

Paweł Rajba

pawel@cs.uni.wroc.pl

<http://pawel.ii.uni.wroc.pl/>

Integracja aplikacji

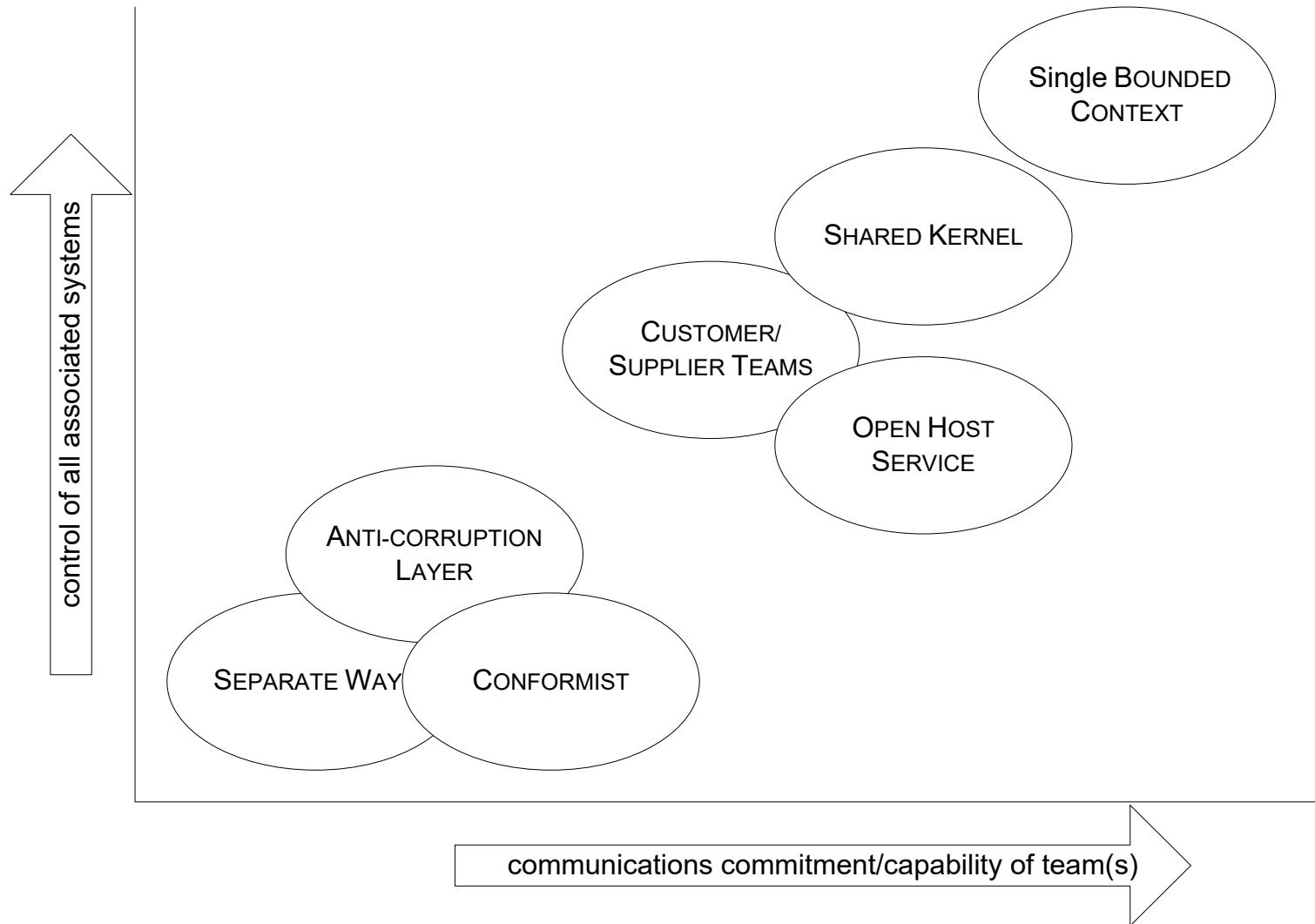
Agenda

- DDD & Context Maps
- Rodzaje integracji
- Messaging Patterns
- SOA
- API Management

DDD & Context Maps

- W integracji aplikacji istotna jest relacja pomiędzy aplikacjami i tej relacji konsekwencje
- Problem modeli i ich translacji
- Przyjrzymy się, jak wygląda to z perspektywy DDD
 - I zamiast aplikacji będziemy rozpatrywać Bounded Context

Context Maps



Wzorce relacji

- Partnership
 - Zespoły obu systemów razem pracują nad integracją i wypracowują kompromisy
 - Jest wspólny cel i albo oba zespoły będą miały sukces, albo porażkę
- Shared kernel
 - Wyróżniona wspólna część modelu
 - Powinna być jak najmniejsza i zgodna z Ubiquitous Language obu systemów
 - Ewentualne zmiany zawsze powinny być uzgodnione z drugą stroną

Wzorce relacji

- Customer-Supplier
 - Relacja oparta na upstream-downstream
 - Relacja partnerska: customer może określić potrzeby, które supplier stara się dostarczyć
- Conformist
 - Relacja oparta na upstream-downstream
 - Dostawca ma interesu w realizacji potrzeb odbiorcy i to odbiorca musi się dostawać

Wzorce relacji

- Separate Ways
 - Bounded contexty, które nie mają z sobą relacji
- Big Ball of Mud
 - Jeden wielki monolit

