

Web Services

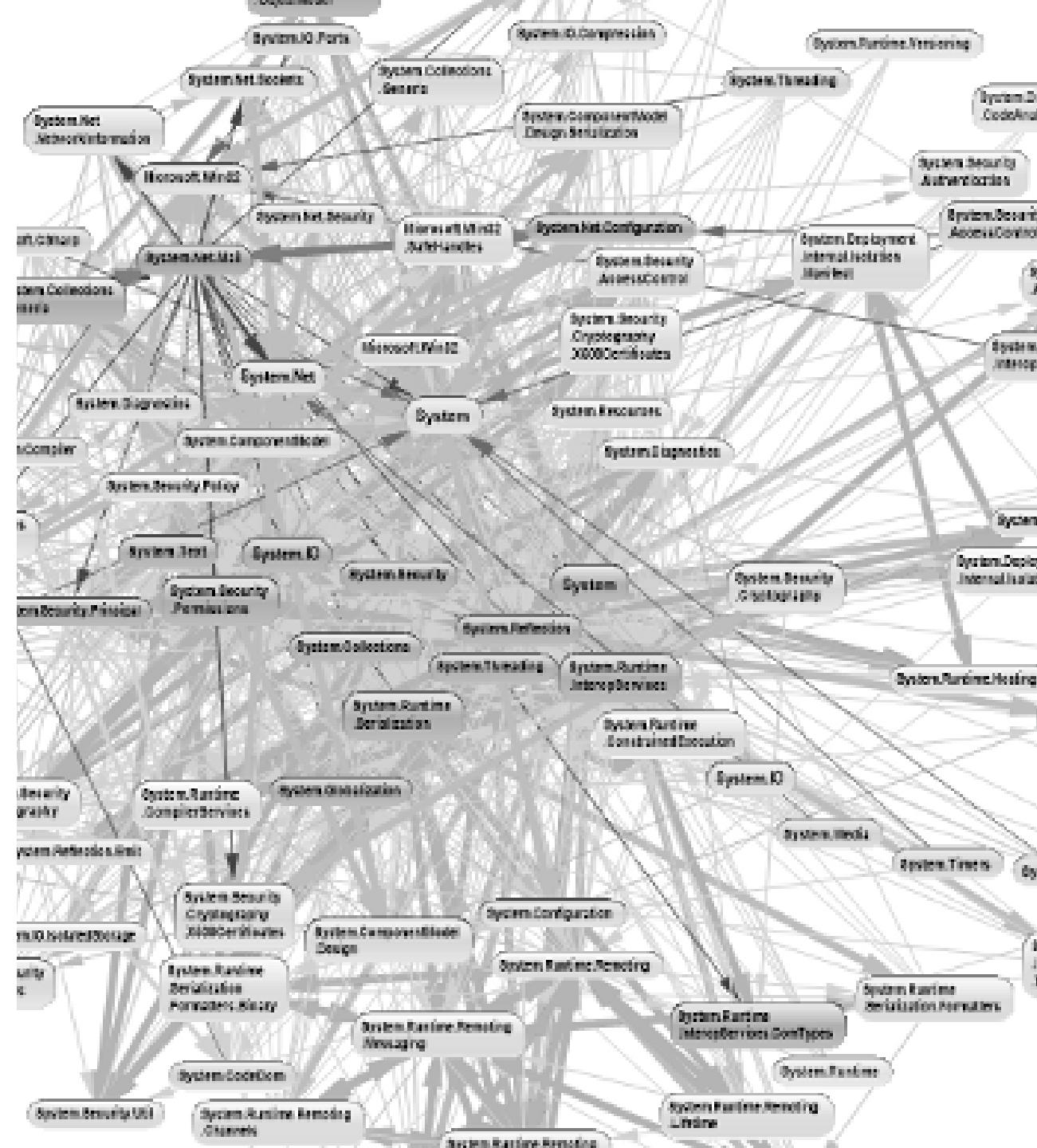
Agenda

- Introduction
- Web Services
- SOA
- API
- Restcountries API - *Code*
- Spotify API - *Code*
- ToDo FastAPI App - *Demo*

Why We Need Software Architecture

Architecture-less software becomes **unmanageable** with time and hence enhance the maintenance cost drastically with every new iteration.

As each and every change becomes costlier, this approach is termed as **Big Ball of Mud**



In your programs, have you thought
about **Software Architecture**?

If **Yes**, in what way?

If **No**, is it something you missed?



Architectural approaches

Over the years of evolutions in software design, developers have come up with different architectural approaches in order to avoid the issues of architecture less software design - **Big Ball of Mud**.

The most *famous* ones.

- *Layered Architecture*
- *Tiered Architecture*
- Service Oriented Architecture (SOA)

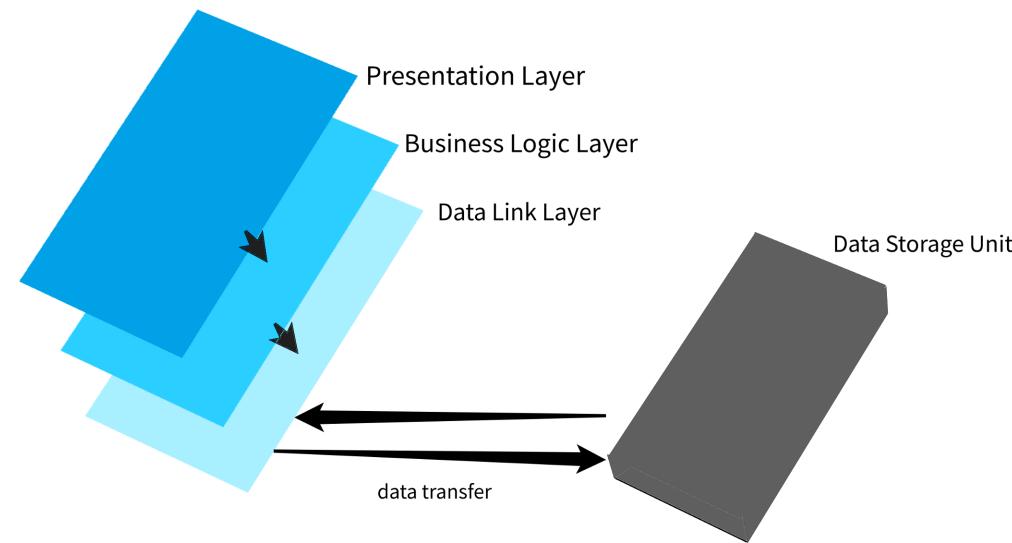
Layered Architecture

This approach works on principle of separation of concerns.

Software design is **divided into layer** laid over one another. Each layer performs a **dedicated responsibility**.

