



**Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря Сікорського»**

# Інформаційні системи

Викладач: к.т.н., доц. Саяпіна Інна Олександрівна

# План заняття:

- ▶ API
  - ▶ Види
  - ▶ Технології реалізації:
    - ▶ REST API
    - ▶ GraphQL
    - ▶ gRPC

# Application Programming Interface

- ▶ це набір чітко визначених методів для взаємодії різних компонентів

<https://en.wikipedia.org/wiki/API>

# Web API

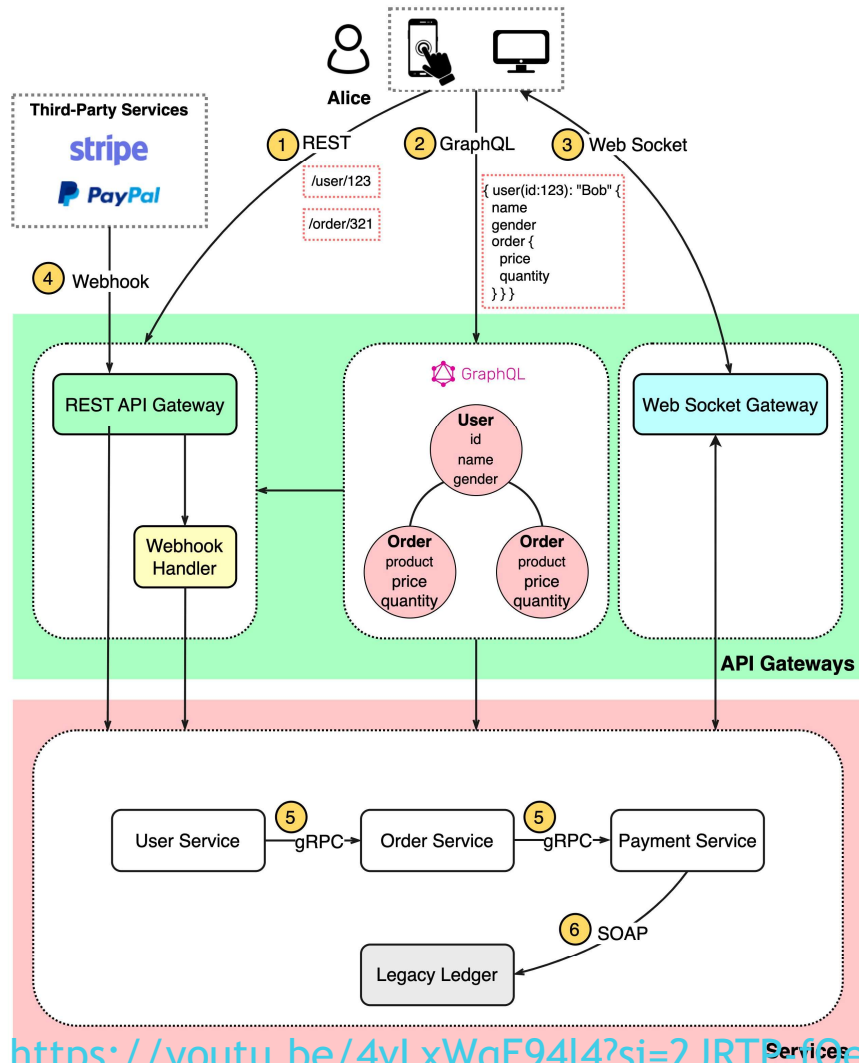
Найкраще підходить для систем, які потребують:

- ▶ Отримання та зберігання даних
- ▶ Дії, ініційовані користувачем
- ▶ Короткі, цілеспрямовані дії
- ▶ На основі запиту-відповіді



