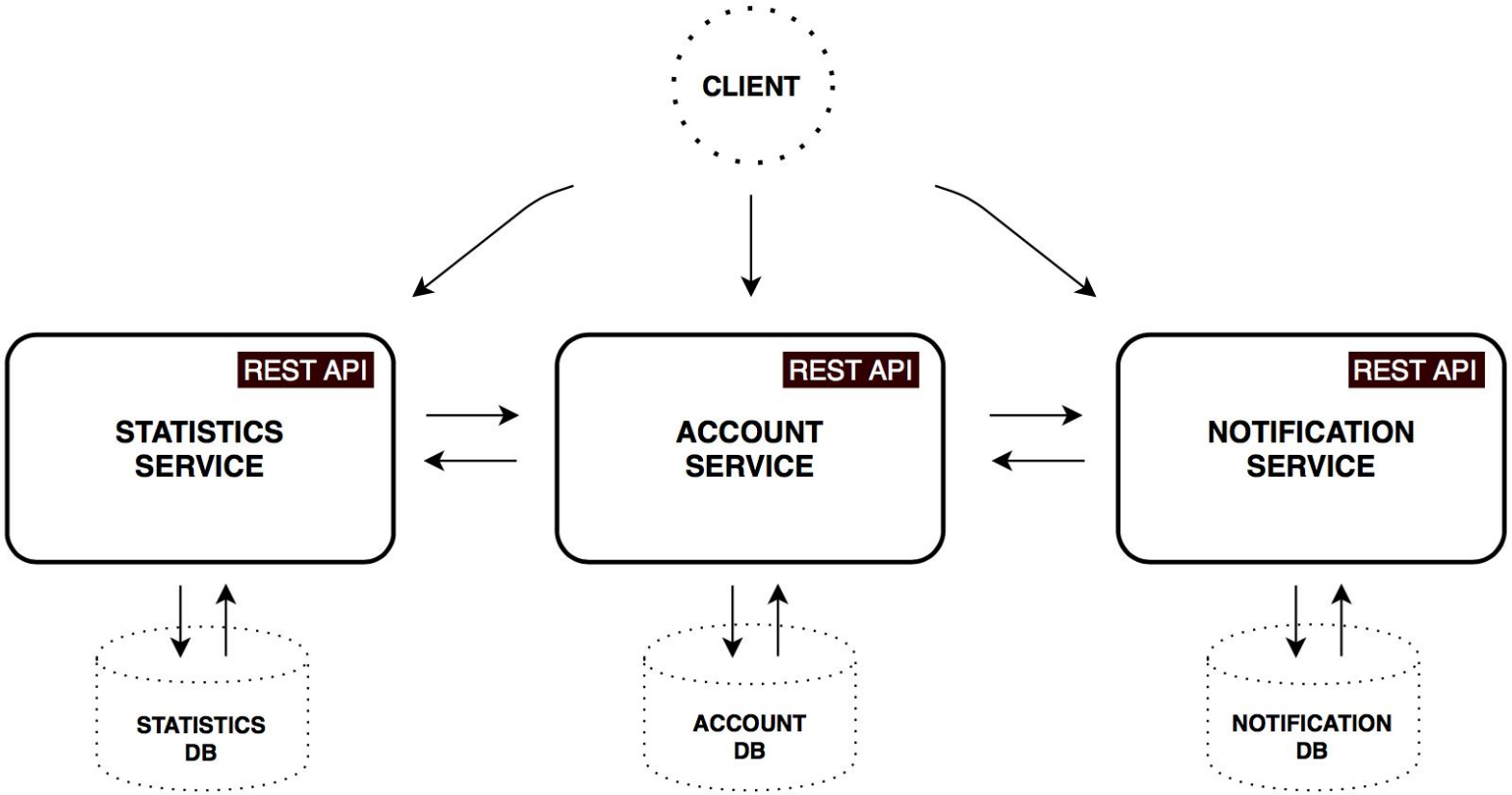
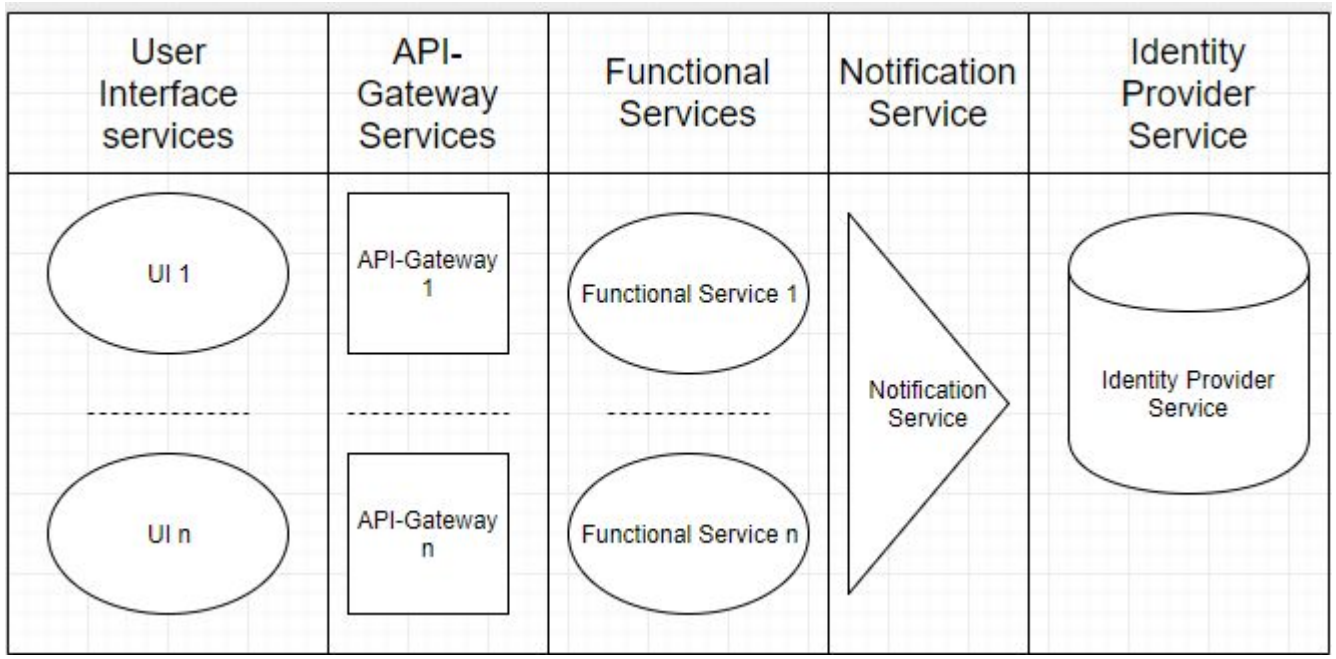
Тема 1. Основы микросервисной архитектуры

