**Коллоквиум 1-ПWS-ПОИТ+ПОИБМС-4**

***Вопросы не пишем, пишем номер и ответ***.

1. В правом верхнем углу листа:

**1-ПWS-{ПОИТ,ПОИБМС}-4*/группа, Фамилия И.О.***

1. Расшифруйте аббревиатуру SOA, поясните что это.

**Service-oriented architecture**, **сервис-ориентированная архитектура** – парадигма проектирования и разработки программного обеспечения, основанная на применении распределенных слабосвязанных компонентов, обеспечивающих стандартные интерфейсы (открытые протоколы).

**SOA:** стиль архитектуры информационной системы, который позволяет создавать приложения путем комбинации слабосвязанных распределенных компонент.

**SOA:** программная платформа, позволяющая разрабатывать приложения, состоящие из слабо связанных распределенных компонент (AWS, Azure).

1. SOA: назначение компоненты ESB.

**SOA: ESB – Enterprise Service Bus:** программный компонент обеспечивающий обмен сообщениями между различными информационными системами, имеющих сервис-ориентированную структуру.

**SOA: ESB**

* синхронный и асинхронный вызов сервисов;
* гарантированная доставка сообщений;
* поддержка транзакций;
* маршрутизация сообщений;
* мониторинг, аудит и протоколирование.



**SOA: специальные компоненты** (как правило часть ESB)

* SOA Registry;
* Workflow Engine;
* Service Broker;
* SOA Supervisor.

1. SOA: назначение компоненты Registry.

**SOA:** **SOA Registry –** реестр сервисов, информация о сервисах и их интерфейсах. Gateway Service



1. SOA: назначение компоненты Workflow Engine.

**SOA:** **Workflow Engine –** программный компонент, предназначенный для построить (построить модель бизнес процесса) и выполнить бизнес процесс на основе группы сервисов. Другими словами разработать новый сервис на основе последовательного выполнения нескольких сервисов.



1. SOA: назначение компоненты Service Broker.

**SOA: Service broker** –программный компонент, позволяющий трансформировать запрос пользователя в системе в запуск и координированную работу.

**В архитектуре SOA (Service-Oriented Architecture) компонент Service Broker (посредник сервисов) выполняет роль диспетчера, обеспечивающего взаимодействие между клиентами и сервисами. Он упрощает процесс вызова сервисов, абстрагируя от клиентов детали их реализации и расположения.**



1. SOA: назначение компоненты Supervisor.

**SOA: SOA Supervisor** – служебный сервис, предназначенный для управления и мониторинга других служебных сервисов.

**SOA: Health Check Service** – служебный сервис для проверки работоспособности других сервисов, часто в составе Supervisor; **Health Check API:** API для проверки работоспособности. Системы мониторинга серверов: Nagios, Zabbiks, Icinga (Nagios Core), dotcom.monitor.

1. SOA: назначение компоненты Identity.

**SOA: Identity Service -**  сервис аутентификации, предназначен для реализации SSO (Single Sign On), протоколы токен-авторизации (JWT), OpenID Connect, OAuth2. IDaaS. (AWS IAM).

1. SOA: 5 способов взаимодействия сервисов с клиентом.

**SOA: способы клиент-серверного взаимодействия**











1. Расшифруйте аббревиатуру RPC, поясните что это.

**RPC** расшифровывается как **Remote Procedure Call** (удалённый вызов процедур). Это механизм, который позволяет программам вызывать функции или процедуры, расположенные на удалённом сервере, так, как если бы они находились локально в памяти компьютера.

1. Расшифруйте аббревиатуру REST, поясните что это.

**REST** расшифровывается как **Representational State Transfer** (передача состояния представления). Это архитектурный стиль, предназначенный для создания масштабируемых веб-сервисов, основанный на использовании стандартных протоколов и методов, таких как HTTP.

**REST**: **Representational State Transfer** - передача состояния представления.

**REST**: представление – это URI (идентификатор ресурса), управление ресурсом с помощью глаголов (GET, POST, PUT, DELETE).

**REST**: Архитектурный стиль взаимодействия компонентов распределенного приложения.

**REST**: автор **Roy Fielding**, описан в диссертации «Архитектурные стили и дизайн сетевых программных архитектур», 2000 г.

**REST**: реализация (альтернатива) RPC.

**RESTful**: описание web-службы, поддерживающей REST-интерфейс в полном объеме (со строгим соблюдением правил).

**REST**: нет официального стандарта, но REST использует стандарты HTTP, URL/URI, XML, JSON.

**REST:** два основныхтипа ресурса – коллекции и элемент коллекции: /api/users, /api/users/288.

**REST**: шесть обязательных ограничений:

* модель клиент-сервер;
* отсутствие состояния на стороне сервера, сохранение состояния допускается на стороне клиента, допускается сохранение состояния в другом сервисе (например, в БД);
* кэширование на стороне клиента, сервер явно управляет кэшированием;
* единообразие интерфейсов (идентификация ресурсов, манипуляция ресурсами через представления, самодостаточные сообщения, HATEOAS);
* для клиента сервер должен представляться конечным;
* код по требованию: допускается (необязательно) выгрузка на клиент апплетов или сценариев для расширения его функциональности.