## API Architectural Styles

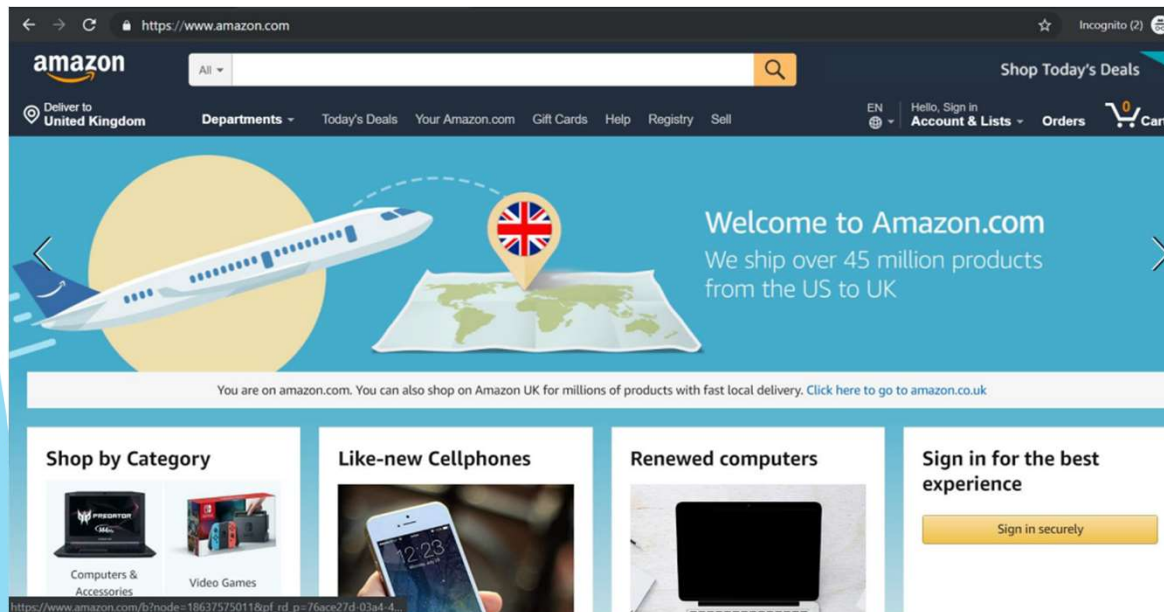
blog.bytebytego.com



<https://youtu.be/4vLxWqE94l4?si=2JRTFv0eoW9Cac4>

# User Interface vs Application Interface

## ► User Interface



## ► Application interface

GET /account HTTP/1.1

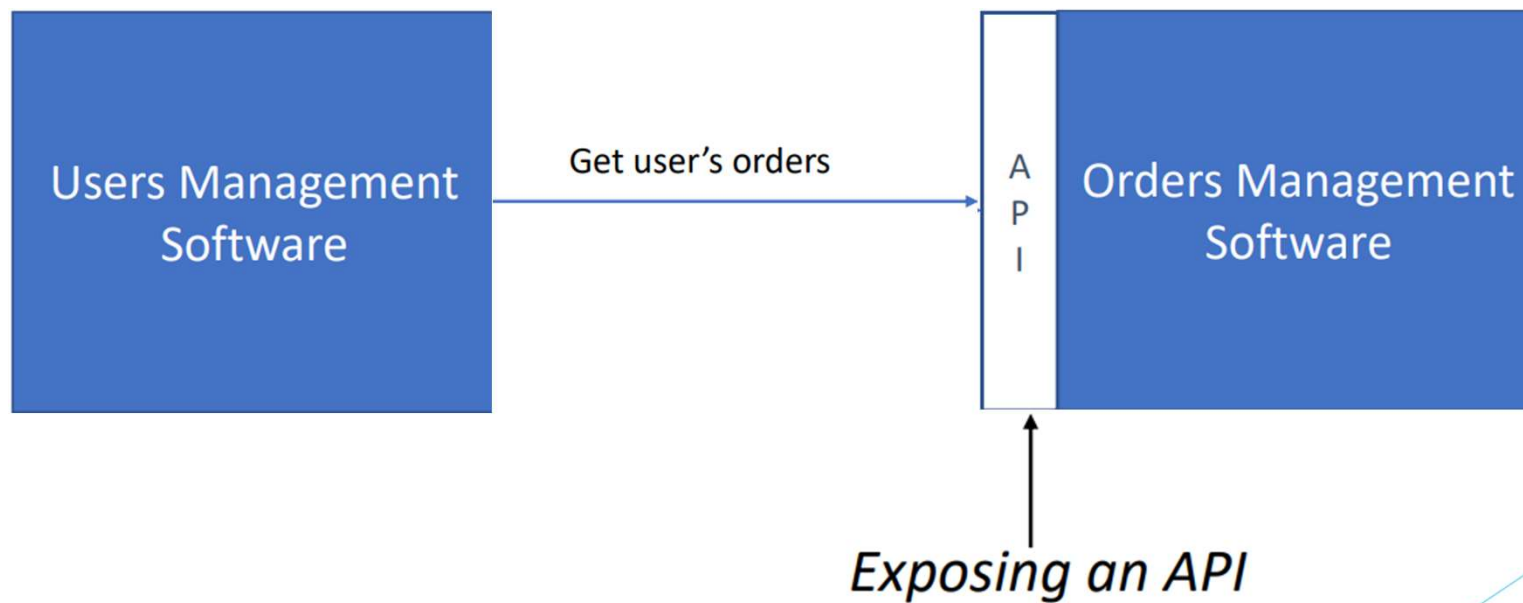
Content-Type: application/json

Host: apigateway.us-east-1.amazonaws.com

X-Amz-Date: 20160531T184618Z

Authorization: AWS4-HMAC-SHA256 Credential={access\_key\_ID}/us-east-1/apigateway/aws4\_request, SignedHeaders=content-type;host;x-amz-date, Signature={sig4\_hash}