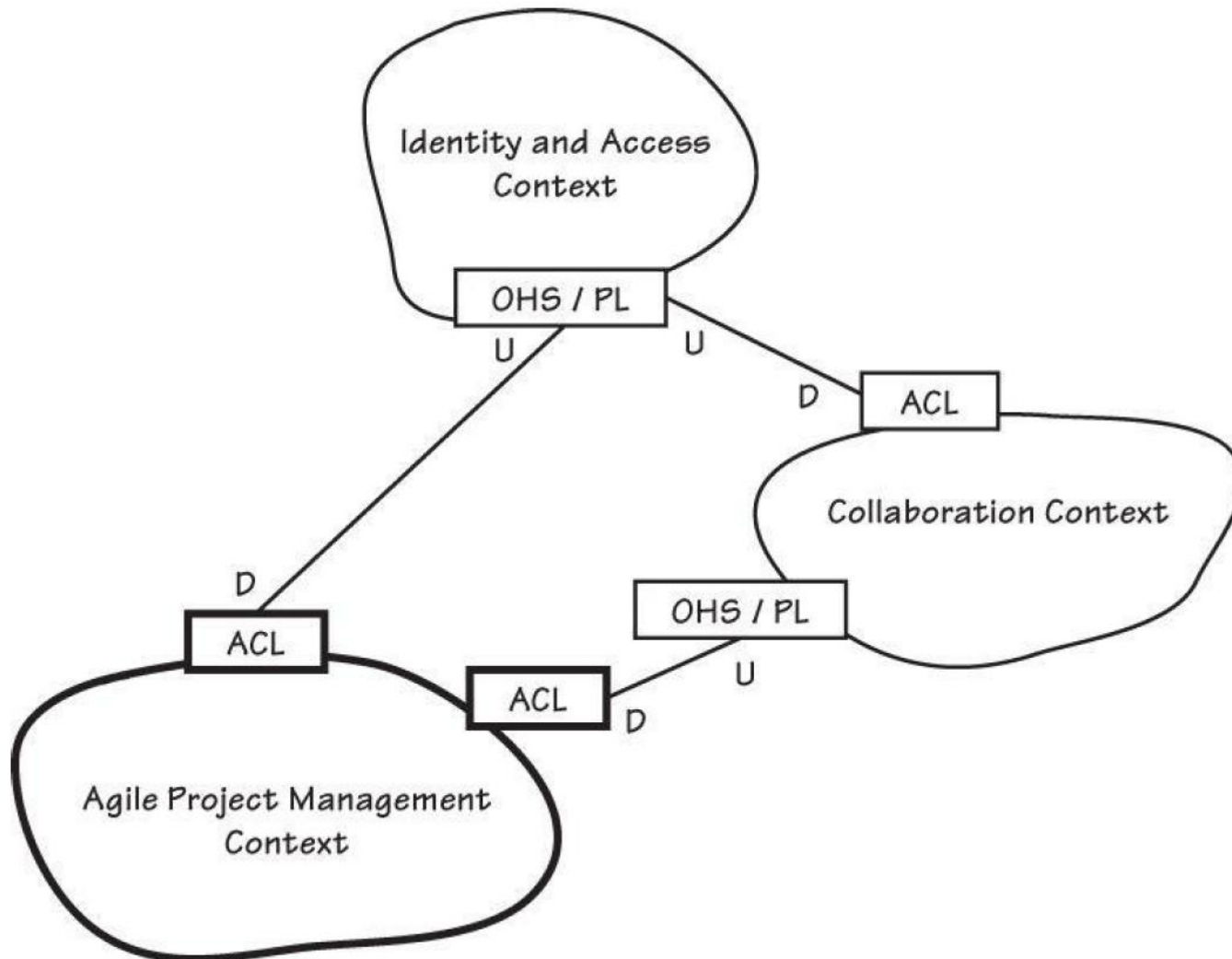
Dodatkowe elementy

- Anticorruption Layer (ACL)
 - Warstwa zapewniająca translacje zewnętrznego modelu do wewnętrznego
 - Zwykle po stronie downstream
- Open Host Service (OHS)
 - Definicja protokołu, dzięki któremu można zbudować integrację
- Published Language (PL)
 - Definicja języka (formatu) udostępnionych danych

Context Map

- Diagram przedstawiający
 - Bounded contexty
 - Translacje
 - Relacje między nimi
 - Elementy dodatkowe jak OHS czy PL


Context Map



Rodzaje integracji

- File transfer (e.g. FTP)
- Shared database
- Remote procedure call (RPC)
- REST/OData/GraphQL
- Messaging

API Architecture Types

@Rapid_API 

REST (Representational State Transfer)

- Follows six REST architectural constraints
- Can use JSON, XML, HTML, or plain text
- Flexible, lightweight, and Scalable
- Most-used API format on the Web
- Uses HTTP



GraphQL

- A query language for APIs
- Uses a Schema to describe data
- Functions using queries and mutations
- Uses a single endpoint to fetch specific data
- Used in apps requiring low bandwidth



SOAP (Simple Object Access Protocol)

- Strictly defined messaging framework that relies on XML
- Protocol independent
- Secure and extensible
- Used in secure enterprise environments



RPC (Remote procedure Call)

- Action-based procedure great for command-based systems
- Uses only HTTP GET and POST
- Has lightweight payloads that allow for high performance
- Used for distributed systems

Apache Kafka

- Used for live event streaming
- Communicates over TCP protocol
- Can publish, store, and process data as it occurs
- Captures and delivers real-time data e.g. Stock markets

Rodzaje integracji

- Dodatkowe elementy
 - Translacja modeli
 - Wydajność systemu
 - Warstwa abstrakcji i wersjonowanie modeli
 - Gwarancja dostarczenia (MQ vs. WS)
 - Synchroniczna vs. Asynchroniczna
 - Znaczenie correlation ID
 - Naliczanie opłat za subskrypcje

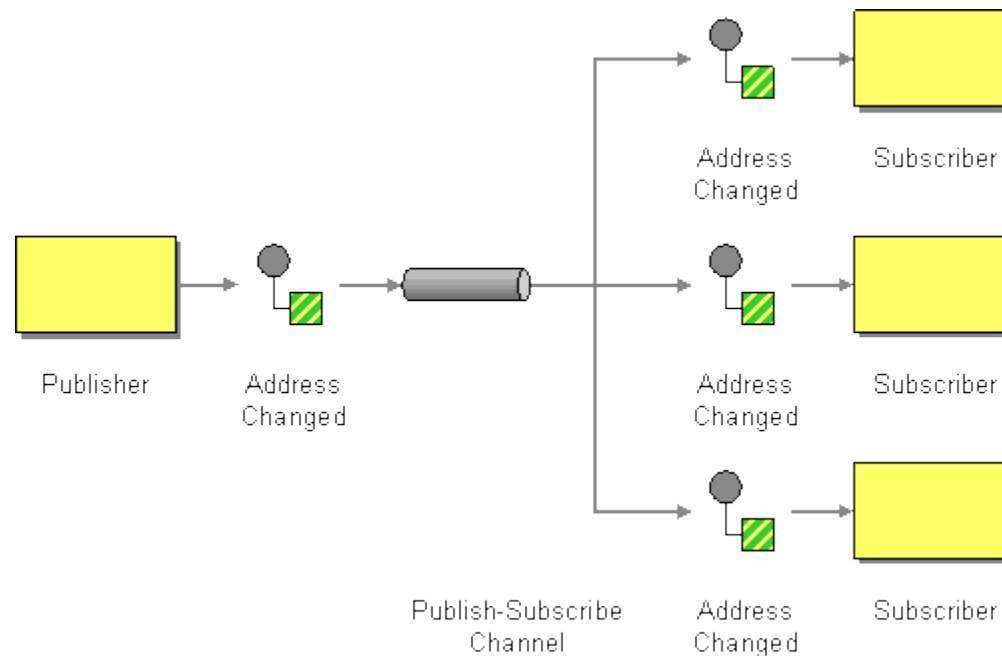
Messaging Patterns

- Publish/Subscribe
- Request/Reply
- Fire/Forget

- Realizacja
 - Synchroniczna
 - Asynchroniczna

Messaging Patterns

- Publish/Subscribe



Messaging Patterns

- Topic concept (example from MQTT)
 - Topics
 - house/room1/main-light
 - house/room1/side-light
 - house/room1/alarm
 - house/room2/main-light
 - house/room2/side-light
 - house/room2/alarm
 - house/garage/main-light
 - house/main-door

Messaging Patterns

- Topic concept (example from MQTT)

- Subscriptions examples

- house/#

- house/room1/main-light
 - house/room1/side-light
 - house/room2/alarm
 - house/garage/main-light
 - Etc.