Architecture divides the software into the following layers

- Presentation Layer
- Business Logic Layer
- Data Link Layer



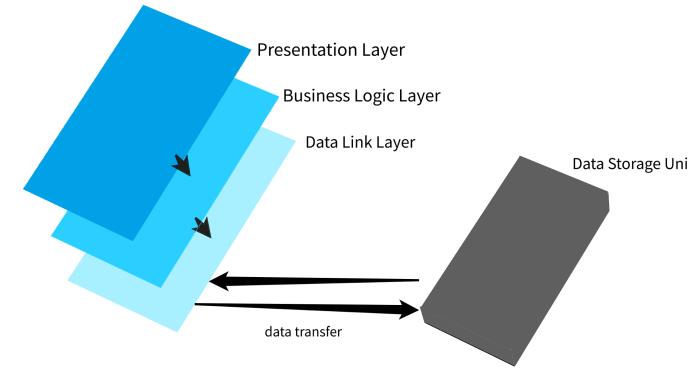
Layered Architecture

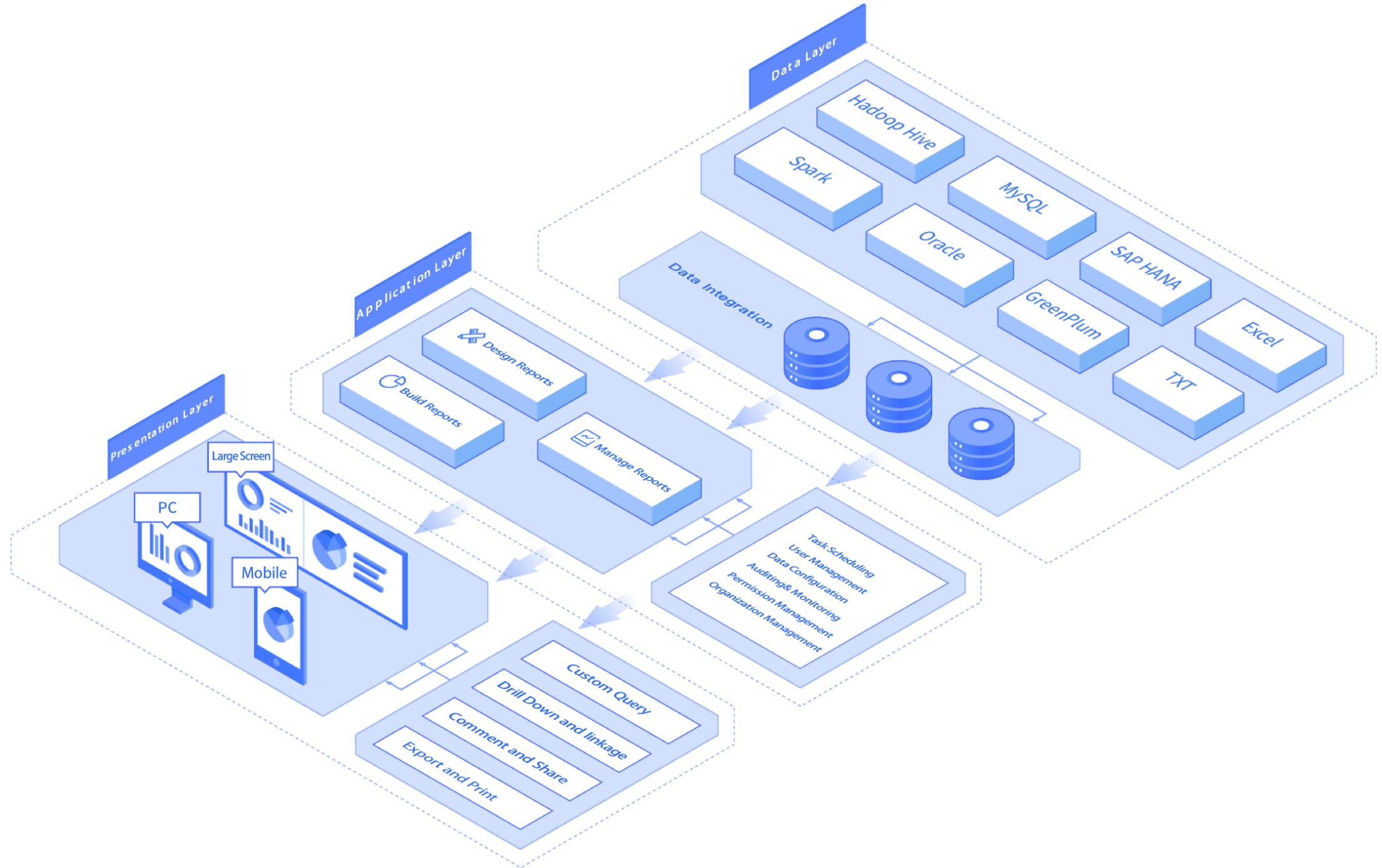
Advantages

- Simpler to implement
- Abstraction due to separation of concerns among layers
- Isolation between layers
- More manageable due to **low coupling**

Disadvantage

- Less scalability
- Monolith structure, lacking ease of modifications
- Data has to flow from each layer one after another





Tiered Architecture

Divided the software into **into tiers** based on **client server communication principle**.

Can have **one, two or n-tiered** system separating the responsibilities among data provider and the consumer.

Single Tiered System

In this approach, **single system** is responsible to work **as client as well as server** and can offer ease of deployment eliminating the need of *Inter System Communications* (ISC).

This system are suitable **only for small scale single user application** and should not be used for multi user complex applications.

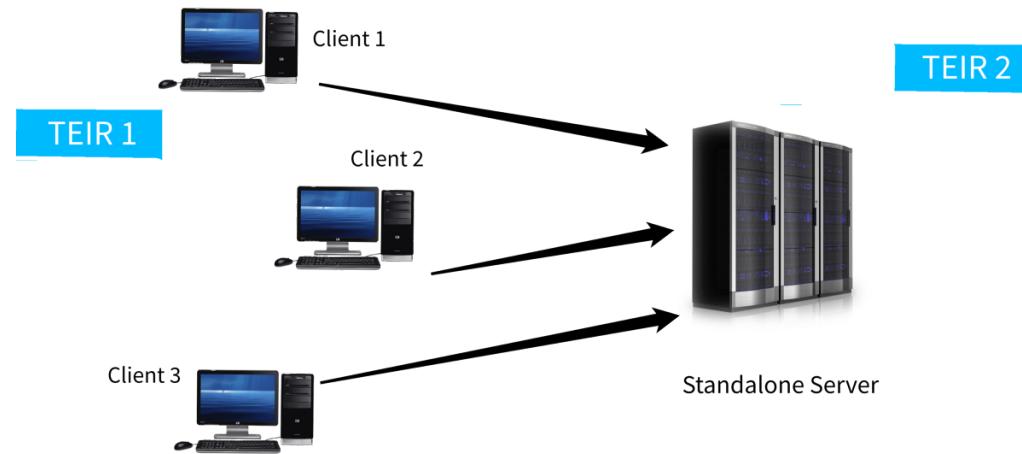
2-Tiered System

This system consist of two physical machines

- server
- client

It provides **isolation** among the **data management** operations and **data processing** and representation operations.

- *Client* holds **Presentation, Business Logic and Data link layer.**
- *Server* holds the **Data stores** such as **Databases**



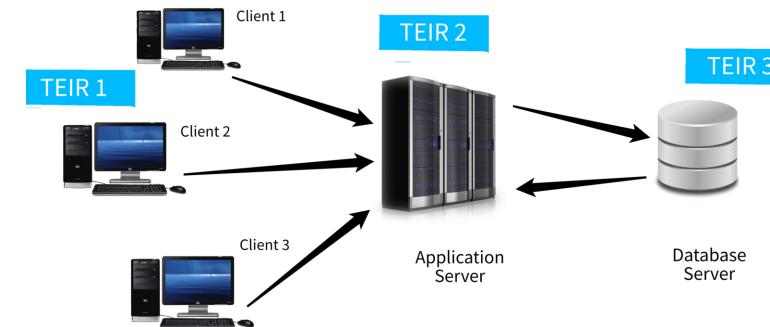
3-Tiered / n-Tiered System

Highly scalable both horizontally and vertically.

Implementing n-tiered architecture is generally **costlier** but offer high performance. Hence it is preferred in **large complex software solutions**.

It can be **combined** with advanced **Service Oriented Architectural** style to generate highly sophisticated model.

It is **recommended** to use this architecture when the software is **complex** and requires **performance** as well as **scaling** as it can be a costlier approach in terms of resources as well as time.