# Принцип роботи



# Типи API

- ▶ Operating System
- ▶ Library
- ▶ Remote
- ▶ Web

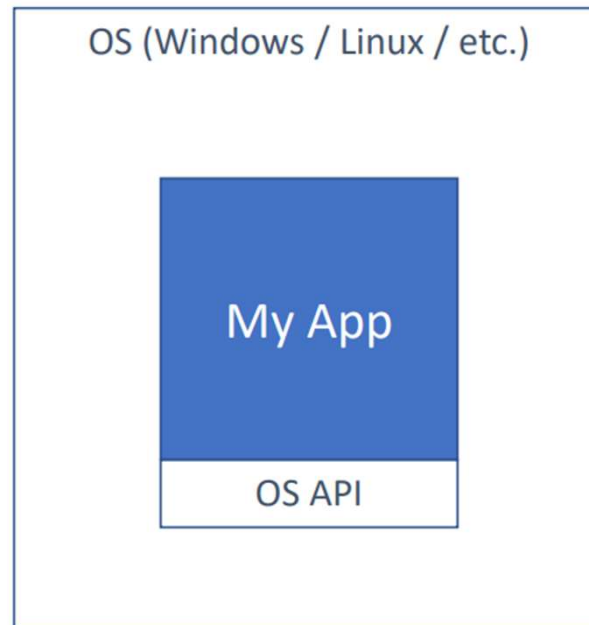




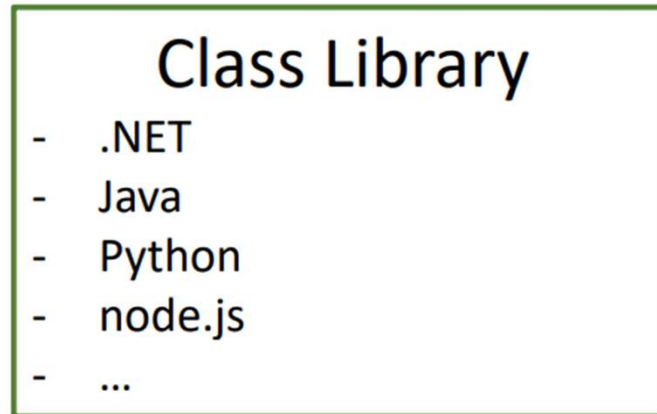
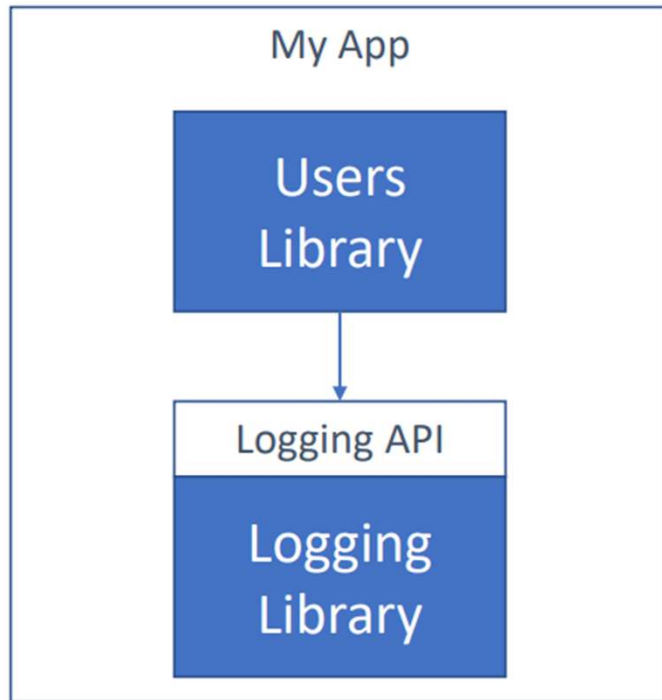
# Operating System API