- house+/main-light

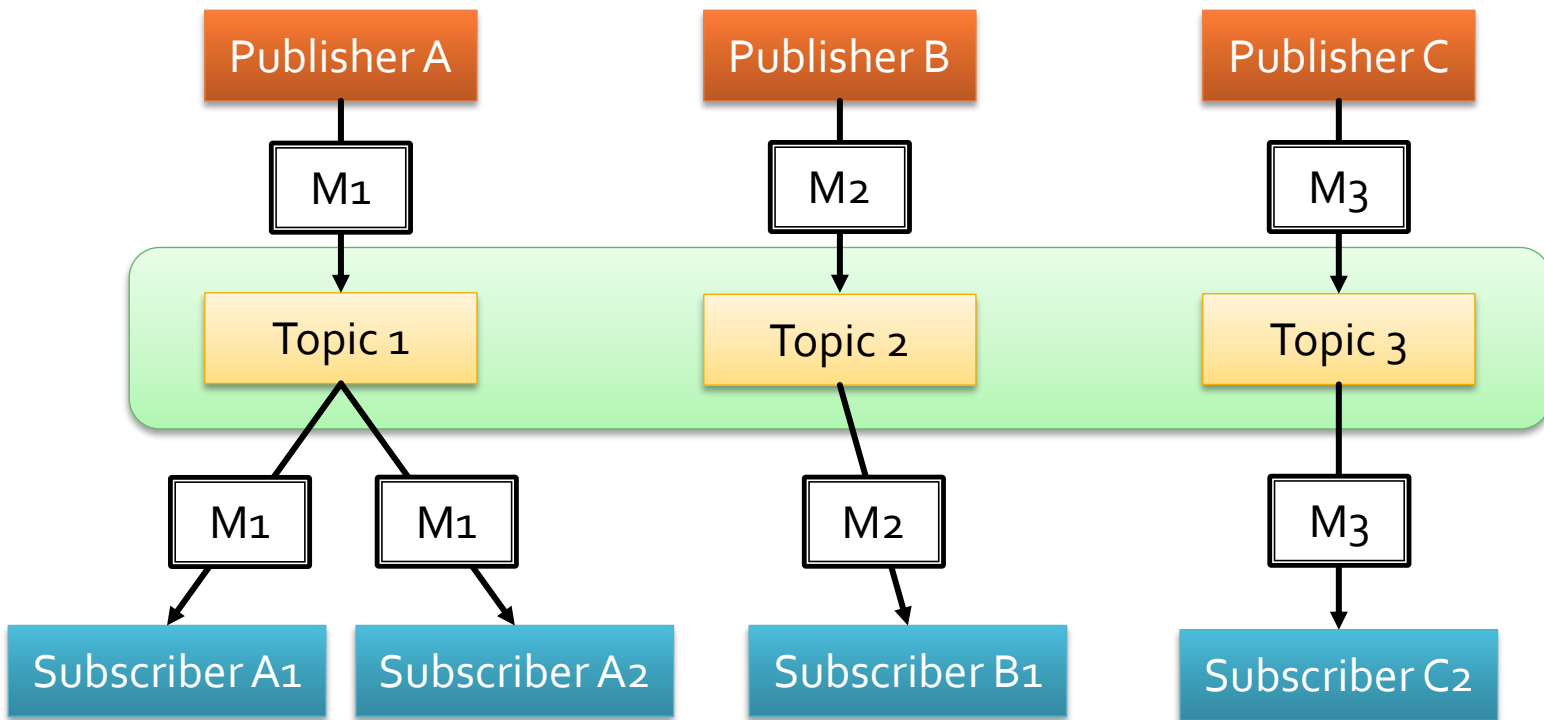
- house/room1/main-light
 - house/room2/main-light
 - house/garage/main-light

- multi level wildcard

+ - single level wildcard

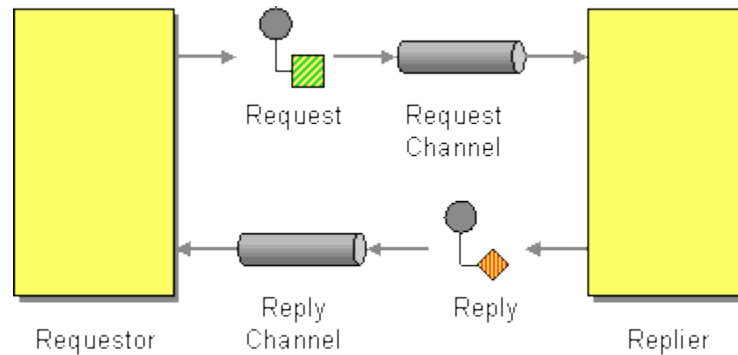
Messaging Patterns

- Topic concept



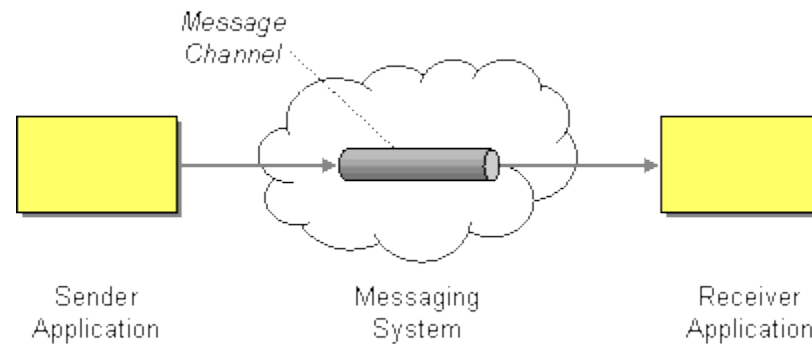
Messaging Patterns

- Request/Reply



Messaging Patterns

- Fire/forget (request only)