1. Поясните разницу между REST и RPC.

REST фокусируется на работе с **ресурсами**. Клиент выполняет операции над ресурсами, идентифицируемыми через уникальные URI, с использованием стандартных HTTP-методов (GET, POST, PUT, DELETE и т.д.).

RPC ориентирован на вызов удалённых **функций или процедур**. Клиент вызывает метод на сервере так, как если бы он был локальным.

**REST**:

* Использует HTTP как транспортный протокол.
* Форматы данных: JSON, XML, HTML, текст.
* HTTP-методы соответствуют действиям (CRUD):
  + **GET**: Чтение.
  + **POST**: Создание.
  + **PUT**: Обновление.
  + **DELETE**: Удаление.

**RPC**:

* Может использовать любой протокол: HTTP, TCP, UDP.
* Форматы данных зависят от реализации: JSON-RPC, XML-RPC, gRPC (Protocol Buffers), CORBA.
* Вызов метода включает имя процедуры и параметры.

1. HTTP-методы применяемые в REST API и их специализация.

**REST:** GET (select), POST(insert), PUT (update), DELETE (delete).



1. Расшифруйте аббревиатуру HATEOS, поясните что это.

Это один из принципов архитектуры **REST**, который предполагает, что клиент может взаимодействовать с сервером не только через заранее заданные API-эндпоинты, но и через динамически предоставляемую информацию о доступных действиях, встроенную в ресурсы.

**HATEOAS: Hypermedia As The Engine Of Application State** – гипермедиа в качестве управления состоянием.

**Гипермедиа**: технология обработки, структурирования информации и произвольного доступа к ее элементам с помощью гиперсвязей (Тед Нильсон, 1965), WWW – реализация гипермедиа.

**HATEOAS: Hypermedia As The Engine Of Application State** – гипермедиа в качестве управления состоянием.

**Гипермедиа**: технология обработки, структурирования информации и произвольного доступа к ее элементам с помощью гиперсвязей (Тед Нильсон, 1965), WWW – реализация гипермедиа.

**HATEOAS: пример**



1. Правило сохранения состояния в REST API.

В **REST API** основное правило сохранения состояния заключается в том, что взаимодействие между клиентом и сервером должно быть **безсостоящим** (**stateless**). Это означает, что сервер не должен хранить состояние клиента между запросами. Каждое взаимодействие клиента с сервером должно быть полностью **независимым**.

1. Правило кеширования в REST API.

В **REST API** **правило кеширования** предполагает, что **сервер может кешировать ответы** для улучшения производительности и уменьшения нагрузки на сервер, а также для ускорения времени отклика для повторных запросов. Однако кеширование должно быть явно указано и контролируемо, чтобы избежать ошибок с устаревшими данными

1. Два типа ресурса в REST API.

**REST:** два основныхтипа ресурса – коллекции и элемент коллекции: /api/users, /api/users/288.

1. Расшифруйте аббревиатуру SOAP, поясните что это.

**SOAP:** Simple Object Access Protocol – простой протокол доступа к объекту.

**SOAP:** первоначально - реализация RPC, расширение протокола XML-RPC, позже - произвольный обмен XML-сообщениями.

**SOAP:** чаще всего используется над HTTP, но не обязательно.

1. SOAP: что такое Envelop?

***Envelope*** - обязательный элемент, который определяет начало и конец сообщения. Благодаря этому элементу, клиент, принимающий сообщение, знает, когда сообщение полностью получено.

В **SOAP (Simple Object Access Protocol)**, **Envelope** (или «конверт») — это основная часть SOAP-сообщения, которая содержит всю информацию, необходимую для того, чтобы запрос или ответ был правильно обработан. Он служит для упаковки и структурирования данных, а также для их идентификации.

1. SOAP: структура Envelop?

* ***Header*** - опциональный элемент, который может включать в себя любые опциональные атрибуты сообщения. Дает возможность определять дополнительные свойства приложения. Можно использовать несколько раз в файле.

Атрибуты:

* Actor - клиент может определить получателя элемента header.
* MustUnderstand - указывает является ли этот элемент обязательным или опциональным
* ***Body*** - обязательный элемент, который включает в себя данные XML, которые определяют сообщение, которое должны быть отправлено
* ***Fault*** - опциональный элемент, который предоставляет информацию об ошибках, которые возникают в процессе обработки сообщения
  + env: Сode – код сообщения;
  + env: Reason – пояснение к коду;
  + env: Node – узел назначения;
  + env: Role – роль назначения;
  + env: Detail – детальная информация.

Внутри каждого элемента Envelope должен быть один элемент Body.

Элемент Envelope определяется с помощью **пространства имён ENV** и элемента Enveloper.

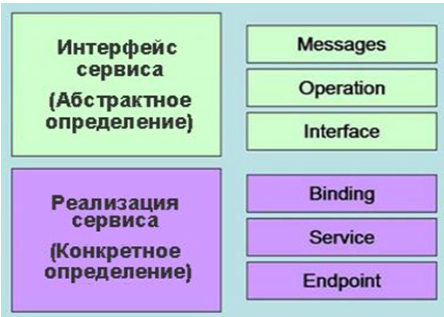


1. Расшифруйте аббревиатуру WSDL, поясните что это.