- ▶ File System
- ▶ Network Devices
- ▶ User Interface Elements

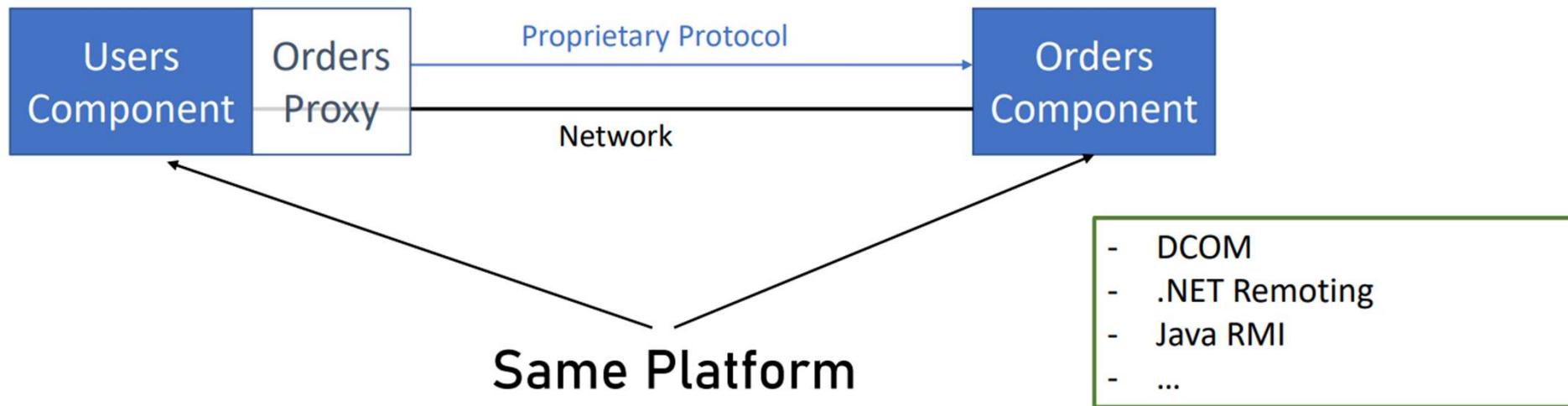
Win32 API



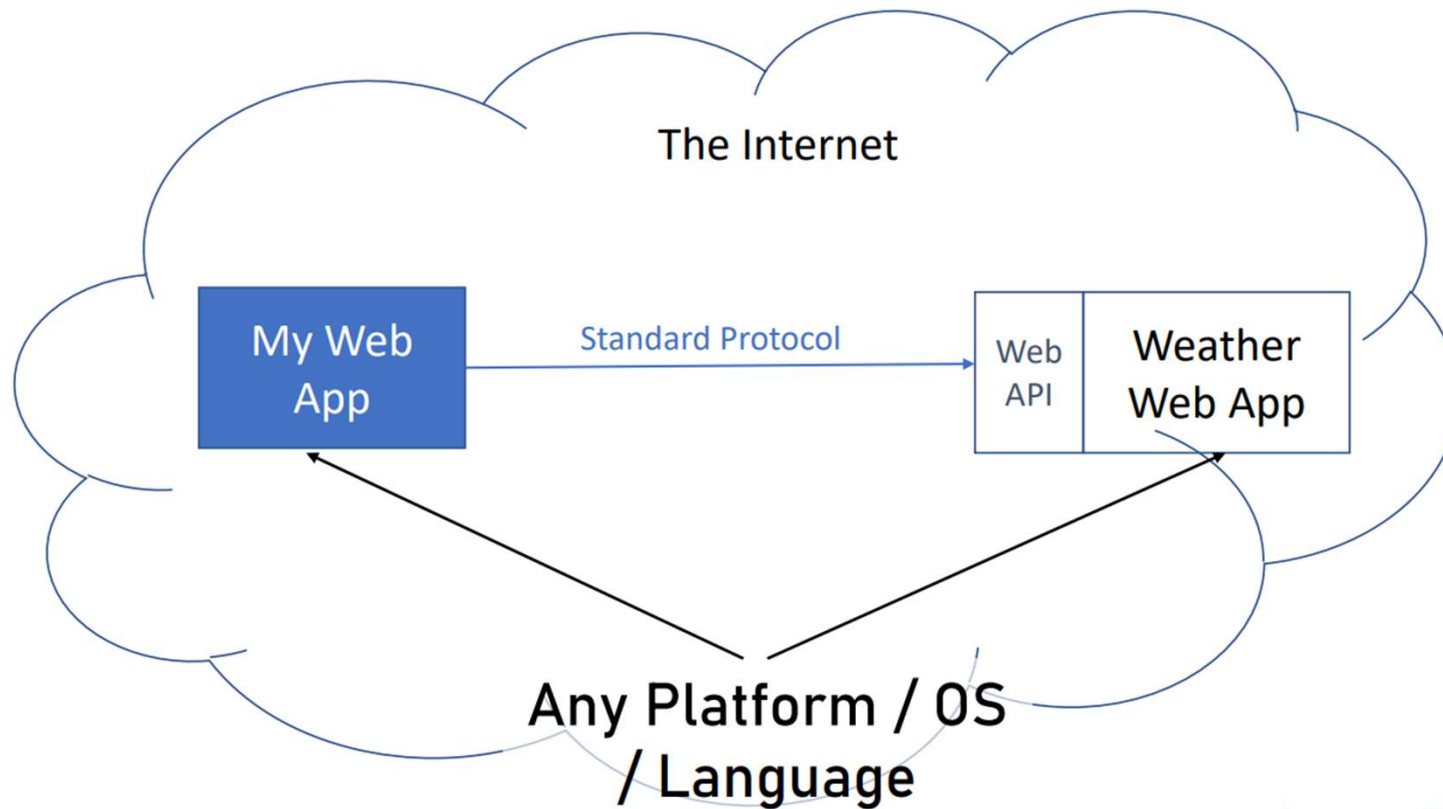
# Library API



# Remote API



# Web API



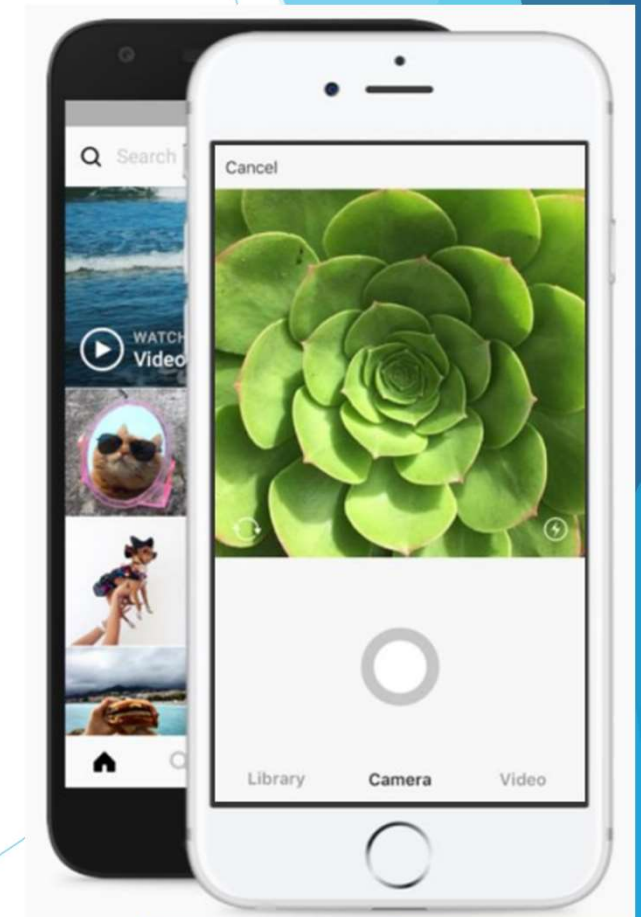
# Важливість API

- ▶ API використовуватиметься:
  - Вашим інтерфейсом користувача
  - Для розширення охоплення вашого додатка
  - Для можливості монетизації



