрема БД: MongoDB, Cassandra, Postgres ...
* для одного з мікросервісов, сервера додатків повинен дублюватися і має бути налаштований таким чином щоб в разі відмови одного з серверів, запити йшли на інший сервер. При цьому, дані користувача (наприклад вміст сесії) повинні зберігатися. Для цього можна використовувати in-memory data grid (distributed cache): Redis, Hazelcast, ...
* для однієї з NoSQL БД повинна бути налаштована реплікація, що забезпечує її відмовостійкість. Наприклад, в разі MongoDB якщо кількість нод що залишилися стає менше кворуму, то БД повинна переходити в режим read-only
* для одного з мікросервісов запити повинні оброблятися асинхронно з використанням message queue. Вітається реалізація підходу event sourcing і CQRS
* клієнтська частина програми може безпосередньо виступати агрегатором мікросервісов або працювати через якийсь API Gateway. Вітається побудови програми з використанням RESTful API
* мікросервіси можуть бути написана на будь-яких з мов програмування



**Результати розробки**

* Вимоги до системи у вигляді:
  + короткого опису призначення (Vision),
  + короткий опис сценаріїв / варіантів використання (use cases) / product backlog
* Діаграми: архітектури системи
* Вихідний код системи в системі контролю версій
* Працююча система

Вся документація також може бути розміщена в системі контролю версій