**WSDL: Web Services Description Language**

**WSDL:** разработанаIBM (NASSL), Microsoft (SCL), webMethods (WIDL).

**WSDL:** средство XML-описания интерфейса для доступа к web-службе; платформонезависимое описание;



***Абстрактная*** - описывается в языке WSDL с помощью системы типов, XML-схемы, в терминах сообщений, которые этот сервис отправляет и получает.

1. **Шаблоны обмена** сообщениями определяют последовательность и количество сообщений.
2. Элемент **operation** связывает шаблоны обмена сообщениями с одним или несколькими сообщениями.
3. Элемент **interface** группирует ***operation*** независимо от транспорта и способа доставки.

В ***конкретной части***

1. Элементы **binding** задают транспорт и формат доставки для ***interface***.
2. Элемент **endpoint** связывает сетевой адрес в соответствие с *элементом* ***binding***.
3. Элемент **service** группирует *элементы* ***endpoint***, которые реализуют общий ***interface***
4. Основные свойства ASMX-приложения.

**ASMX: Active Server Method Extended;** технология Microsoft для разработки web-сервисов(2002, 2007), основанная на XML, WSDL, SOAP; официальное название в MSDN «XML Web Services»; легкая технология: для работы с ней не обязательно знать XML, SOAP и WSDL.

Основные свойства ASMX-приложения

** Использование SOAP — передача данных через протокол SOAP.**

** Простота в разработке — быстрое создание веб-сервисов с помощью .asmx-файлов.**

** XML-сообщения — данные передаются в формате XML.**

** Поддержка HTTP и HTTPS — использование HTTP для передачи запросов и ответов.**

** Автоматическое генерирование WSDL — автоматически генерируется описание веб-сервиса (WSDL).**

** Совместимость с клиентами — поддержка различных клиентов, таких как .NET, Java, и другие.**

** Поддержка WebMethods — методы, помеченные атрибутом [WebMethod], могут быть вызваны удаленно.**

** Поддержка сессионности — возможность использования сессий (если необходимо).**

** Безопасность — поддержка аутентификации и авторизации через стандартные механизмы.**

** Поддержка расширений — возможность добавления пользовательских обработчиков и фильтров.**

1. ASMX: что такое Proxy-клиент.

**Proxy-клиент** в контексте **ASMX** — это сгенерированный клиентский класс, который позволяет взаимодействовать с веб-сервисом, доступным через **ASMX** (ASP.NET Web Service). Proxy-клиент служит промежуточным звеном между приложением-клиентом и веб-сервисом, обеспечивая абстракцию от подробностей сетевого взаимодействия и SOAP-протокола.

1. Расшифруйте аббревиатуру WCF, поясните что это.

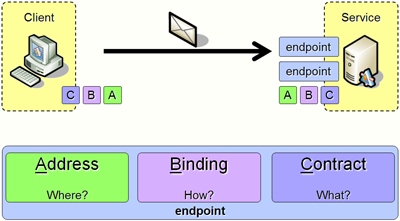
**WCF** (Windows Communication Foundation) — это фреймворк от Microsoft для создания распределённых приложений, который позволяет обмениваться сообщениями между различными компонентами через различные протоколы (например, HTTP, TCP, MSMQ). WCF обеспечивает безопасное, надёжное и масштабируемое взаимодействие между сервисами.

**WCF:** Windows Communication Foundation; технология основанная на .NET FRAMEWORK для разработки приложений SOA-архитектуры, первая версия 2006 (Indigo).

**WCF:** основные принципы

* разработка сервиса должна быть простой и иметь способность к расширению его функциональных возможностей;
* один API для всех коммуникационных протоколов;
* сервис должен быть интероперабельным (функционировать по отрытым телекоммуникационным стандартам);
* сервис должен поддерживать стандарты WS-\*;
* сервис должен поддерживать REST, RPC и др. архитектуры;

**WCF:** коммуникация модель



**WCF:** контракты

* контракт службы;
* контракт данных;
* контракт сообщений.

1. Что можно сделать с WCF, но нельзя с ASMX?

**Разница между WCF и ASMX веб сервисами**

ASMX простой, но во многих отношениях ограничен по сравнению с WCF.

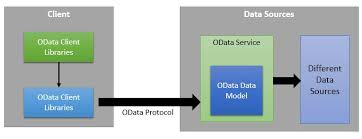
1. ASMX веб сервисы могут быть размещены только в IIS, в то время как WCF сервис имеет больше вариантов хостинг (IIS, WAS (Windows Process Activation Services), Console Application, Windows NT Services, WCF provided Host)
2. ASMX веб сервисы ограничены в поддержке одного лишь HTTP, в то время как WCF поддерживает HTTP, TCP, MSMQ, Named Pipes. (ключевое!)
3. Безопасность ASMX ограничена. WCF поддерживает больше протоколов. Обычно аутентификация и авторизация ASMX производится используя конфигурацию безопасности IIS и ASP.NET и безопасность транспортного слоя. Для слоя сообщений может быть использовано WSE. WCF предоставляет согласованную программную модель безопасности для любого протокола и поддерживает много таких же возможностей как IIS и WS-\* протоколы безопасности. Дополнительно, WCF предоставляет поддержку claims-based авторизации которая позволяет контролировать ресурсы более четко чем безопасность на основе ролей (role-based security). WCF безопасность остается согласованной не зависимо от того какой хост был использован для реализации WCF сервиса.
4. OData, что это?