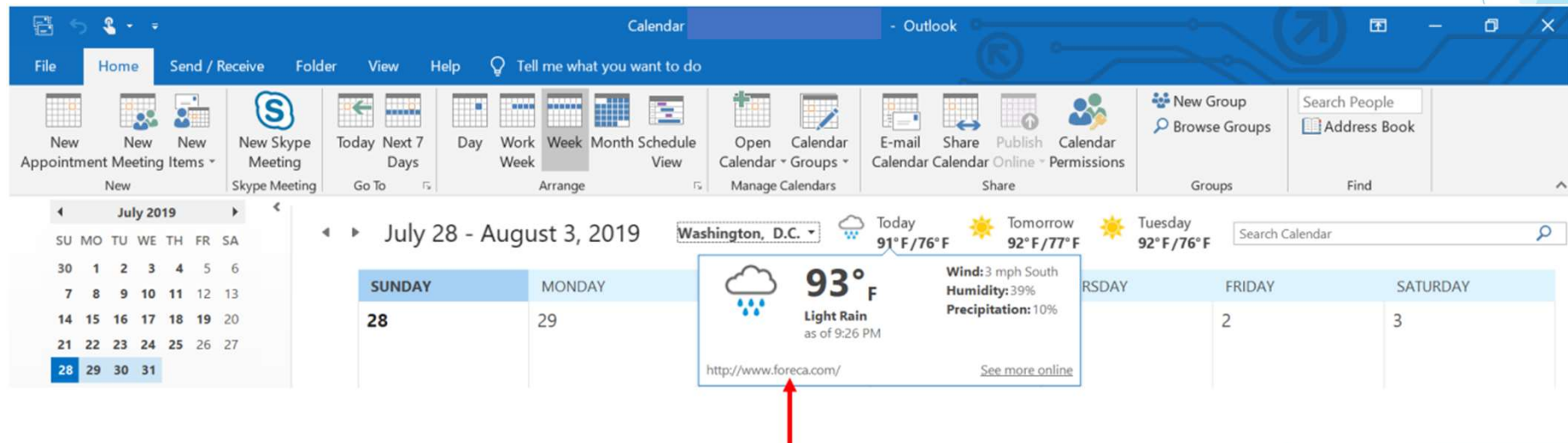
# API для інтерфейсу користувача

- ▶ Сучасні програми створюються у веб-версії та для мобільних пристроїв
- ▶ Мобільні додатки **ПОВИННІ** мати API серверу, щоб з'єднатися з ним
- ▶ Веб-клієнти, створені за допомогою SPA, також **ПОВИННІ** мати API сервер




# Розширення охоплення вашого додатка

- Розкриваючи API, інші програми можуть використовувати ваші дані



# Монетизація

## ► Плата за доступ до ваших даних

 **Developer Platform**

[Products ▾](#) [Docs ▾](#) [Use Cases ▾](#) [Community ▾](#)

[Support ▾](#) [Developer Portal](#) [Sign in](#)

## Find the right access for you

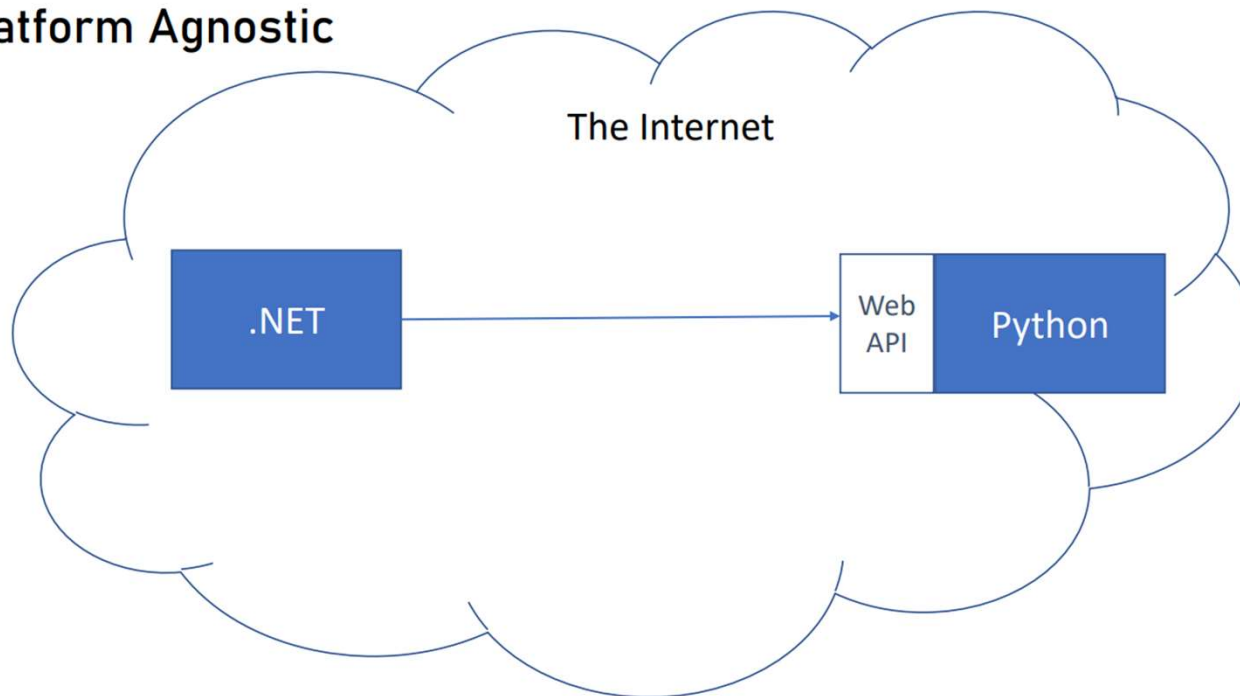
Free	Basic	Pro	Enterprise
For write-only use cases and testing the X API	For hobbyists or prototypes	For startups scaling their business	For businesses and scaled commercial projects
<ul style="list-style-type: none"><li>• Rate limited access to v2 posting and media upload endpoints</li><li>• 1,500 posts per month - posting limit at the app level</li><li>• 1 app ID</li><li>• Login with X</li><li>• Free</li></ul>	<ul style="list-style-type: none"><li>• Rate limited access to suite of v2 endpoints</li><li>• 3,000 posts per month - posting limit at the user level</li><li>• 50,000 posts per month - posting limit at the app level</li><li>• 10,000 posts per month - read-limit rate cap</li><li>• 2 app IDs</li><li>• Login with X</li><li>• \$100 per month</li></ul>	<ul style="list-style-type: none"><li>• Rate-limited access to suite of v2 endpoints, including search and filtered stream</li><li>• 1,000,000 posts per month - GET at the app level</li><li>• 300,000 posts per month - posting limit at the app level</li><li>• 3 app IDs</li><li>• Login with X</li><li>• \$5,000 per month</li></ul>	<ul style="list-style-type: none"><li>• Commercial-level access that meets your and your customer's specific needs</li><li>• Managed services by a dedicated account team</li><li>• Complete streams: replay, engagement metrics, backfill, and more features</li><li>• Monthly subscription tiers</li></ul>
<a href="#">Get started</a>	<a href="#">Subscribe now</a>	<a href="#">Subscribe now</a>	<a href="#">Apply now</a>