Difference between Layers and Tiers

Layer

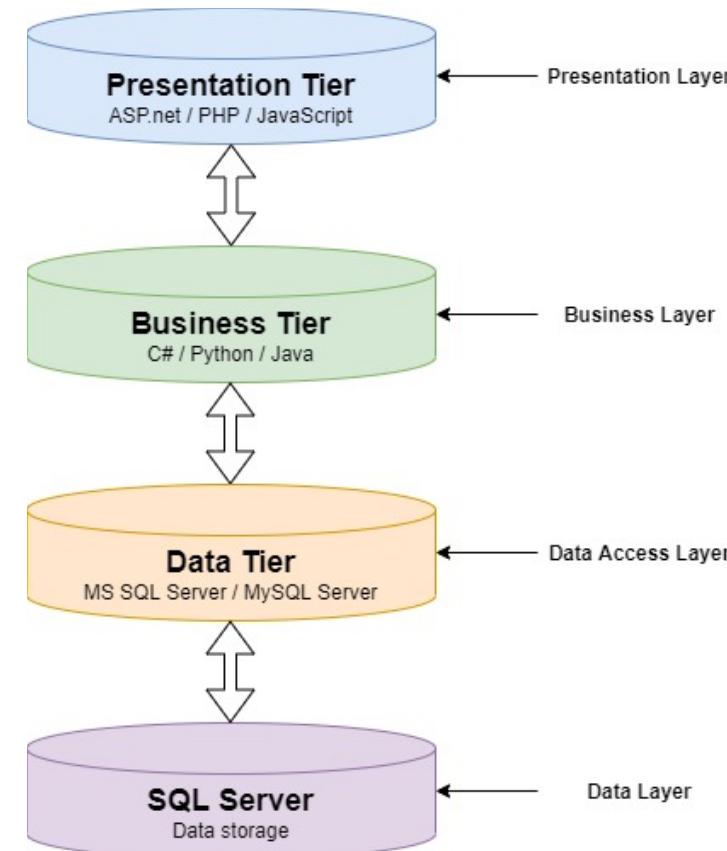
*Layers are the **logical** separation of code*

- Presentation Layer or *UI Layer*
- Business Layer or Business Logic Layer
- Data Access Layer and/or Data Layer

Tiers

*Tiers are the **physical** deployment of layers*

- Presentation Tier - *UI Tier*
- The Application Tier or Business Tier
- The Data Access Tier
- The Database Tier – *SQL Server, MySQL*



Web Services

Web Services

There are different definitions to Web Services - W3C is using:

A web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL).

Other systems interact with the web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other web-related standards.

For many, web services are synonymous with **SOA** (*Services Oriented Architecture*) and primarily rely on standards such as **XML-RPC** and **SOAP** (*Simple Object Access Protocol*).

Web Services Assignment

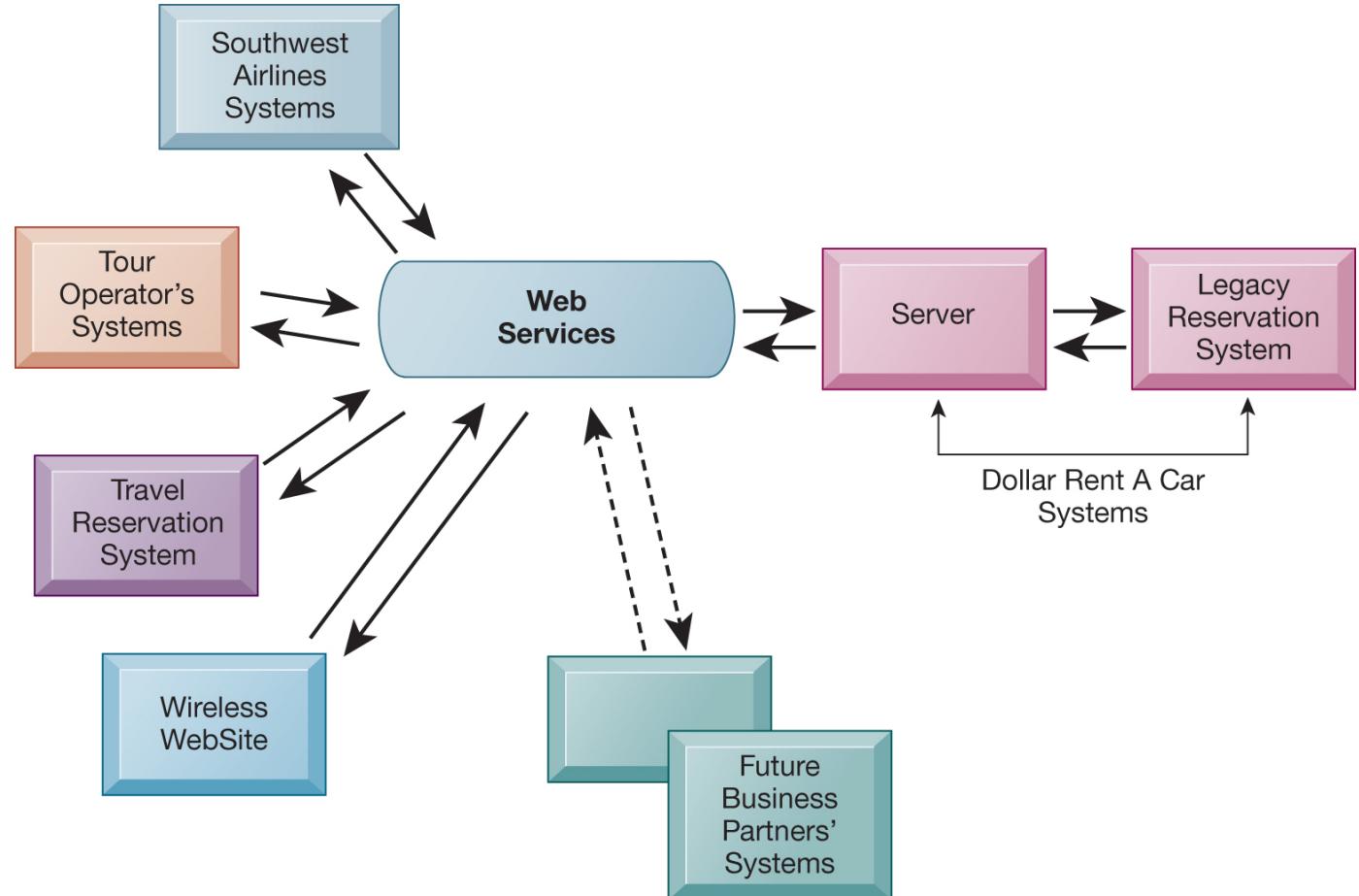
What are Web Services?

Find **3** uses of web services



Web Service example - Dollar Rent Car

The **frontend or presentation layer** can be in different programming languages and still have the ability to communicate with the Web Service.



What are Web Services?

- Web Services include any **software, application, or cloud technology** that provides standardized web protocols (*HTTP* or *HTTPS*) to **interoperate, communicate, and exchange data messaging**
- Applications can be written in **various languages** and are still able to communicate by **exchanging data** with one another via a web service between clients and servers.