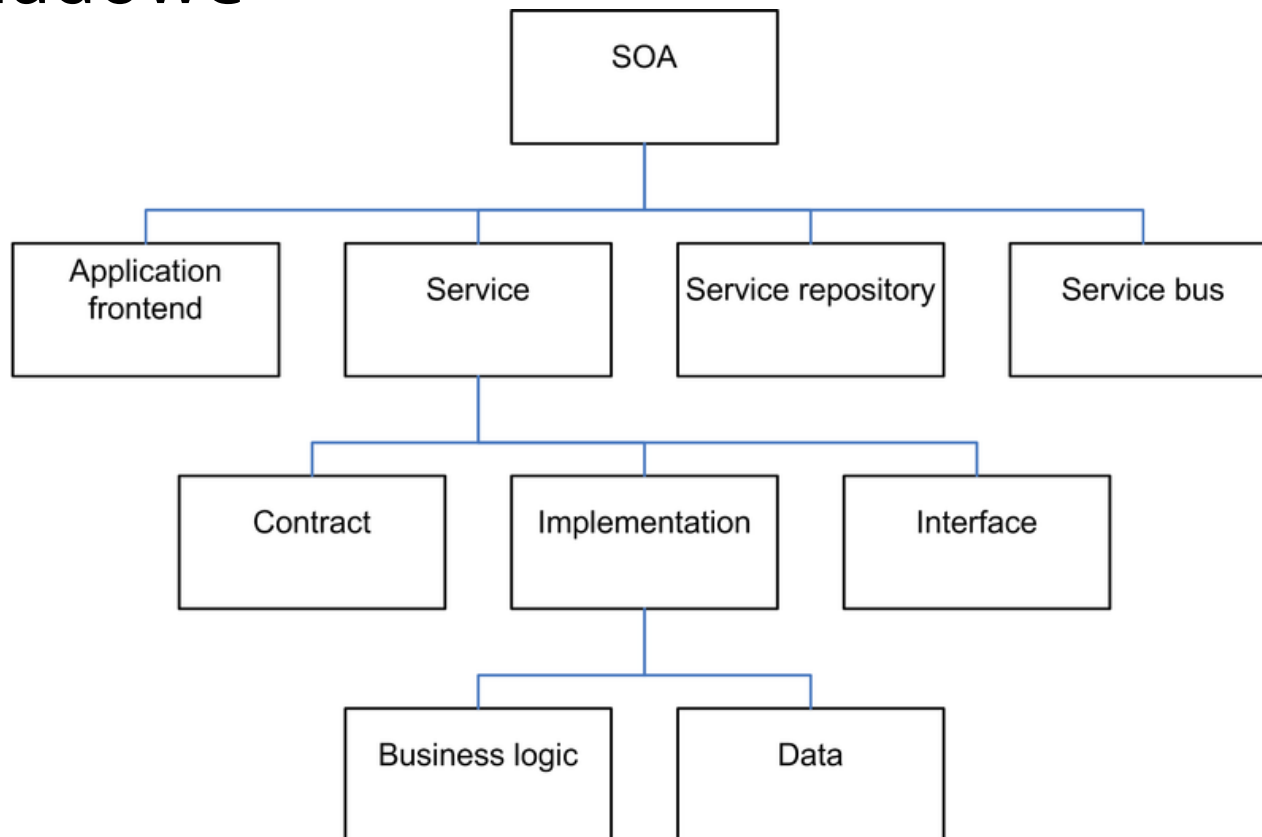


SOA

- SOA czyli Service Oriented Architecture
 - Podział systemu na komponenty, które udostępniają swoją funkcjonalności poprzez określony interfejs
 - oraz z wykorzystaniem standardów dot. komunikacji
 - Kilka własności
 - Stanowi fasadę dla funkcjonalności, które udostępnia
 - Jest taką „czarną skrzynką” o określonym API
 - Po wykonaniu każdej operacji powinien być spójny (transakcje)
 - Powinien być zabezpieczony i odporny na błędne użycie
- Uwaga: WebServices i SOA to nie to samo
 - Chociaż można zaimplementować SOA za pomocą WS