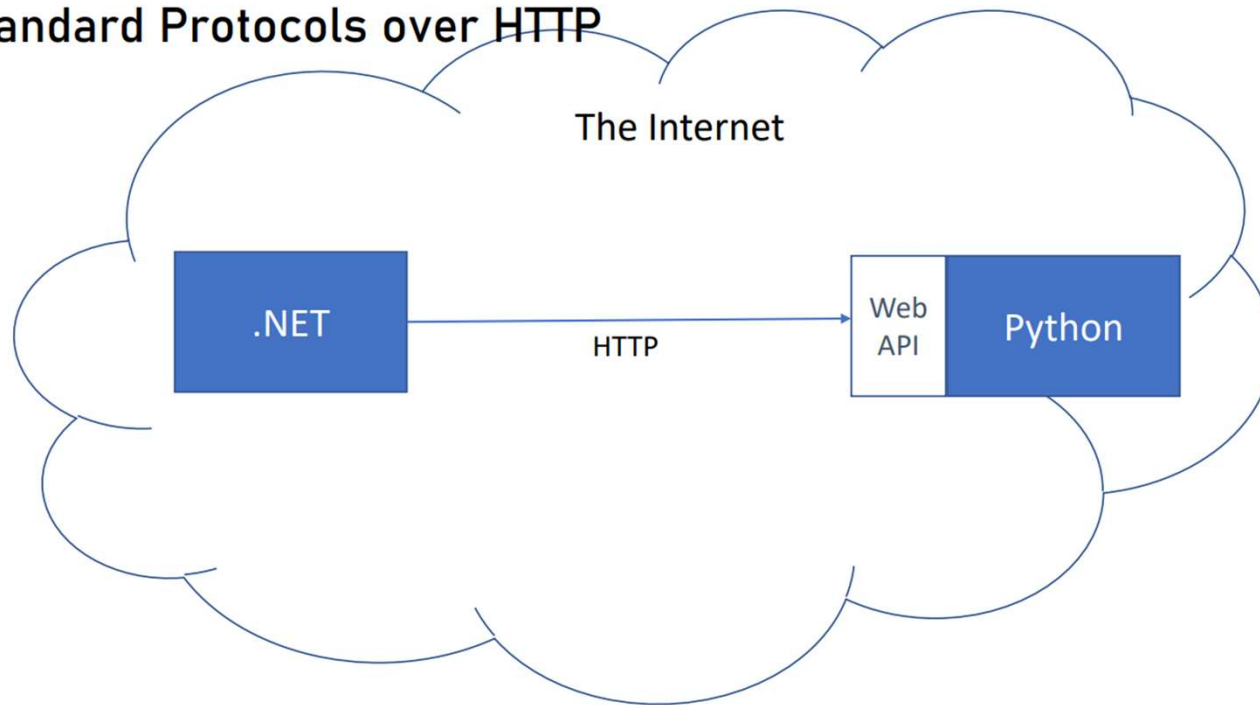
# Основні характеристики Web API

- Platform Agnostic



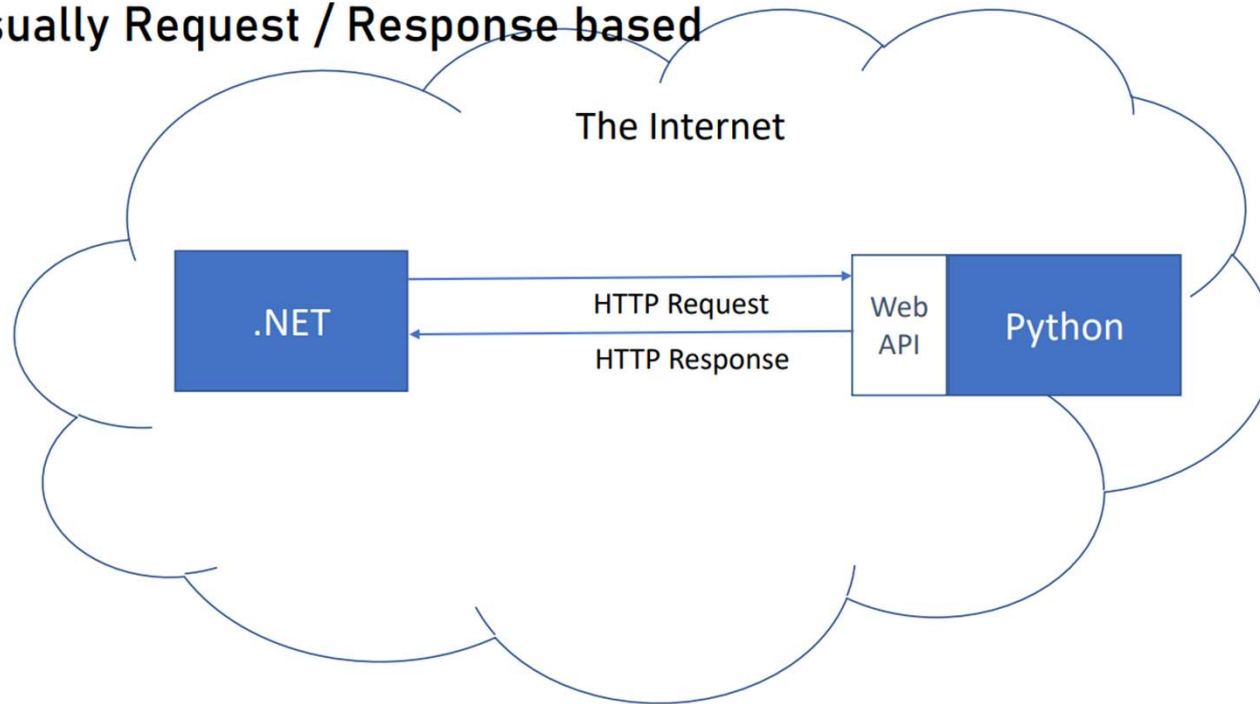
# Основні характеристики Web API

- **Standard Protocols over HTTP**



# Основні характеристики Web API

- Usually Request / Response based



# Типи Web API

- ▶ SOAP
- ▶ REST
- ▶ GraphQL
- ▶ gRPC

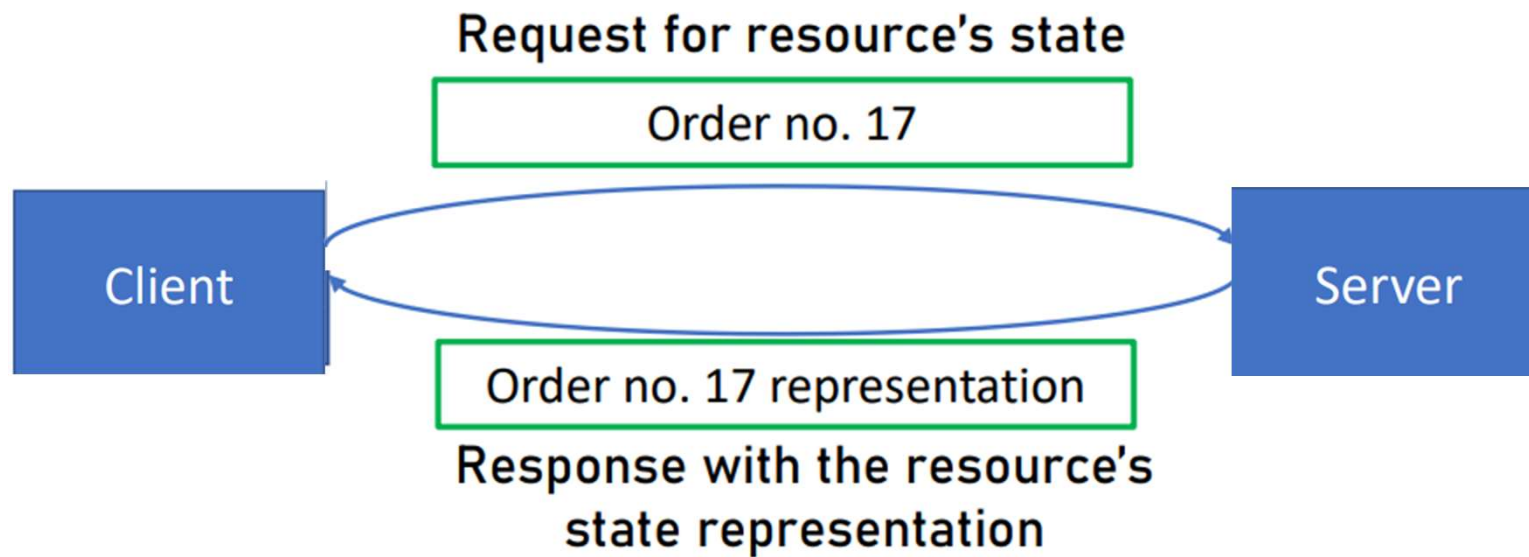


# SOAP

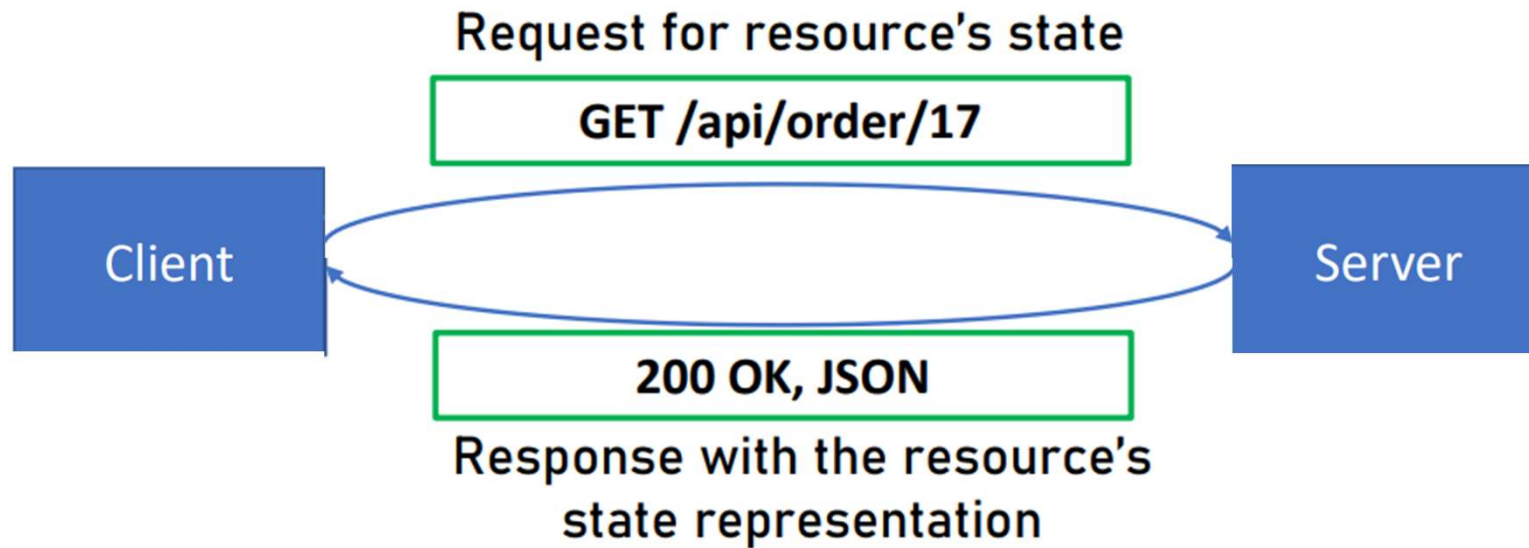
- ▶ Simple Object Access Protocol
- ▶ Designed in 1998 for Microsoft
- ▶ XML-Based
- ▶ RPC Style
- ▶ Extensible
- ▶ Outdated
- ▶ Do not use, unless have to

```
<?xml version="1.0"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body xmlns:m="http://www.stock.org/stock">
    <m:GetStockPrice>
      <m:StockName>IBM</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```

# REST: REpresentational State Transfer



# REST: Request / Response over HTTP



# Структура REST запиту

Method	→ HTTP Verb (GET, POST, PUT, DELETE...)
URL	→ Location of the resource + parameters
Headers	→ Meta-data of the request (User Agent...)
Body	→ Contents of the request (optional)

Method URL

```
GET /api/v1/entities/17
User-Agent: PostmanRuntime/7.15.2
Accept: */*
Cache-Control: no-cache