**OData** (Open Data Protocol) — это открытый протокол для создания и использования запросов к данным, который позволяет клиентам получать и манипулировать данными через стандартные HTTP-запросы. OData стандартизирует способы доступа к данным, предоставляет удобный интерфейс для работы с ними и поддерживает различные форматы данных, такие как **JSON** и **XML**.

**OData: Open Data Protocol –** открытый web-протокол; позволяет выполнять операции с ресурсами и получать ответы в форматах XML, JSON.

**OData:** поддерживается OASIS

**OData:** архитектура

****

для получения JSON, необходимо обеспечить в заголовке заголовок ACCEPT:application/json или $format=json

Возможности предоставляемые OData-интерфейсом:

* Простое чтение данных (запросы без параметров) : одного элемента, коллекции
* Параметризованные запросы (поиск, сортировка, и т.д.)
* Добавление данных
* Модификация данных
* Удаление данных
* Добавление, модификация и удаление связей между объектами

Порядок разработки:

1. Создать модель ADO.NET EDM (подключиться к бд, выьрать таблицы и тд)
2. Создать WCF Data Service
3. Add connected service (1)
4. На клиенте добавить ссылку на службу (2)
5. RSS и ATOM, что это?

**RSS (Really Simple Syndication) и ATOM — это форматы для распространения обновлений и публикаций контента, таких как новости, блоги или другие типы информации через интернет. Оба формата используются для того, чтобы пользователи могли автоматически получать свежие обновления с веб-сайтов.**

**RSS: (Really Simple Syndication)** семейство XML-форматов, предназначенного для описания новостных лент, анонсов и статей (199) .

**RSS:** последняя версия 2.0.(2007)

**RSS:** MIME: application/rss+xml.

**RSS:** Интернет-ресурс в формате RSS называется RSS-каналом, RSS-лентой, RSS-фидом.

**RSS:** RSS-агрегатор – web-приложение, которое автоматически с заданным интервалом времени проверяют, указанные в настройках ресурсы и экспортируют в формате RSS или ATOM информацию (например, заголовки новостей, блогов и пр.). Feedly, Яндекс.Лента, My Yahoo.

**ATOM**: формат описания web-ресурсов и протокола для их публикации (2005г.).

**ATOM**: последняя версия 1.0

**ATOM**: MIME: application/atom+xml.

**ATOM**: формат RFC4287.

**WCF Syndication Services:** - реализация WCF для новостных каналов

**ATOM** - формат описания web-ресурсов и протокола для их публикации (application/atom+xml)

ATOM более универсальный и чаще применяется; следует использовать его при разработке новых приложений.

**ATOM:**

* **Сокращение**: ATOM не является акронимом, а просто названием формата.
* **Формат**: ATOM — это также формат XML, схожий с RSS, но с более строгими правилами и большими возможностями для метаданных.
* **Преимущества**: ATOM предоставляет более четкую спецификацию для метаданных, таких как даты и авторство, и поддерживает дополнения, что делает его более гибким и расширяемым по сравнению с RSS.
* **Популярность**: ATOM, как правило, используется для более сложных или корпоративных приложений, где требуются дополнительные возможности.

<feed>

<id> — Идентификатор фида, постоянный URI.

<title> — Название канала. Не может быть пустым.

<updated> — Дата последнего обновления

<author>

<link> — Адрес связанного сайта. Тип связи определяется в атрибуте rel.

<category> — Задает категории, к которым относится канал. Элемент feed может включать несколько элементов category.

<generator> — Название программы, с помощью которой собран канал.

<icon> — Маленькое изображение, пиктограмма канала.

<logo> — Большое изображение, логотип канала.

<rights> — Информация об авторском и смежных правах.

<subtitle> — Человеко-читаемое описание или подзаголовок канала.

<entry>

<id>uuid:2e6c2527-997a-4310-8028-5626bef6fdbd;id=4</id>

<title type="text">Java</title>

<updated>2022-12-27T19:03:00Z</updated>

<content type="text">8</content>

</entry>

</feed>

**RSS** - семейство XML-форматов, предназначенного для описания новостных лент, анонсов и статей

<rss xmlns:a10="http://www.w3.org/2005/Atom" version="2.0">

<channel>

<title>lab 7</title>

<description>description 7 lab</description>

<language>ru-ru</language>

<managingEditor>tukaalena2@gmail.com</managingEditor>

<lastBuildDate>Tue, 27 Dec 2022 22:02:46 +0300</lastBuildDate>

<category>students</category>

<generator>C# RSS</generator>

<item>

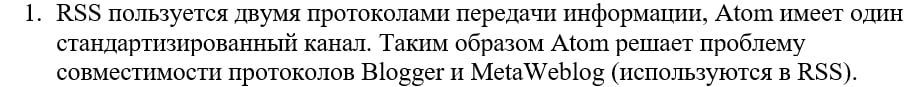
<title>Java</title>

<description>8</description>

</item>

</channel>

</rss>



Порядок разработки:

1.Создать проект Библиотека службы публикации

2. на клиента добавить reference на службу

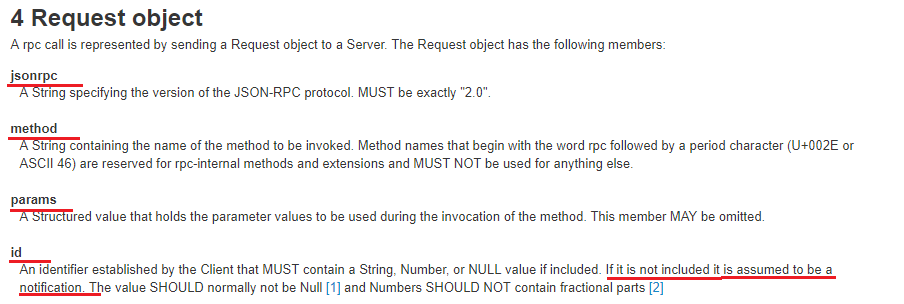
1. JSON-RPC, что это?

**JSON-RPC:** протокол удаленного вызова процедур, использующий формат JSON для передачи сообщений.

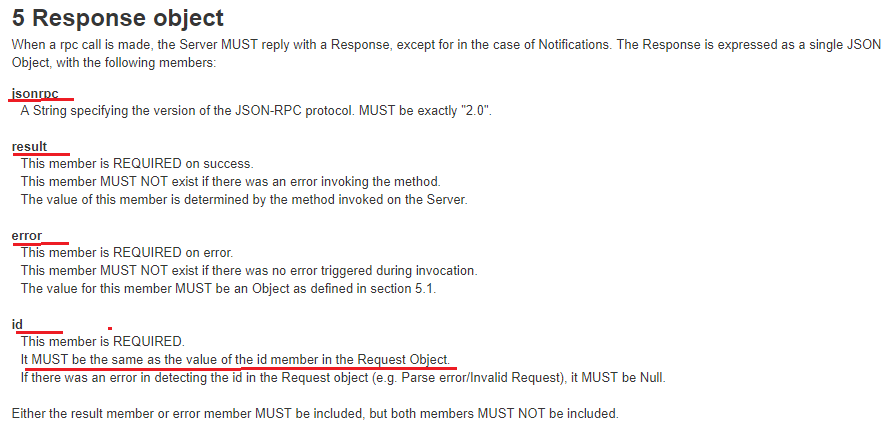
**JSON-RPC:** последняя версия 2.0.

транспортHTTP/HTTPS или TCP Socket.

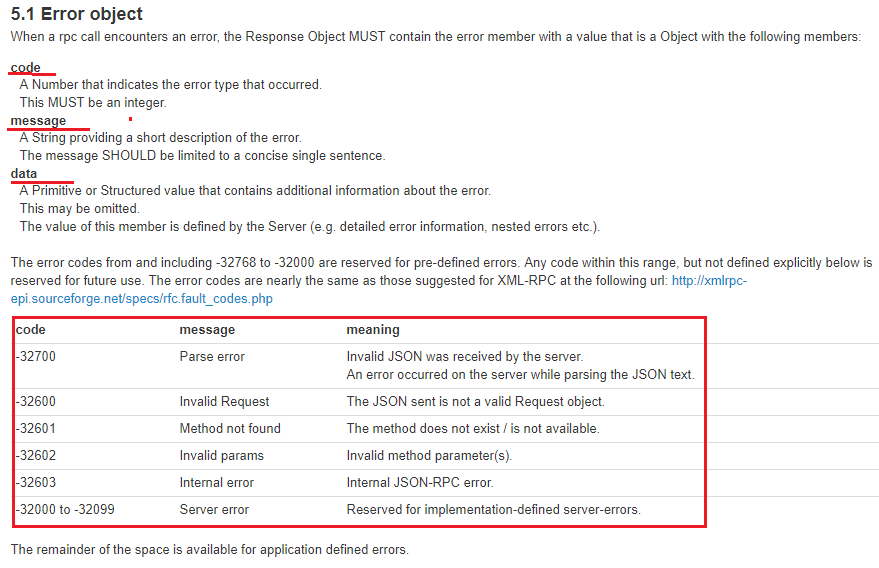
1. **JSON-RPC:** объект запроса

****

1. **JSON-RPC:** объект ответа

****

1. **JSON-RPC:** объект ответа (ошибка)

****

1. Дайте определение понятию «микросервис».

**Микросервисы:** один из подходов к разработке SOA-приложений, основной принцип – сервисы должны быть легкими (сервисы маленькие и протоколы легковесные). Легкий сервис – сервис, который может быть переписан за 2 недели (Real Estate).

**Микросервисы:** цель: 1) устойчивость к сбоям; 2) облегчить понимание и поддержку кода; 3) усилить работу команды программистов.

**Микросервисы:** микросервис – сервис, выполняющий одну элементарную функцию; основной принцип разбиения – изменение сервиса не затрагивает другие сервисы.