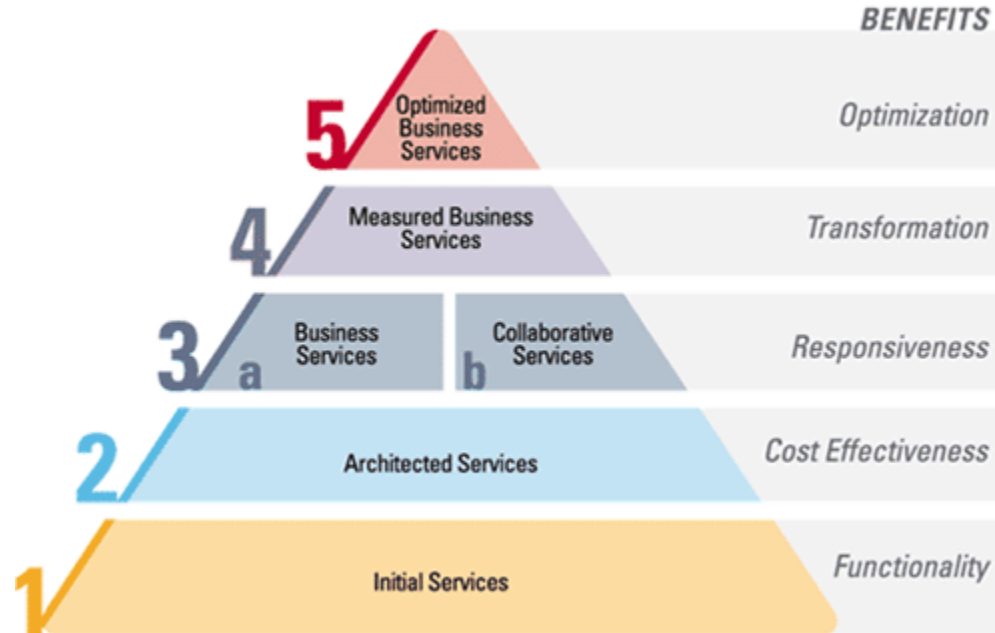
SOA

■ Składowe



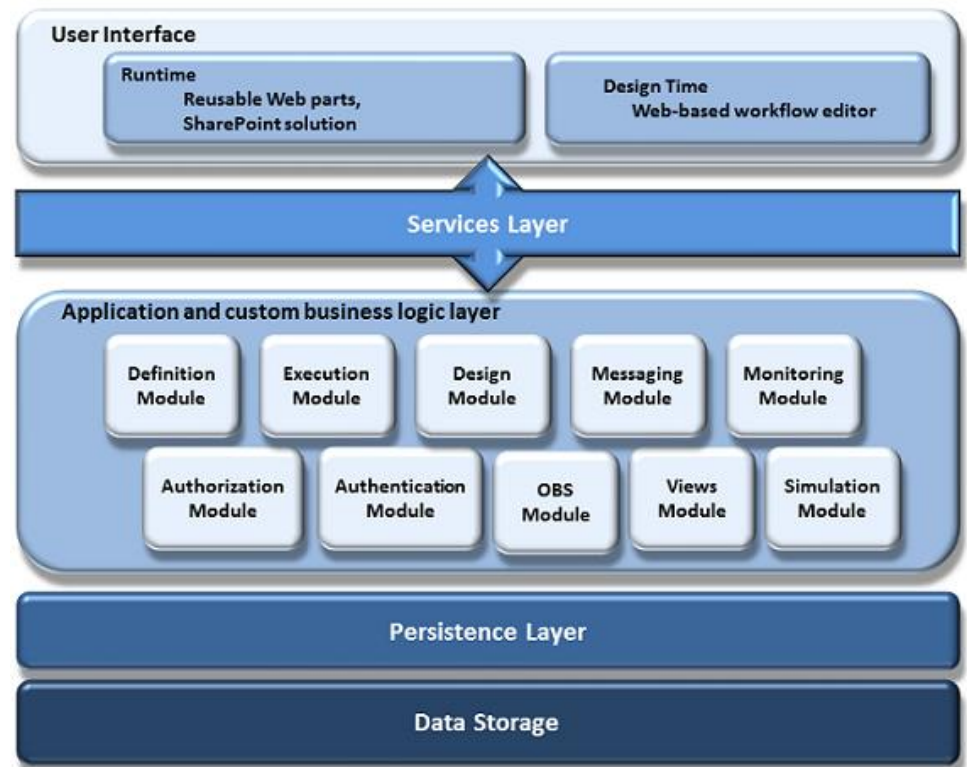
SOA

- Dojrzałość zastosowania usług



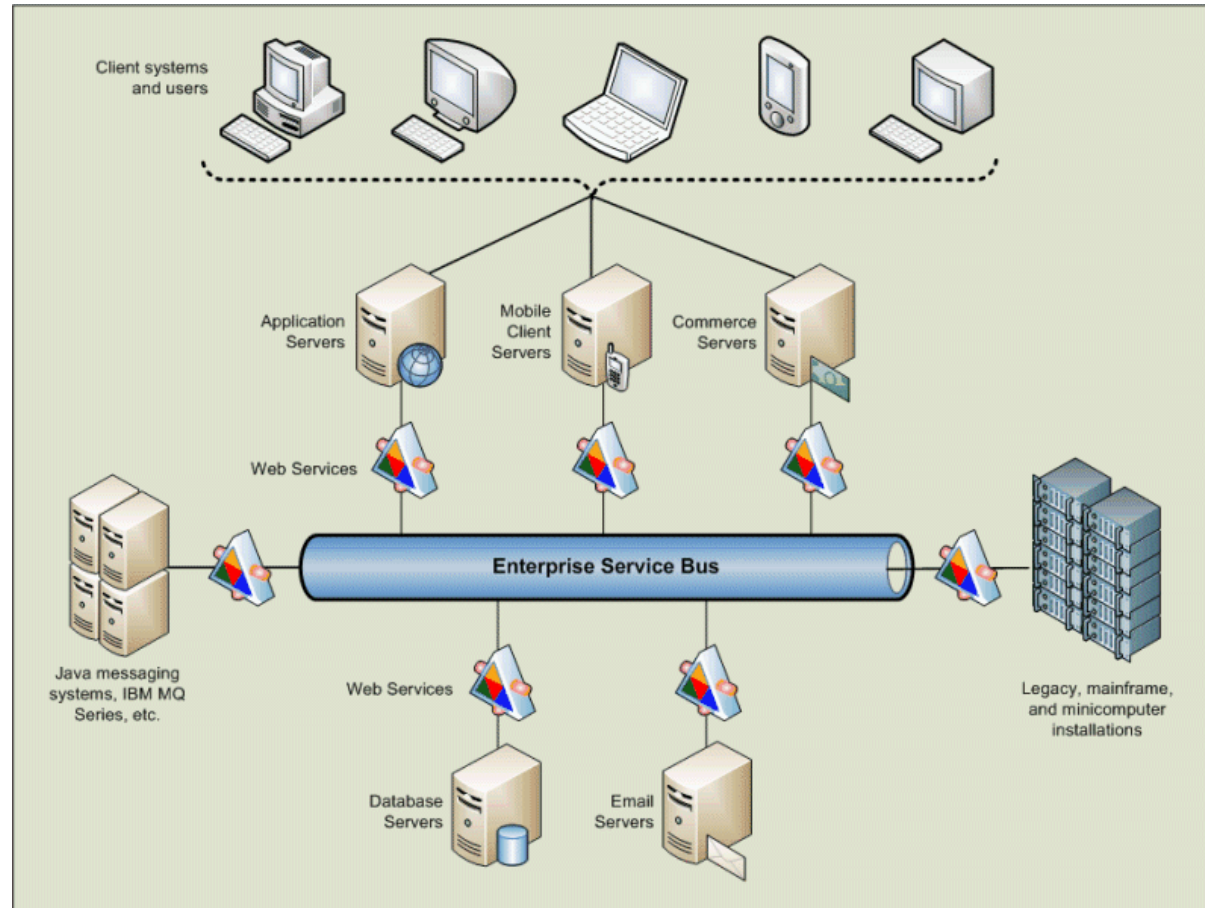
SOA

- Główne zyski z zastosowania SOA
 - Abstraction
 - Autonomy
 - Formal contract
 - Loose coupling
 - Reusability
 - Statelessness



Implementacja SOA

- WebServices
- ESB



API Management

- Definition

API management is the process of creating and publishing web application programming interfaces (APIs), enforcing their usage policies, controlling access, nurturing the subscriber community, collecting and analyzing usage statistics, and reporting on performance.

Wikipedia

- Usually supported COTS products

- e.g. IBM API Connect, ApiGee, MS Azure API Mgmt

- Mockup tools:

- e.g. Mockup.io or Beeceptor.com

API Management

- Common functionalities
 - Gateway
 - Versioning
 - Monitoring
 - Access management
 - Publishing tools
 - Reporting and analytics
 - Monetization and cost control
 - Developers Portal

REST

- Stands for **R**epresentational **S**tate **T**ransfer
- Architectural style
- Closely based on HTTP
- Key features
 - Stateless
 - Client–server
 - Uniform interface
 - Cacheable
 - Layered system

REST

- Key element: Uniform Interface
- Main constituents:
 - Resources structure
 - Manipulation through representations
 - Self-descriptive messages
 - e.g. by adding media types information
 - HATEOAS
 - i.e. hypermedia as the engine of application state

REST

- Resource structure
 - Allows to access resources
 - Uniquely identify resources
 - Examples
 - `http://ex.com/customers`
 - `http://ex.com/customers/<id>/orders`
 - `http://ex.com/customers/<cid>/orders/<oid>/status`

REST

- Representation
 - Resource state at point in time
 - Resource may have a few representations
 - Representation Metadata describes representation
 - Helps client and server with understanding how to process data (structure, meaning)
 - Content negotiation is a process of selecting the representation
 - Accept (e.g. **application/json**, but also **image/png**)
 - Accept-Encoding
 - Accept-Language
 - Accept-Charset

REST

- Common interpretation of HTTP methods
 - GET – retrieve
 - POST – create
 - PUT – full entity update
 - PATCH – delta update
 - DELETE – delete

REST

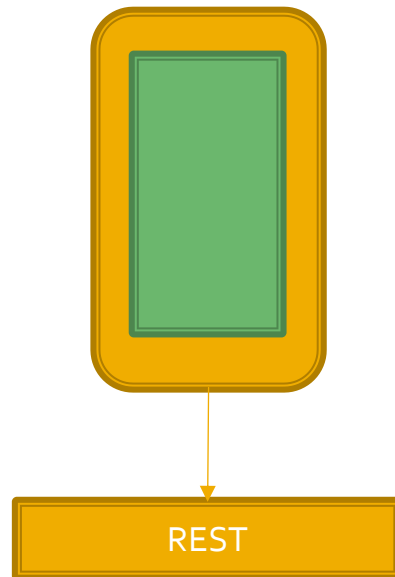
- HTTP Status Codes
 - Very important to use it
 - Several most common ones
 - 200 Ok
 - 201 Created
 - 204 No content
 - 400 Bad request
 - 401 Unauthorized
 - 404 Not found
 - More
 - <http://www.restapitutorial.com/httpstatuscodes.html>

REST

- More will be on a dedicated presentation...

How to fetch data from API?

- Let's review OData and GraphQL



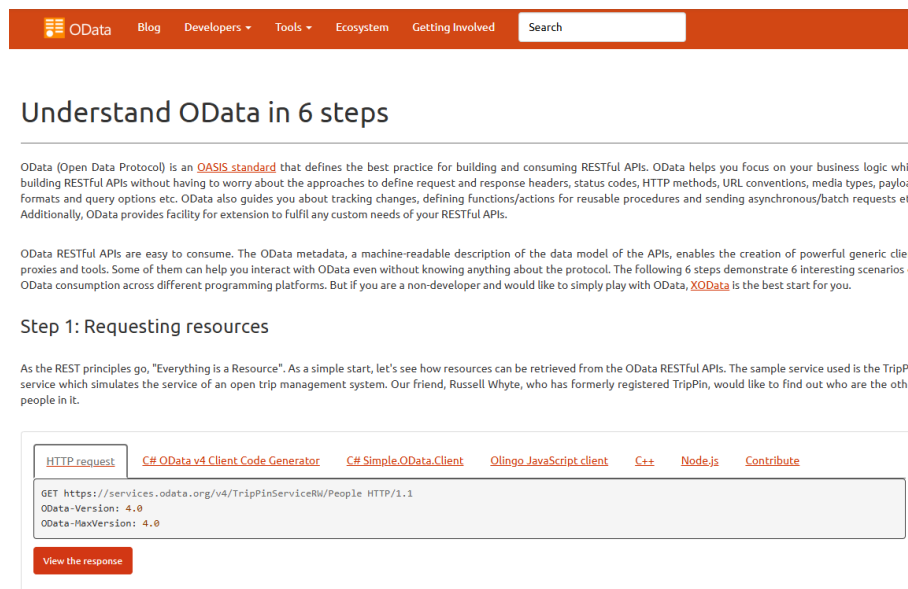
- 5 page
- fit into screen
- items starts with „M”
- sort by price (from max)