Summarize - Web Service

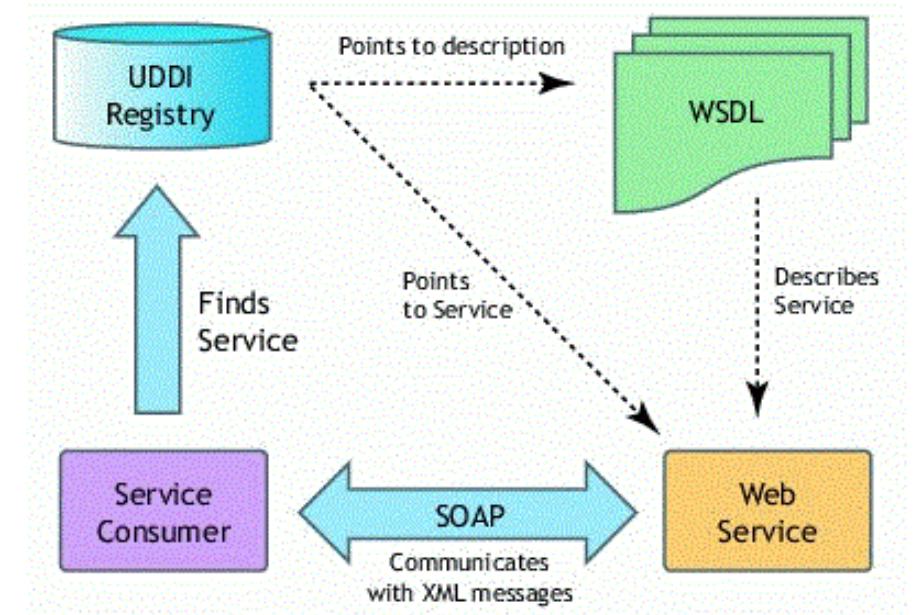
- Is available over the **Internet** or private (intranet) **networks**
- Uses a **standardized XML** messaging system
- Is **not tied** to any one operating system or programming language
- Is **self-describing** via a common **XML grammar**
- Is **discoverable** via a simple find mechanism

Components - Web Services

The basic web services platform is XML + HTTP

All the standard web services work using the following components:

- **SOAP**
Simple Object Access Protocol
- **UDDI**
Universal Description, Discovery and Integration
- **WSDL**
Web Services Description Language



How does a Web Service work?

A Web Service enables communication among various applications by using open standards such as **HTML**, **XML**, **WSDL**, and **SOAP**

- XML to **tag the data**
- SOAP to **transfer a message**
- WSDL to **describe the availability of service**

You can use **C#** to build a new **Web Services on Windows** that can be invoked from your web application that is based on **JavaServer Pages (JSP)** and runs on **Linux**.

XML

Extensible Markup Language (XML) is a **markup language** that defines a set of **rules for encoding documents** in a format that is both **human-readable** and **machine-readable**.

The World Wide Web Consortium's XML 1.0 Specification of 1998 and several other related specifications—all of them free open standards—define XML.

```
<?xml version="1.0"?>
<quiz>
  <qanda seq="1">
    <question>
      Who was the forty-second
      president of the U.S.A.?
    </question>
    <answer>
      William Jefferson Clinton
    </answer>
  </qanda>
  <!-- Note: We need to add
       more questions later.-->
</quiz>
```

XML

WSDL

- WSDL stands for Web Services Description Language
- WSDL is used to describe web services
- WSDL is written in XML
- WSDL is a W3C recommendation from 26.

June 2007

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="AktienKurs"
  targetNamespace="http://localhost:8080/AktienKurs"
  xmlns:xsd="http://schemas.xmlsoap.org/XMLSchema"
  xmlns="http://schemas.xmlsoap.org/wsdl">
  <service name="AktienKurs">
    <port name="AktienSoapPort" binding="tns:AktienSoapBinding">
      <soap:address location="http://localhost:8080/AktienKurs/AktienSoapPort" />
    </port>
    <message name="Aktie.HoleWert">
      <part name="body" element="xsd:TypeA" />
    </message>
    ...
  </service>
</definitions>
```

WSDL

SOAP

- SOAP stands for Simple Object Access Protocol
- SOAP is an application communication protocol
- SOAP is a format for sending and receiving messages
- SOAP is platform independent
- SOAP is based on XML
- SOAP is a W3C recommendation

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 299
SOAPAction: "http://www.w3.org/2003/05/soap-envelope"

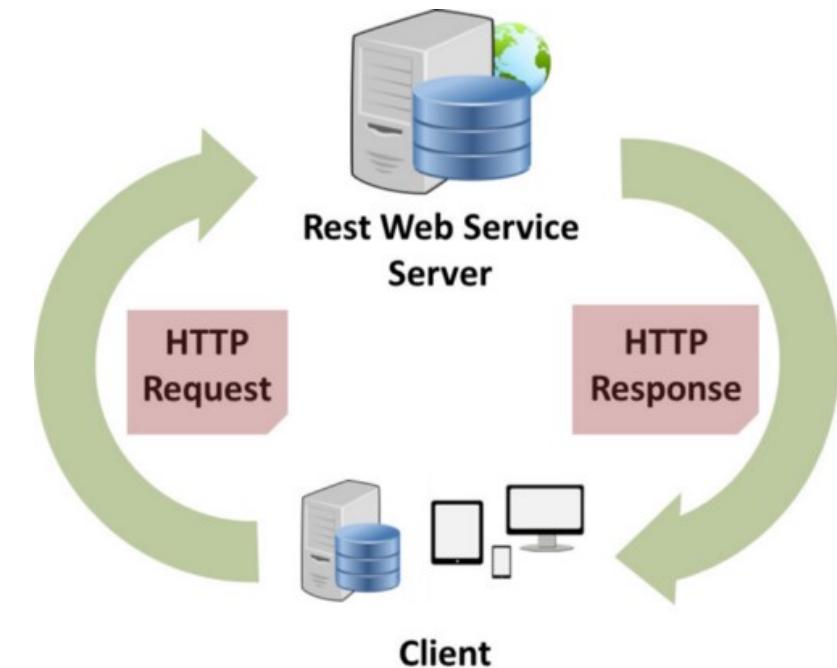
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:m="http://www.example.org">
  <soap:Header>
  </soap:Header>
  <soap:Body>
    <m:GetStockPrice>
      <m:StockName>T</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```

REST - Representational State Transfer

In 2000 Roy Thomas Fielding defines REST (REpresentational State Transfer), which is the software architectural style of WWW.

It consist of a coordinated set of architectural constraints applied to components, connectors, and data elements, within a distributed hypermedia system.

Performance, Scalability, Simplicity, Modifiability, Visibility, Portability and Reliability



REST - Definition

REST is intended to evoke an image of how a well-designed Web application behaves: a network of web pages (a virtual state-machine), where the user progresses through an application by selecting links (state transitions), resulting in the next page (representing the next state of the application) being transferred to the user and rendered for their use.

Roy Fielding in his Ph.D. dissertation in 2000

REST style

REST can be described as Set of **formal** and **informal** guides to creating architectures — *constraints*

- Client-server
- Stateless
- Cacheable
- Uniform interface
- Layered system
- Code on demand (optional)

REST vs. SOAP - 1

<https://youtu.be/bPNfu0IZhoE>

**RESTful Web Services
vs.
SOAP Architecture**

SOAP Python Demo

- Python SOAP Country code

```
webservices.oorsprong.org/websamples.countryinfo/CountryInfoService?wsdl

//webservices.oorsprong.org/websamples.countryinfo/CountryInfoService?wsdl

://webservices.oorsprong.org/websamples.countryinfo/CountryInfoService?wsdl

element
lement(
xType(
d.Element(
http://www.w3.org/2005/08/addressing}Action", zeep.xsd.String()