**Микросервисы:** микросервисная архитектура - набор принципов, которым должны соответствовать сервисы. Задается или в форме правил или основывается на применении готовых фреймворков (например, Karyon, Dropwiard,…).

**Микросервисы:**  умные приемники и глупые каналы. Enterprise Service Bus (ESB) – не является приемлемым решением. Обычно применяется простой HTTP/REST, но необязательно. Есть легковесные шины для обеспечения асинхронности (MOM: RabbitMQ, ZeroMQ).

**Микросервисы:** HTTP+HATEOAS, XML/JSON.

1. DevOps, что это?

**DevOps** (от английского *Development* и *Operations*) — это набор практик и культурных принципов, направленных на интеграцию и автоматизацию процессов разработки (Dev) и операций (Ops) в разработке программного обеспечения. Основная цель DevOps — улучшить взаимодействие между разработчиками и операционными командами для повышения скорости разработки, улучшения качества программного обеспечения и сокращения времени на внедрение изменений.

**Микросервисы:** требуется ***DevOps*** (Development & Operation)- набор технологий нацеленных на интеграцию процессов разработки и информационно-техническому обслуживанию. Цели DevOps: сокращение выхода продукта на рынок, снижение частоты отказов релизов, сокращение времени на изменения, сокращение времени на восстановление. Задача DevOps сделать согласованным процесс разработки и эксплуатации приложений. Основные задачи DevOps: контроль версий, непрерывная сборка, непрерывное тестирование, поддержка репозиториев артефактов, конфигурация инфраструктуры, мониторинг работоспособности и производительности. DevOps – следствие увеличения релизов (Agile-технологии) и усложнения инфраструктуры

1. Расшифруйте аббревиатуры DDD и CQRS и поясните, что это.

**DDD — Domain-Driven Design**

**Domain-Driven Design (DDD)** — это методология проектирования программного обеспечения, ориентированная на углубленное понимание и моделирование бизнес-области (домена) с целью создания более эффективных и понятных решений

**DDD:** Domain-Driven Design - предметно(проблемно)-ориентированное проектирование: набор принципов направленных на создание систем объектов со сложной предметной областью.

**DDD:** цель–ускоритьпроцесс проектирование программного обеспечения.

**DDD:** введен Э. Эвансом в книге «Domain-Driven Design», 2004г.

**DDD:** Domain – предметная область. **Эксперт домена** – специалист в предметной области.

**DDD:** 1) создание единого языка домена (Ubiquitous Language).Язык - набор однозначно определенных терминов. Тезаурус.

**DDD:** 2) выделение ограниченных контекстов (имена).

**DDD:** 3) определение контрактов (имена) (интерфейсов взаимодействия контекстов); контракты описывают команды (приводит к изменению состояния контекста, могут быть отклонены) и события (сообщение системы об изменении состояния, на них могут подписываться другие контексты, могут быть проигнорированы, но не могут быть отклонены).

**DDD:** 4) описание контекстов.

**DDD:** система имеет 4уровня: interface (пользовательское взаимодействие, взаимодействие с другими системами), application (реализация, технические аспекты приложения, хранение данных, создание данных), business (реализация доменного языка), infrastructure (платформа и дополнительное программное обеспечение).

**DDD:** рекомендует паттерны: Value Object, Entity Object, Aggregate (граница транзакции, граница Entity), Repository, Factory, Service.

**DDD:** модули (высокая степень независимости).

**DDD:** документация на языке

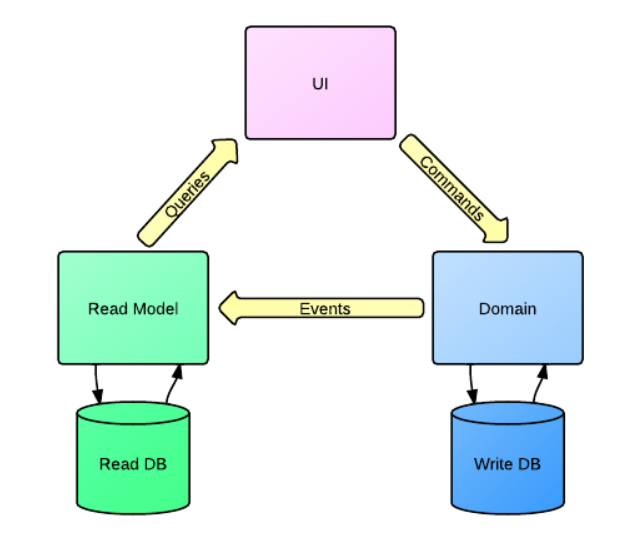
**DDD:** используется для больших проектов, длительностью более 6 месяцев.

**Command-Query responsibility segregation(CQRS)**

**CQRS**: разделение ответственности на команды и запросы ; CQS – command-query separation(Б. Майер) = принцип императивного программирования (команда-память-команда-память, ассемблер – типичный императивный язык). Основной принцип – команда-чтение не вызывает побочного эффекта(не изменяет состояние объекта).