OData

- Open protocol based on REST
- OASIS standard
- Supports more flexible
- Main constituents
 - OData query syntax: URLs structure
 - OData Formatting:
 - ATOM Publishing Protocol (XML) vs. JSON
 - CRUD support
 - Metadata: [http://odataserwis.pl/\\$metadata](http://odataserwis.pl/$metadata)
- Web site: www.odata.org

OData quick overview

- <http://www.odata.org/getting-started/understand-odata-in-6-steps/>



OData (Open Data Protocol) is an [OASIS standard](#) that defines the best practice for building and consuming RESTful APIs. OData helps you focus on your business logic while building RESTful APIs without having to worry about the approaches to define request and response headers, status codes, HTTP methods, URL conventions, media types, payload formats and query options etc. OData also guides you about tracking changes, defining functions/actions for reusable procedures and sending asynchronous/batch requests etc. Additionally, OData provides facility for extension to fulfil any custom needs of your RESTful APIs.

OData RESTful APIs are easy to consume. The OData metadata, a machine-readable description of the data model of the APIs, enables the creation of powerful generic client proxies and tools. Some of them can help you interact with OData even without knowing anything about the protocol. The following 6 steps demonstrate 6 interesting scenarios of OData consumption across different programming platforms. But if you are a non-developer and would like to simply play with OData, [XOData](#) is the best start for you.


Step 1: Requesting resources

As the REST principles go, "Everything is a Resource". As a simple start, let's see how resources can be retrieved from the OData RESTful APIs. The sample service used is the TripPin service which simulates the service of an open trip management system. Our friend, Russell Whyte, who has formerly registered TripPin, would like to find out who are the other people in it.

```
HTTP request: C# OData v4 Client Code Generator C# Simple.OData.Client Olingo JavaScript client C++ Node.js Contribute
GET https://services.odata.org/v4/TripPinServiceRM/People HTTP/1.1
OData-Version: 4.0
OData-MaxVersion: 4.0
View the response
```

Step 2: Requesting an individual resource

REST principles also say, that every resource is identified by a unique identifier. OData also enables defining key properties of a resource and retrieving it using the keys. In this step, Russell wants to find the information about himself by specifying his username as the key.



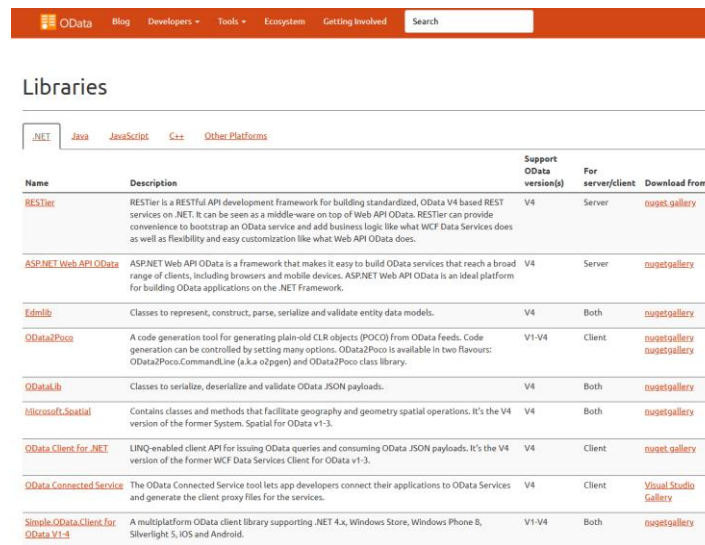
```
HTTP request: C# OData v4 Client Code Generator C# Simple.OData.Client Olingo JavaScript client C++ Node.js Contribute
GET https://services.odata.org/v4/TripPinServiceRM/People('russellwhyte') HTTP/1.1
OData-Version: 4.0
OData-MaxVersion: 4.0
View the response
```

OData query syntax

- Zapytania
 - \$filter, \$orderby, \$top, \$skip, \$select, \$expand
 - \$inlinecount=allpages - liczba wszystkich rekordów
 - nawet jeśli użyjemy top, skip
- Operatory logiczne
 - and, or, not, eq, ne, lt, gt, le, ge
- Operatory arytmetyczne
 - add, sub, mul, div, mod
- Funkcje napisów
 - substring,length,indexof,startswith, endswith, insert, remove, replace,tolower,toupper,concat,trim
- Funkcje daty
 - day,hour,minute,month,second,year
- Funkcje matematyczne
 - add, sub, mult, div, round,floor,ceiling

OData support

- Many libraries, tools, and more:
 - See <https://www.odata.org/ecosystem/>
- Specifically, the software dev. support:
 - See: <https://www.odata.org/libraries/>

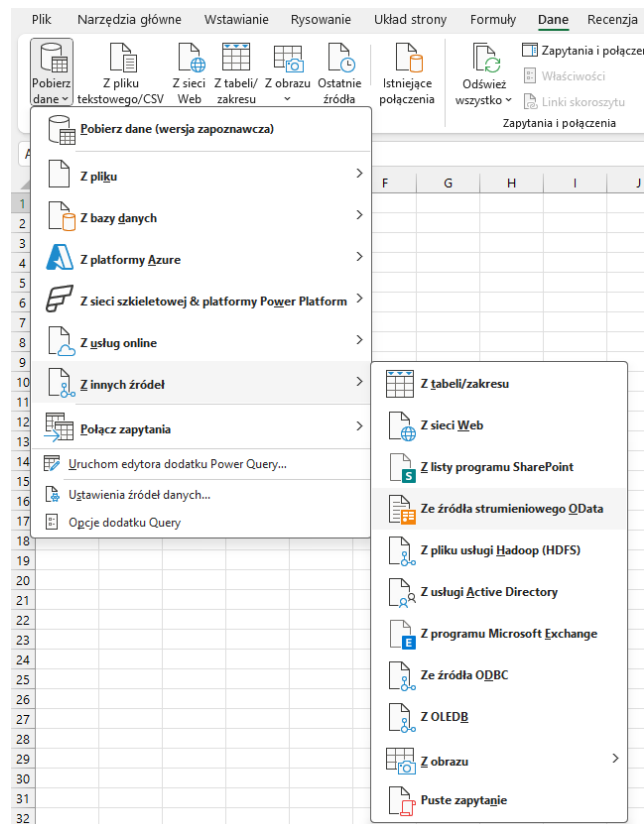


The screenshot shows the 'Libraries' page on the OData website. The page has a navigation bar with 'OData', 'Blog', 'Developers', 'Tools', 'Ecosystem', and 'Getting Involved' links, along with a search box. Below the navigation bar, the 'Libraries' section is displayed with a filter for '.NET' selected. A table lists various OData libraries with columns for Name, Description, Support OData version(s), For server/client, and Download from.