Преимущества и недостатки микросервисной архитектуры



Основные уровни микросервисов



Основные уровни микросервисов

- User Interface Services – уровень микросервисов для пользовательских интерфейсов
- API Gateway Services - уровень микросервисов, предоставляющих единую точку доступа к данным для UI
- Functional Services - уровень микросервисов, непосредственно содержащих логику обработки данных и реализующих правила взаимодействия с БД
- Notification Engeene – микросервис для обмена событиями между микросервисами
- Identity Provider – микросервисы, реализующие политики безопасности и разграничивающие права доступа пользователей к данным

Преимущества подхода

- Изменения в одном микросервисе не ведут к необходимости редеплоя других микросервисов
- Порог вхождения в предметную область ниже
- Разграничение зон ответственности разработчиков
- Возможность периодически менять команды разработки
- Тестирование функционала может производиться маленькими партиями
- Легко можно перевести микросервис на другой язык программирования
- Вертикальная и горизонтальная масштабируемость системы
- Независимость кода одного микросервиса от другого

Преимущества подхода

- Вызовы асинхронные
- Легче находить узкие места и оптимизировать код
- Можно использовать контракты для межсервисного взаимодействия
- Каждый микросервис при правильном проектировании имеет строго ограниченный функционал и за его рамки не выходит
- Добавление совсем нового функционала просто ведет к написанию нового микросервиса, не затрагивающего остальные

Недостатки подхода

- Требуется на этапе проектирования предусматривать больше нюансов и четко очерчивать круг функционала
- Уход от fullStack-разаботчиков
- Низкая связанность данных
- Увеличение сетевой нагрузки
- Возможность атак на API
- Возможно хранение в БД не консистентных данных
- Усложнение процесса CI\CD
- Много времени на отладку межсервисного взаимодействия

Что перевесит, плюсы или минусы – зависит от разработчиков и архитектора

Монолит -> Микросервисы

- Любое монолитное приложение можно разбить на микросервисы
- Наоборот – можно, но это глупо
- Процесс переписывания:
- 1) Очерчивание основных функциональных зон
- 2) Постепенное вынесение функционала в отдельные микросервисы
- 3) Отладка межсервисного взаимодействия
- 4) Отказ от монолита