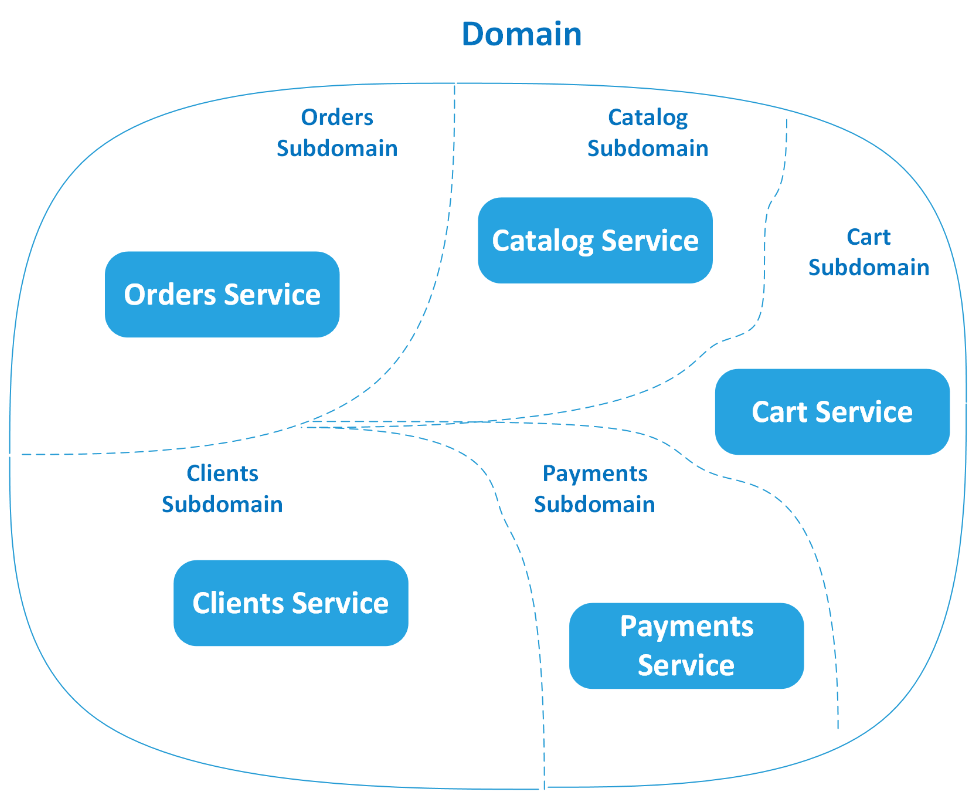
**CQRS**: контрактное программирование (Б.Майер): спецификация методов (интерфейсы), предусловие, постусловие, инвариант (обязательные соглашения в методе), исключения.

**CQRS**: перенос основных принципов CQS на архитектуру (М.Фаулер)

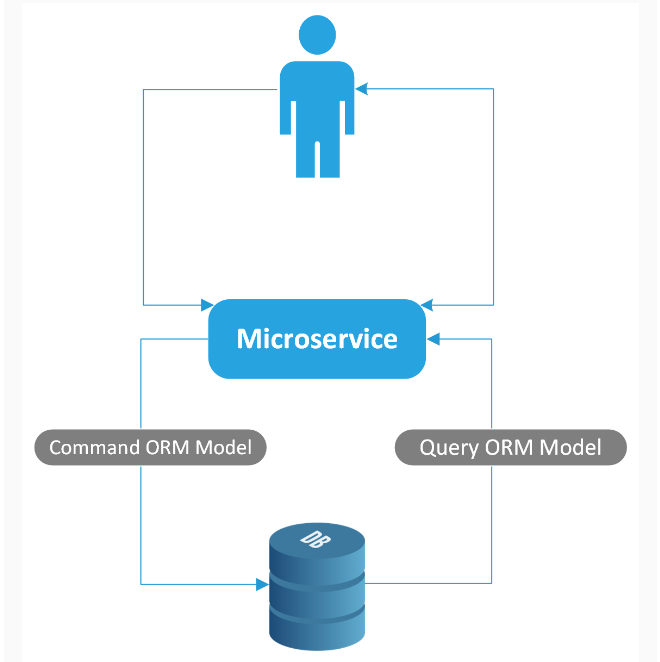


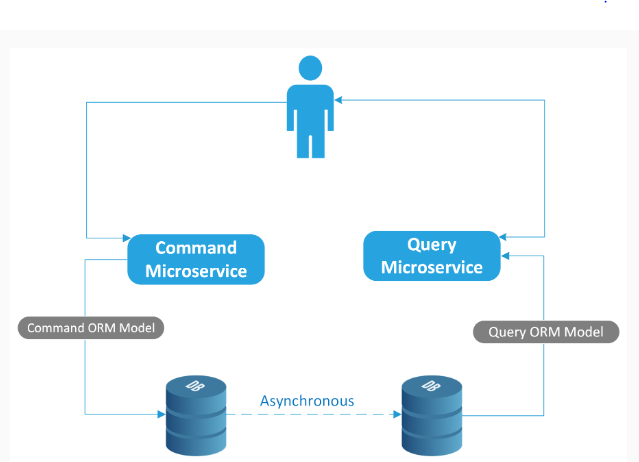
**CQRS:** обычно 2 базы данных: для записи и для чтения. Базы данных синхронизируются.