Name	Description	Support OData version(s)	For server/client	Download from
RESTler	RESTler is a RESTful API development framework for building standardized, OData V4 based REST services on .NET. It can be seen as a middle-ware on top of Web API OData. RESTler can provide convenience to bootstrap an OData service and add business logic like what WCF Data Services does as well as flexibility and easy customization like what Web API OData does.	V4	Server	nuget gallery
ASP.NET Web API OData	ASP.NET Web API OData is a framework that makes it easy to build OData services that reach a broad range of clients, including browsers and mobile devices. ASP.NET Web API OData is an ideal platform for building OData applications on the .NET Framework.	V4	Server	nuget gallery
Edmlib	Classes to represent, construct, parse, serialize and validate entity data models.	V4	Both	nuget gallery
OData2Poco	A code generation tool for generating plain-old CLR objects (POCO) from OData feeds. Code generation can be controlled by setting many options. OData2Poco is available in two flavours: OData2Poco.CommandLine (x.k.a o2pgen) and OData2Poco class library.	V1-V4	Client	nuget gallery nuget gallery
ODataLib	Classes to serialize, deserialize and validate OData JSON payloads.	V4	Both	nuget gallery
Microsoft.Spatial	Contains classes and methods that facilitate geography and geometry spatial operations. It's the V4 version of the former System.Spatial for OData v1-3.	V4	Both	nuget gallery
OData Client for .NET	LINQ-enabled client API for issuing OData queries and consuming OData JSON payloads. It's the V4 version of the former WCF Data Services Client for OData v1-3.	V4	Client	nuget gallery
OData.Connected.Service	The OData Connected Service tool lets app developers connect their applications to OData Services and generate the client proxy files for the services.	V4	Client	Visual Studio Gallery
Simple.OData.Client.for.OData.V1-4	A multiplatform OData client library supporting .NET 4.x, Windows Store, Windows Phone 8, Silverlight 5, IOS and Android.	V1-V4	Both	nuget gallery

DEMO

- Loading data to Excel
 - <https://services.odata.org/AdventureWorksV3/AdventureWorks.svc/>



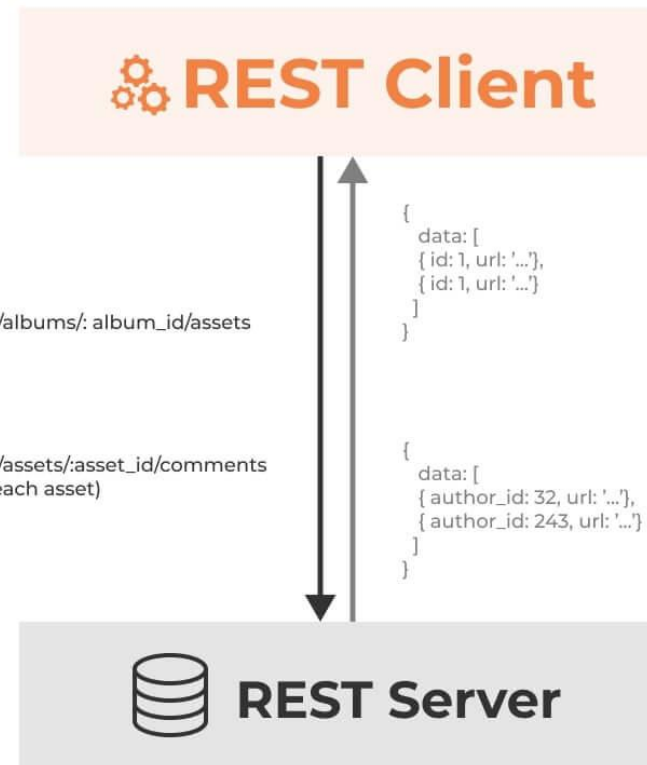
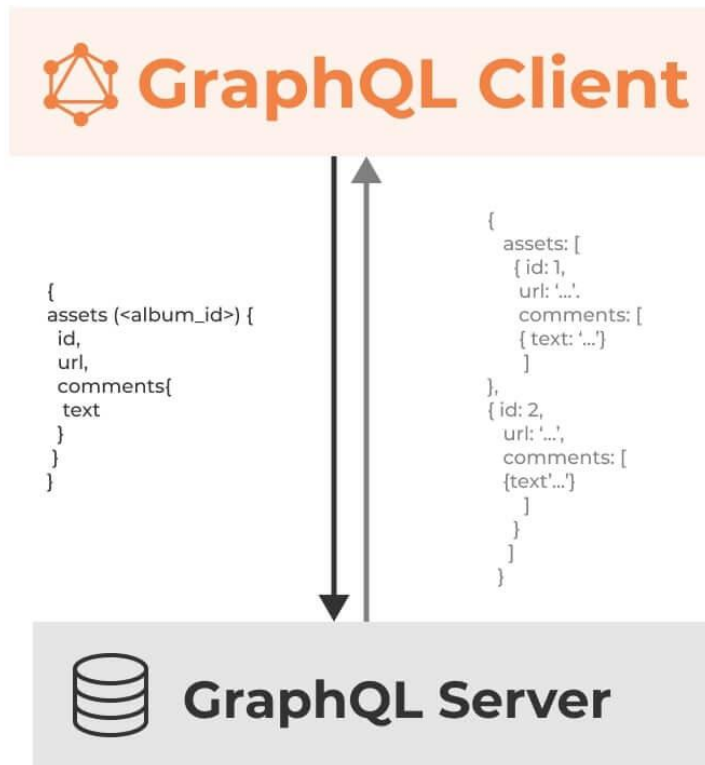
OData getting started

- Example service:
 - <https://services.odata.org/TripPinRESTierService/>
- Getting started:
 - <https://www.odata.org/getting-started/>

GraphQL

- A query language to for getting data from the remote services
- Strongly typed
- Created by Facebook in 2012 to improve fetching data in their services
- Published as open source in 2015
- Website: <https://graphql.org/>

Concept



GraphQL vs REST comparison



GraphQL



REST

	GraphQL	REST
Architecture	client-driven	server-driven
Organized in terms of	schema & type system	endpoints
Operations	Query Mutation Subscription	Create, Read, Update, Delete
Data fetching	specific data with a single API call	fixed data with multiple API calls
Community	growing	large
Performance	fast	multiple network calls take up more time
Development speed	rapid	slower
Learning curve	difficult	moderate
Self-documenting	✓	—
File uploading	—	✓
Web caching	(via libraries built on top)	✓
Stability	less error prone, automatic validation and type checking	better choice for complex queries
Use cases	multiple microservices, mobile apps	simple apps, resource-driven apps

GraphQL vs OData comparison

GraphQL	OData
Quite complex to understand	Easy to understand
Supports subscriptions which is more powerful than Delta feeds	Supports Delta Feeds (way for providing the client-specific link without being stateful)
Easy to Scale using just a single data graph	Hard to build and maintain in terms of Scaling
Broader adoption by big companies	Open standard used by companies like SAP, .NET, etc
Self-contained and powerful querying capabilities than OData	Simpler concept but less powerful than GraphQL
Comparatively harder to implement and is not RESTful, but provides more facilities.	Simple to implement as it provides a simpler concept
Uses HTTP with POST request only to perform operations	Uses POST, PUT, GET, DELETE to perform operations
Supports versioning, joins, selects, etc., from request/ response round trip	Supports joins and relations, but it does not support the versioning
User has to do a lot of work like creating the schema, query, creating types for each query	Provides a simpler definition for queries with easy and out of the box integration to migrate with the already existing webAPIs