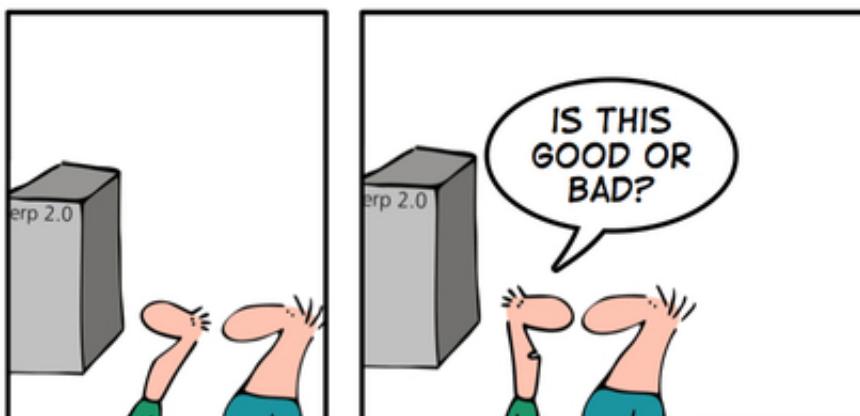
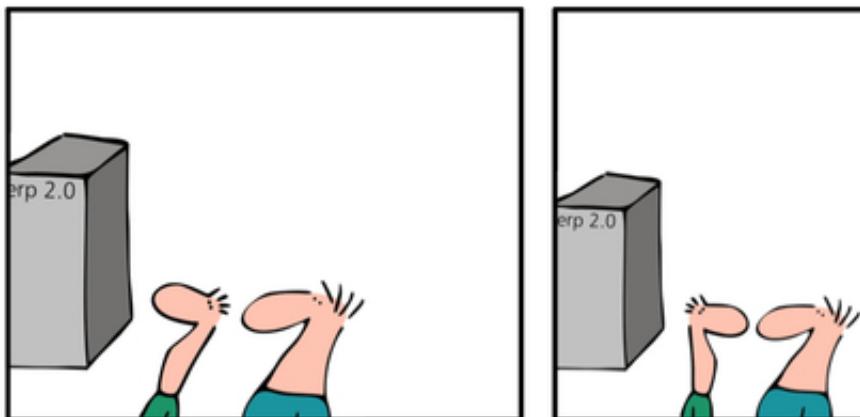
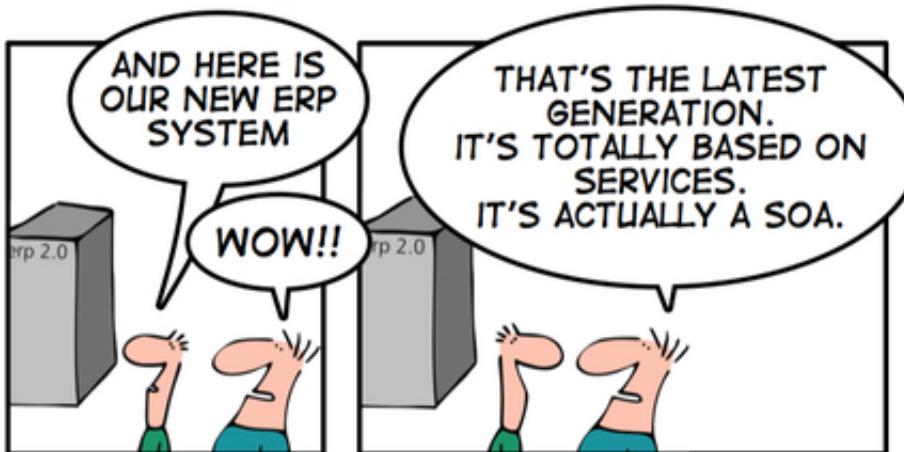
d.Element(
http://www.w3.org/2005/08/addressing}To", zeep.xsd.String()

lue from header element
er(Action=method_url, To=service_url)

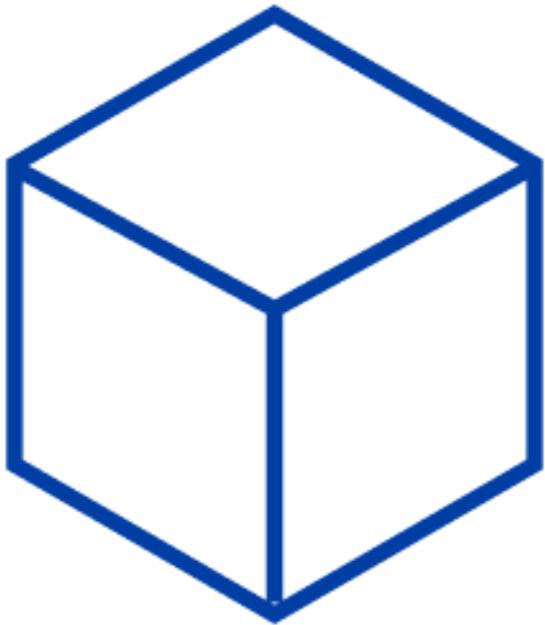
lient
t(wsdl=wsdl_url)

