**WCF**

Wcf технология включающая в себя asmx .

Обязательно полудуплекс. Обязательно хостинг .

Можно описать несколько кнтрактов. Тогда сервис будет реализовывать несколько интерфейсов.

Можем сделать много точек и указать в каждой точке свой интерфей. В точке указываем сервис. Тот контракт котоырй мы хотим.

Сервис- это класс, который реализует контракты . это значит что он реализует контракты сервиса, методы которые работают , они используют типы данных которые описаны в контракте данных, и то что находится в сообщениях , оно описывается в messageContract.  
Все работает через контракты.

Контракт данных нужен- если комплексные данные .

Контракт сообщений-

Сообщения бывают: запроса и ответ

В сообщения можно вносить доп информацию. Ее можно извлекать на стороне сервера.

Данные можно встраивать в headers/body.

Self hosting- Это значит что в качестве хоста может выступать любое .net приложение.

Для чего нужен хост?

Сервис это класс, не может сам работать по себе . Мы должны организовать процесс в рамках которого будет работать этот класс. То есть мы должны написать приложение, чтобы работал этот класс. Все ресурсы системой выделяются только процессу. Поэтому нужен процесс.   
Нужно приложение в которое мы сможем встроить этот сервис.

Взаимодействие клиента и хоста.

Надежность- связана с возможностью защиты от искажений.

Кодирование- есть возможность кодировать во все UTF . Можно создавать свои алгоритмы.

Транспорт - различные протоколы. HTTP-TCP- namePipe или собственный протокол.

WCF- платформа с помощью которой мы можем разрабатывать приложений клиент серверной архитектуры. Главная особенность – Она расширяемая.

**ODATA**

Это специальная надстройка над Wcf которая позволяет создавать сервис. И позволяет выполнять запросы к БД.   
Использую entity framework. Потом строим сервис и скармливаем ентити .

**МикроСервисы**

Микросервисы- логичское развитие сервисов связанное с тем что жизненный цикл программного обеспечения сократился и требует мгновенных изменений.

Основоной принцип при разработке : сервисы должны быть маленькими. Легковесные протоколы  
Сервис должен быть переписан за 2 недели.

Эта архитектура направлена на быструю адаптивность.   
такой подход усиливает производительность команды разработчиков.  
при разработке нудно придерживаться некоторых правил:

1. Сервис это элементарная функция, которая является независимой ( изменение сервиса никак не должно влиять на другие сервисы)
2. Каждый сервис имеет свою базу данных

Что важно для сервисов: непрерывность их работоспособности.  
основной принцип: Микросервис выполняет одну функцию.  
Микросервисы вызываются асинхронно.   
Проблема микросервисов: это обеспечение транзакций.

1. Нужно разрабатывать так что бы этого не было.
2. Распределенные транзакции.

Два принципа при организации микросервисов:……….