Defining the schema

```
type User {
  username: String! @id
  displayName: String
  avatarImg: String
  posts: [Post!]
  comments: [Comment!]
}

type Post {
  id: ID!
  title: String! @search(by: [term])
  text: String! @search(by: [fulltext])
  tags: String @search(by: [term])
  datePublished: DateTime
  author: User! @hasInverse(field: posts)
  category: Category! @hasInverse(field: posts)
  comments: [Comment!]
}
```

```
type Comment {
  id: ID!
  text: String!
  commentsOn: Post! @hasInverse(field: comments)
  author: User! @hasInverse(field: comments)
}

type Category {
  id: ID!
  name: String! @search(by: [term])
  posts: [Post!]
}
```

Query Example

```
1 ▾ query MyQuery {  
2   ▾ movies(first: 10, orderBy: id_ASC) {  
3     title  
4     imdbId  
5     stage  
6   ▾ createdBy {  
7     name  
8     id  
9   }  
10 }  
11 }
```

How to start

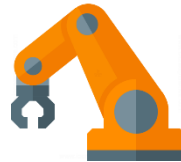
- Introduction:

- <https://bulldogjob.pl/readme/jak-zaczac-z-graphql>
- <https://www.youtube.com/watch?v=Yze5DcjMCzM>
- <https://graphql.org/learn/>
- <https://www.apollographql.com/docs/apollo-server/getting-started>

- Playgrounds:

- <https://hygraph.com/graphql-playground>
- <https://www.devtoolsdaily.com/graphql/playground/>

Przykład



API Management / ESB

High Performance Data
Hub

Cache

Microservice

Microservice

Fast DB

Microservice

Microservice

Integration Middleware

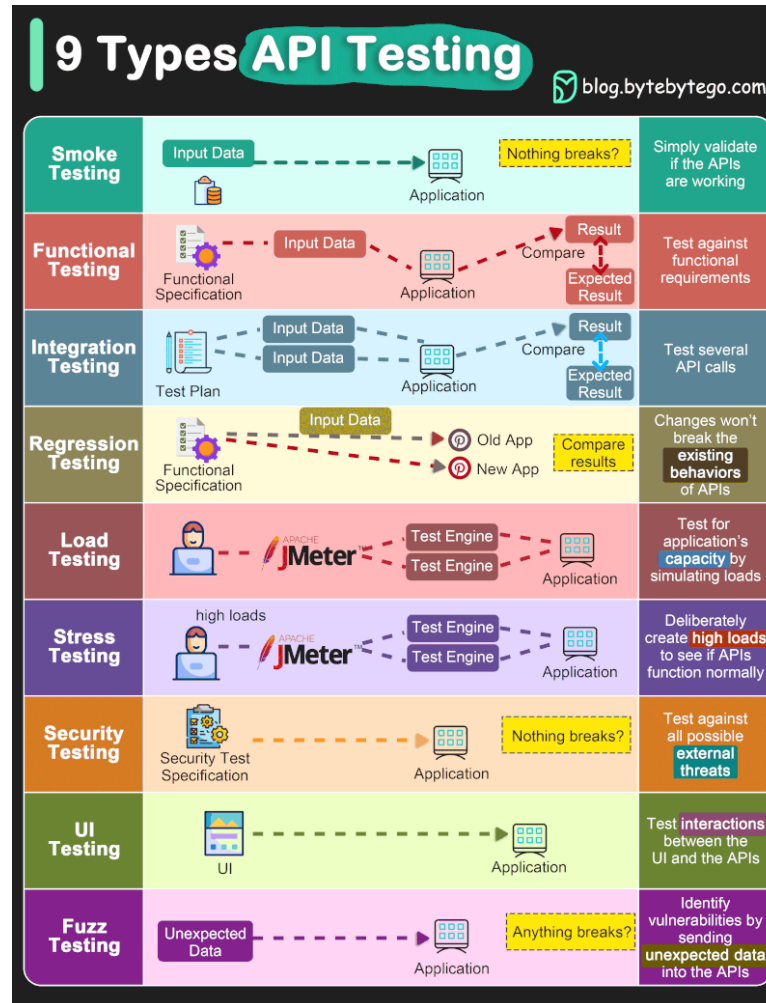
Ordering
system

Product
Information
Management

Data Asset
Management

Translation
Services

Testowanie API

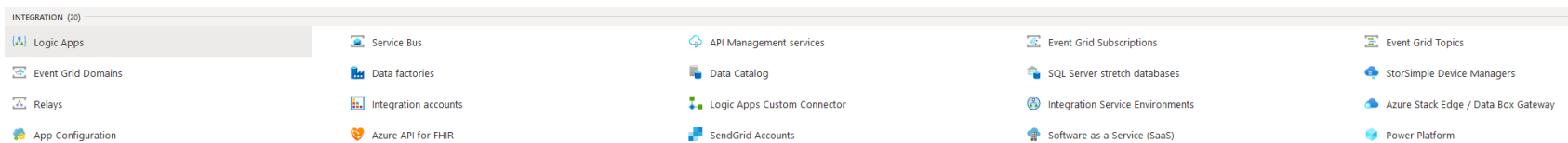


Testowanie API

- Postman

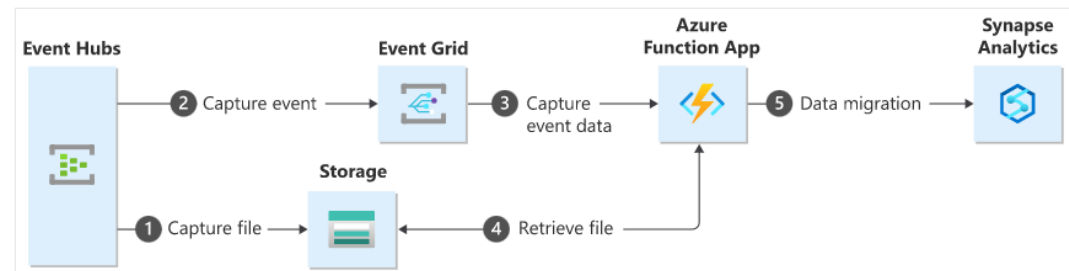
Azure Integration Services

- Wiele różnych usług w kategorii integracji



- Event vs. Message
- Główne Azure messaging services

- Event Hub
- Event Grid
- Service Bus



- Rola API Management Services

<https://learn.microsoft.com/pl-pl/azure/service-bus-messaging/compare-messaging-services>

<https://www.youtube.com/watch?v=Nx866vYoVgo>

Demo

- AzureServiceBus
- Let's review:

<http://www.eaipatterns.com/toc.html>

Literatura

■ SOA

- http://en.wikipedia.org/wiki/Service-oriented_architecture
- <http://www.pnmsoft.com/resources/bpm-tutorial/soa-tutorial/>
- http://www.enterprise-architecture.info/EA_Services-Oriented-Enterprise.htm
- <http://www.soapatterns.org/>
- <http://www.slideshare.net/SubhaPrasad/soa-14718016>

■ ESB

- <http://msdn.microsoft.com/en-us/library/ff648282.aspx>
- http://en.wikipedia.org/wiki/Enterprise_service_bus

■ EIP

- <http://www.eaipatterns.com/>
- https://access.redhat.com/site/documentation/en-US/Fuse_ESB/4.3.1/html-single/Implementing_Enterprise_Integration_Patterns/index.html

■ Messaging Patterns

- <http://msdn.microsoft.com/en-us/library/ff649664.aspx>
- <http://msdn.microsoft.com/en-us/library/aa480027.aspx>
- <http://camel.apache.org/async.html>