Postman-Token: f7f64f44-2580-4707-8f7c-d3e657bf21ee
Host: memitest.free.beeceptor.com
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Headers



# Глаголи HTTP

- ▶ Отримати ресурс - GET
- ▶ Додайте ресурс - POST
- ▶ Змінити ресурс - PUT
- ▶ Видалити ресурс - DELETE

# GET

- ▶ Використовується для отримання ресурсів
- ▶ НІКОЛИ не використовуйте його для додавання/оновлення/видалення ресурсів
- ▶ Дієслово за замовчуванням для адресного рядка браузера
- ▶ Зазвичай поєднується з параметрами
- ▶ Не повинно містити тіло повідомлення

# POST

- ▶ Використовується для додавання ресурсу
- ▶ Має містити тіло повідомлення, яке вказує на ресурс
- ▶ буде додано
- ▶ Не має містити параметрів запити
- ▶ • тобто POST /api/entity?~~company=15~~

# PUT

- ▶ Використовується для зміни ресурсів
- ▶ Має містити тіло повідомлення, яке вказує на ресурс, який має бути змінений
- ▶ Не має містити параметрів рядка запиту
- ▶ • тобто PUT /api/entity?~~company~~=15

# DELETE

- ▶ Використовується для видалення ресурсів
- ▶ НІКОЛИ не використовуйте його для додавання / оновлення / отримання ресурсів
- ▶ Майже завжди поєднується з параметрами
- ▶ Не має містити тіла повідомлення

# HTTP Verbs summary

Verb	Role	Body?	Params In...