**Decompose by Subdomain: разбиение по поддоменам,** Domain-Driven Design (DDD)



1. **Command Query Responsibility(CQRS): разделение команд и запросов, простая и расширенная формы**

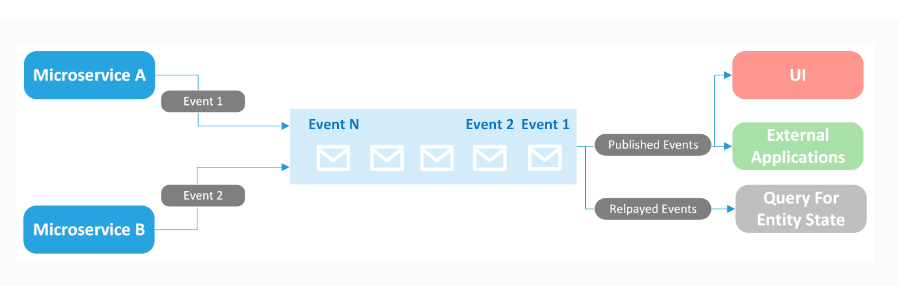




1. Поясните принцип Event Sourcing.

**Event Sourcing** — это архитектурный паттерн, при котором изменения состояния системы сохраняются не как текущие состояния, а как последовательность **событий** (events), описывающих, как это состояние изменялось с течением времени.

**Event Sourcing: поиск событий, изменение в данных приложений (UI, External Application, Entity State) осуществляется только через очередь событий, для высокопроизводительных, масштабируемых приложений, CQRS**

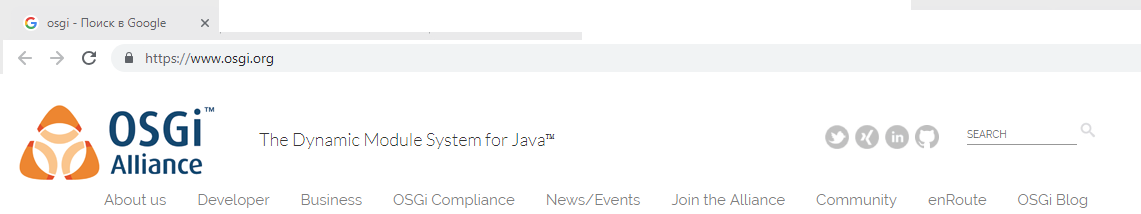


1. **ES:** подход к хранению данных, при котором вместо конечного результата хранится череда событий произошедших с некоторой сущностью.
2. **ES:** каждое событие имеет имя.
3. **ES:** произошедшие события неизменны.
4. **ES:** проекция - вычисленные результаты для UI (аналог view).
5. **ES:** ориентация на процесс (workflow driven), а не на данные (data-input based).
6. **ES:** обычно применяется в архитектуре CQRS для реализации WriteDB.



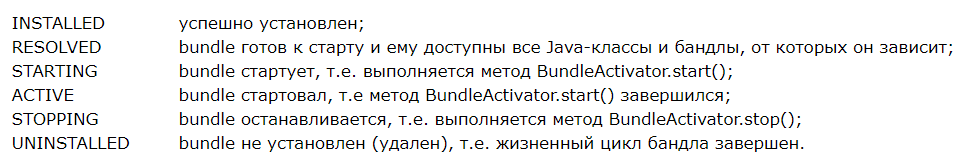
1. Поясните смысл OSGi-технологии.

технология, предназначенная для создания модульных, динамично загружаемых и управляемых приложений в Java-среде.

1. **OSGi:** Open Services Gateway initiative – инициатива доступа к открытым службам.
2. **OSGi:** OSGi Alliance – организация, разрабатывающая спецификацию платформы OSGi (другое название The Dynamic Module System for Java). 
3. **OSGi:** спецификация динамической модульной системы и сервисной платформы для Java-приложений. Позволяет построить приложение из компонентов, которые можно динамически переинсталлировать компоненты (модули).
4. **OSGi:** изначально спецификация разрабатывалась для встроенных систем (BMW, Siemens), затем распространилась на многофункциональные приложения.
5. **OSGi:** процесс

* разрабатывается ***модуль*** (bundle, jar);
* модуль может быть ***динамически подсоединен*** к системе модулей, работающей на платформе OSGi;
* модуль может быть ***динамически отсоединен*** от системы модулей, работающей на платформе OSGi.

1. **OSGi:** последняя версия OSGi Release 7(R7), апрель 2018.
2. **OSGi:** жизненный цикл OSGi bundle (интерфейс **BundleActivator**).

****

1. Расшифруйте аббревиатуру MOM и поясните что это.

**Message-Oriented Middleware (MOM)** — это тип программного обеспечения, которое предоставляет инфраструктуру для обмена сообщениями между различными компонентами распределенных систем. MOM позволяет приложениям обмениваться данными и сообщениями асинхронно, независимо от того, когда и где они были отправлены или получены, и от того, как долго приложение будет ждать ответа.

 **RabbitMQ** — популярная система обмена сообщениями с поддержкой очередей.

 **Apache Kafka** — используется для обработки потоковых данных и обмена сообщениями в реальном времени.

 **ActiveMQ** — брокер сообщений, поддерживающий различные протоколы и механизмы доставки сообщений.