for Denmark
e country you want UK/US/SE
```

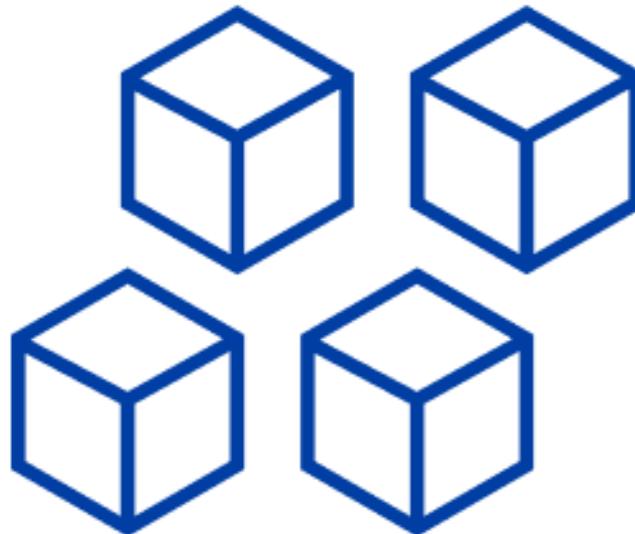
SOA



SOA FOR EVERYBODY



MONOLITHIC
Single unit



SOA
Coarse-grained

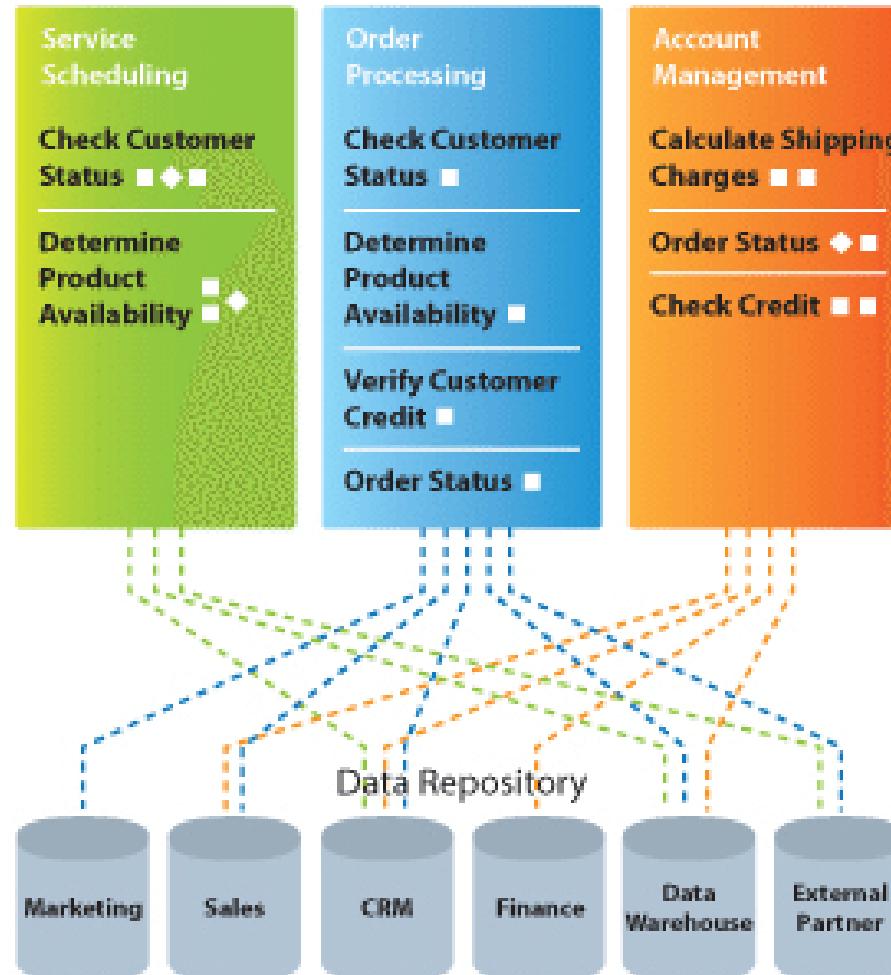


MICROSERVICES
Fine-grained

Before SOA

Closed - Monolithic - Brittle

Application Dependent Business Functions



SOA - Service Oriented Architecture

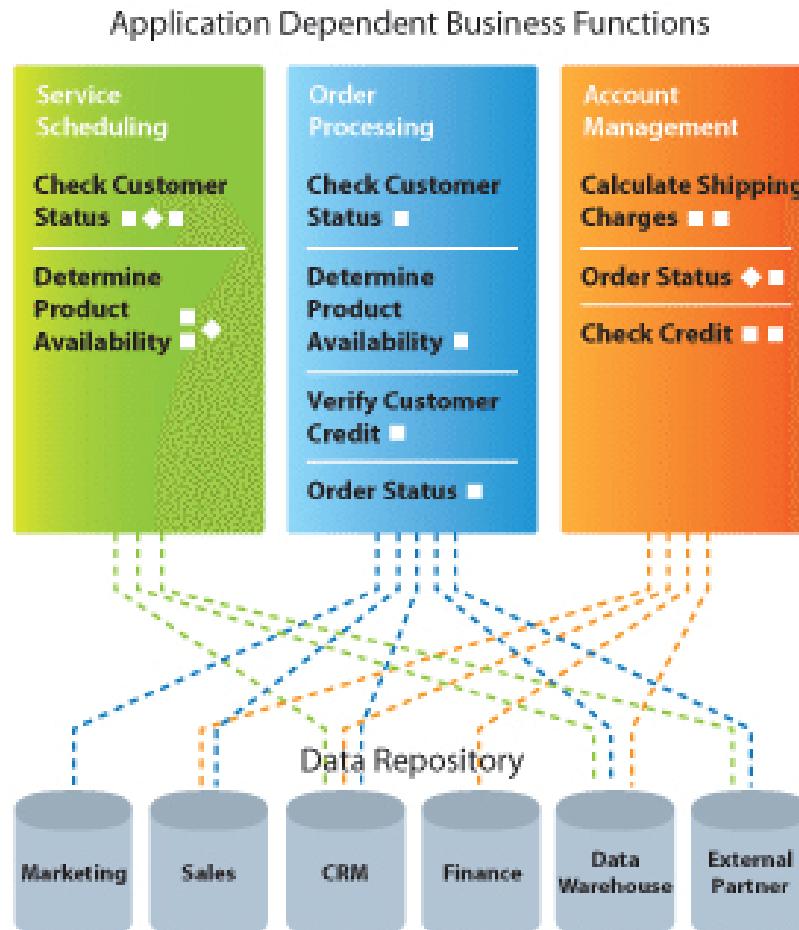
SOA can be described as an approach to the development process, which, based on the business, leads to the development, acquisition and use of IT solutions as a set of business support, **reusable** and **flexible services**.



- SOA organize contexts in a **vertical way**
- **Multiples components** can be part of the **same service** providing multiples capabilities (operations)
- An SOA service is like a bounded context
- SOA fosters **reuse** and composition inside the **same domain**
- Each SOA service represents a group of **smaller components**
- In SOA, it is common to see all **services** using the **same technology stack** and the **same database technology**

Before SOA

Closed - Monolithic - Brittle



After SOA

Shared services - Collaborative - Interoperable - Integrated

Composite Applications



Reusable Business Services



Data Repository



API

What is an API?

Find the best, in your opinion, **description** of an API

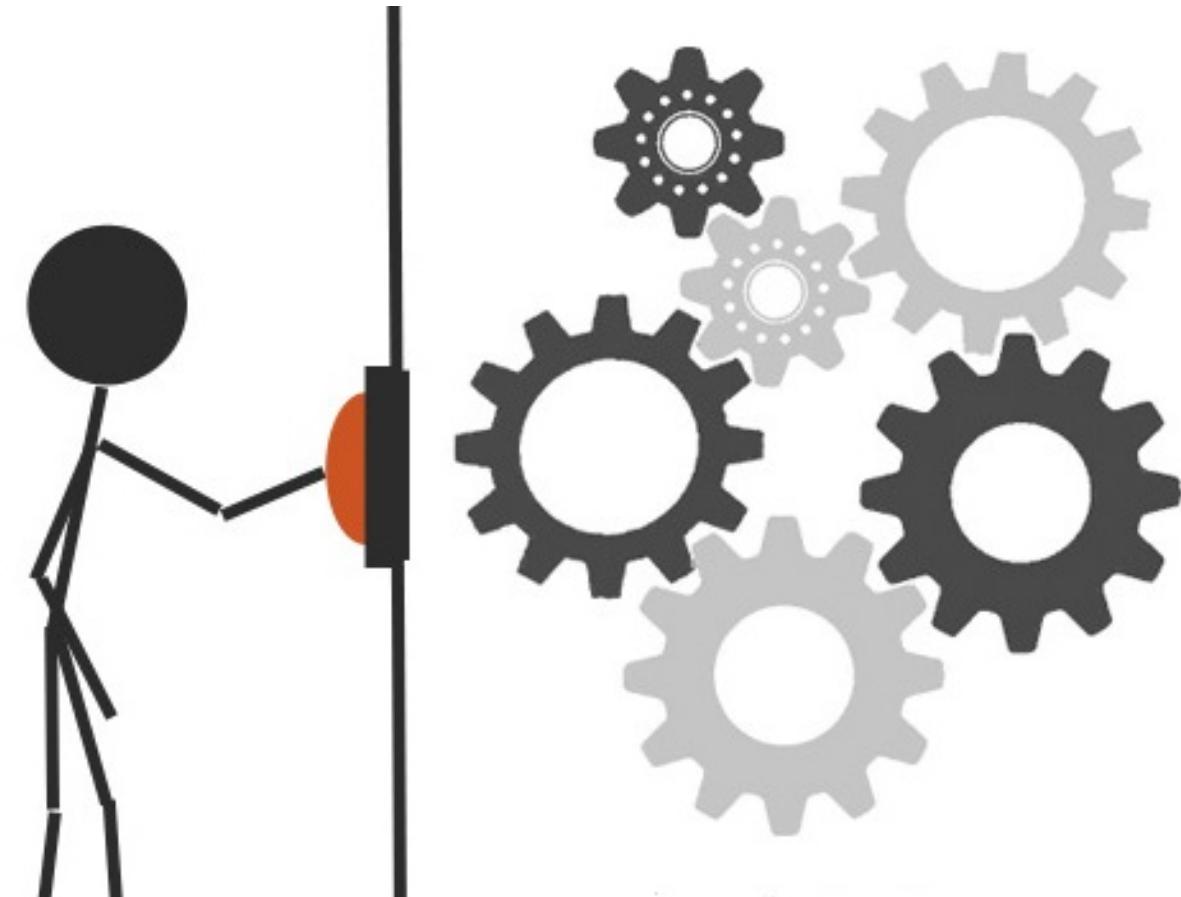
Come up with some **examples** of API's



What is an API?

API stands for Application
Programming Interface

But what is a *Interface*?



Interfaces

Every device you use has some kind of interface.

We use these interfaces to get the device to **do the thing we want**.

We **don't need to understand** the underlying functionality.



Abstraction

API's provide a layer of abstraction for the user.

Abstraction **hides everything but what is relevant** to the user, making it *simple* to use.

An API is how applications talk to each other

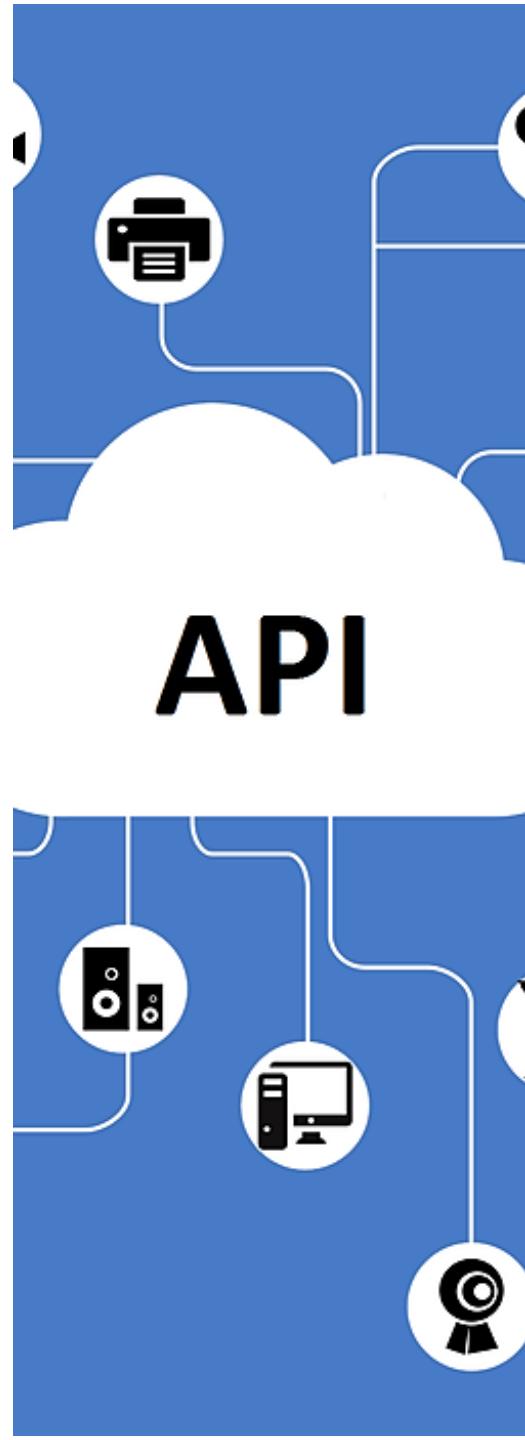


API - Application Programming Interface

API is a software intermediary that allows two applications to talk to each other.

You can ask an API for **data**, and they API will return what you want, usually in the form of **JSON** or **XML**. You can then use the data in your application.

Every time you use an app like Facebook, send an SMS, or check the weather on your phone, you're using an API.



API's as a way to serve your customers

Some companies are packaging API's as products.

- Weather Underground sells access to its weather data API
 - www.wunderground.com
- e-conomic has an API where the customers can access there data
 - www.e-conomic.com

When a company offers an API to their customers, it just means that they've built a set of dedicated URLs that return pure data responses — meaning the responses won't contain the kind of presentational overhead that you would expect in a graphical user interface like a website.

OpenAPI

There is the **OpenAPI Specification** (**OAS**), a technical specification that describes certain APIs, and there is the **OpenAPI Initiative** (**OAI**), an organization that enables specifications like OAS to thrive.

www.openapis.org



What is the difference between a Web service and an API?

An API is an **interface** that allows you to build on the data and functionality of another application, while a **web service** is a **network-based resource** that fulfills a **specific task**.

Yes, there's **overlap between the two**:

- All web services are API's
- Not all API's are web services
- Web services require a network. APIs can be on- or offline, web services must use a network
- Web services are usually associated with SOA
- API's are **protocol agnostic**. API's can use any protocols or design styles - Web services use SOAP, REST, UDDI, XML-RPC

FastAPI

FastAPI is a modern, **fast (*high-performance*)**, **web framework** for **building APIs** with Python 3.7+ based on standard Python type hints.



```
<iframe width="1100" height="1000"  
src="https://www.youtube.com/embed/0RS9W8MtZe4" title="YouTube video player"  
frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media;  
gyroscope; picture-in-picture; web-share" allowfullscreen></iframe>
```

Flask <> FastAPI

When to use Flask?

- To develop web applications
- To develop quick prototypes

When to use FastAPI?

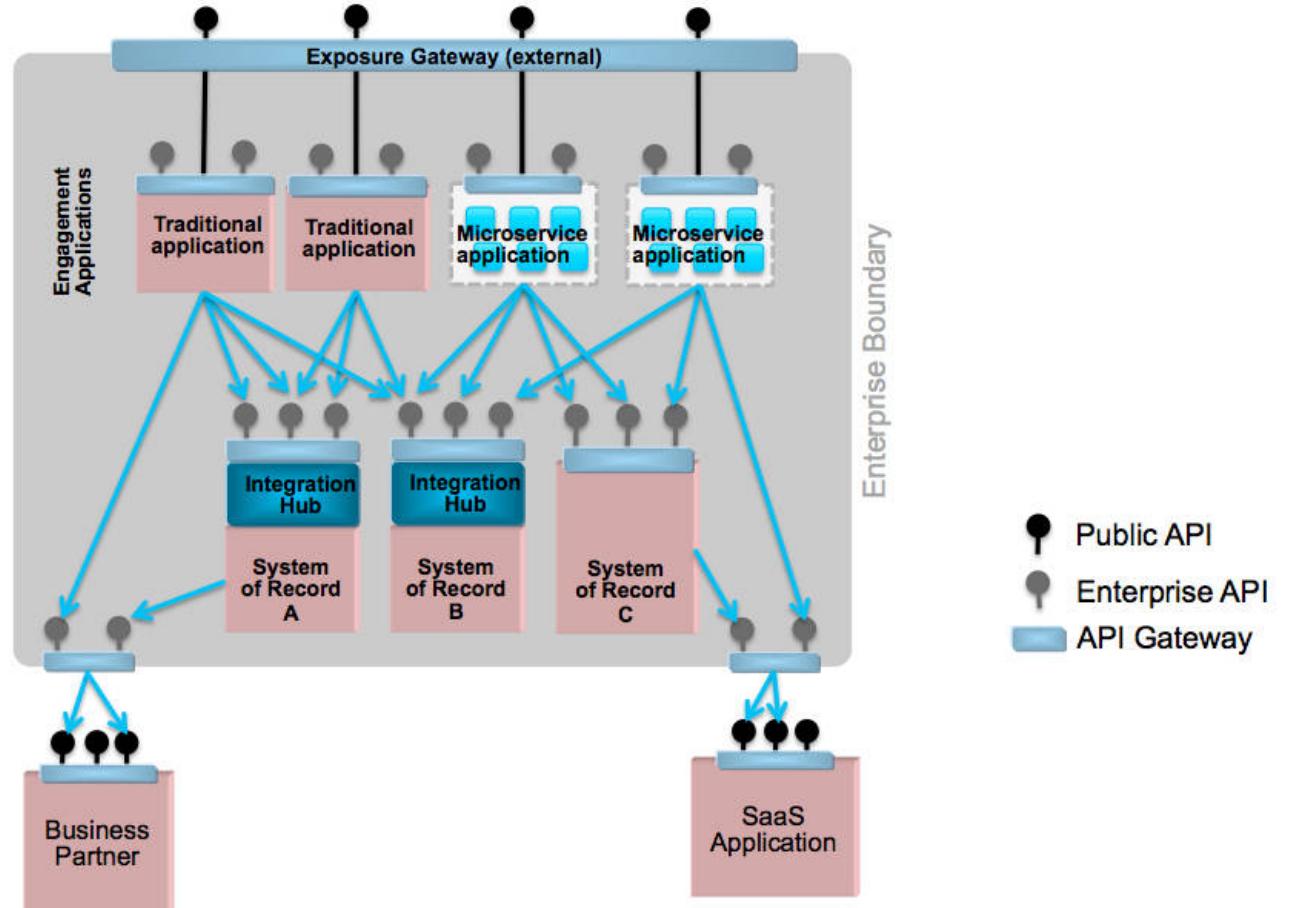
- To develop APIs from scratch
- To lower the number of bugs and errors in code



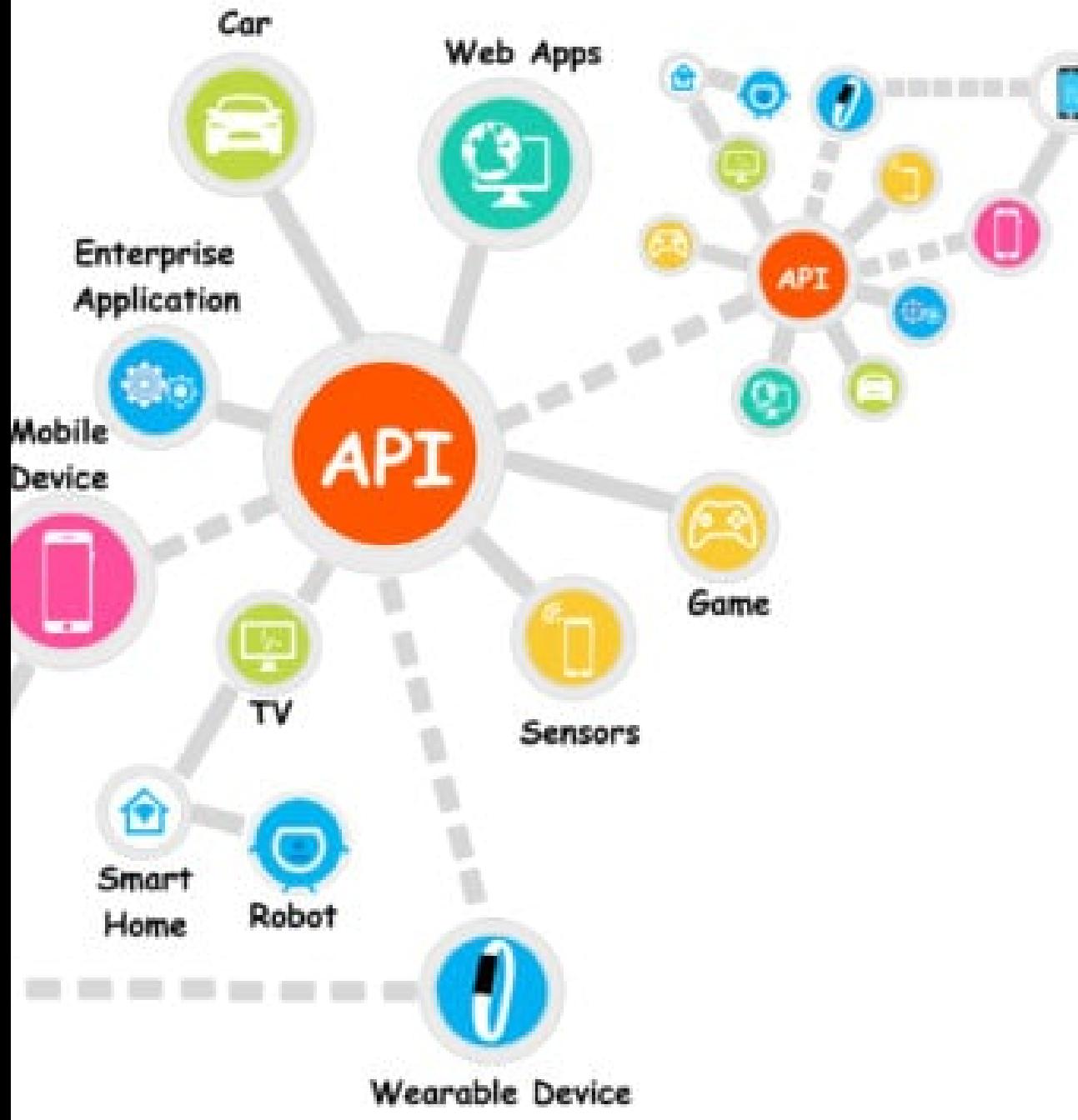
Differences between Flask and FastAPI

| | Flask | FastAPI |
|---------------------------|---|---|
| HTTP methods | <code>@app.route("/", methods = ["GET"])</code> <code>@app.route("/", methods = ["POST"])</code> | <code>@app.get("/")</code> <code>@app.post("/")</code> |
| Data validation | No data validation | In-built data validation feature |
| Error message display | Error messages displayed in HTML pages | Error messages displayed in JSON format |
| Asynchronous task support | No | Yes |
| Performance | Constrained in WSGI applications | Implements ASGI specifications |
| Documentation support | Manual | Automatic |
| Community support | Rich community support | Smaller community support |

Microservices, SOA, and API's combined



API DEMO



API - restcountries

- Python file - .py
- Jupyter Lab - .ipynb

<https://restcountries.eu>

python-restcountries <https://pypi.org/project/python-restcountries/>

```
[5]: # Install  
!pip install python-restcountries  
  
[6]: # From restcountries import RestCountryApi as rapi  
from restcountries import RestCountryApiV2 as rapi  
  
# Get Denmark info  
country_list = rapi.get_countries_by_name('Denmark')  
  
[7]: # Print information  
country = country_list[0]  
print(country.name)  
print(country.capital)  
print(country.calling_codes)  
print(country.population)  
print(country.flag)  
print(country.languages)
```

```
Denmark  
Copenhagen  
['45']  
5717014  
https://restcountries.eu/data/dnk.svg  
[{'iso639_1': 'da', 'iso639_2': 'dan', 'name': 'Danish', 'nativeName': 'dansk'}]
```

Spotify API

Spotify provides software and app developers access to some of their data about users, playlists, and artists through a Web API.

- [Spotify_API_Spotipy.pdf](#)
- [Jupyter Lab Code .ipynb](#)
- [Python Code .py](#)



Newscatcher

- [Demo GitHub Repository](#)
- <https://newscatcherapi.com>

News Data: structured, relevant, real-time

Search multi-language worldwide news articles published online with NewsCatcher's News API

[Live Demo](#)

```
  "title": "Tesla M
  "author": "Rob Cl
  "published_date":
  "published_date_p
  "link": "https://
  "clean_url": "tec
  "excerpt": "What
  "summary": "Tesla
  "rights": "techra
  "rank": 640,
  "topic": "tech",
  "country": "US",
  "language": "en",
  "authors": [
    "Rob Clymo"
  ],
  "media": "https://
  "is_opinion": fal
  "twitter_account"
}
```

IBM - SOA

SOA for Dummies

Compliments of
IBM

Service Oriented Architecture FOR **DUMMIES**

2nd IBM Limited Edition

**A Reference
for the
Rest of Us!**

FREE eTips at dummies.com®

Judith Hurwitz
Robin Bloor
Marcia Kaufman
Dr. Fern Halper

Discover how
companies in seven
different industries
implemented SOA



Links

- <https://martinfowler.com/microservices>
- <https://www.ibm.com/cloud/learn/soa>
- <https://morioh.com/p/422b616d71a2>
- <https://fastapi.tiangolo.com/>