<b>GET</b>	Retrieve resource(s)	No	URL
<b>POST</b>	Add resource(s)	Yes	Body
<b>PUT</b>	Modify resource(s)	Yes	Body
<b>DELETE</b>	Delete resource(s)	No	URL

# Структура URL

- ▶ Має бути:
- ▶ Зрозуміла сама за себе
- ▶ Послідовна у всьому API
- ▶ Передбачувана

F12 (Network)

<https://api.agify.io/?name=>

# Правила створення URL

- ▶ Без дієслів
- ▶ Роль дієслова виконує глагол HTTP
- ▶ Включає версію API в URL
- ▶ Використовує параметр ідентифікатора, щоб вказати певну сутність, з якою потрібно працювати
- ▶ Не має довгих фраз

*API Word*      *Entity*      *Sub Entity*

*Version*      *ID Param*      *Query Parameters*

/api/v1/order/17/items?user=john&date=12/12/2018

A diagram illustrating the structure of the URL "/api/v1/order/17/items?user=john&date=12/12/2018". The URL is broken down into segments with labels above and below them. Above the URL, three green brackets group "api", "v1", and "order" under the label "API Word", "order" and "17" under "Entity", and "items" under "Sub Entity". Below the URL, three green brackets group "v1" under "Version", "17" under "ID Param", and the entire query string "?user=john&date=12/12/2018" under "Query Parameters".



# Коди відповідей

- ▶ 1xx - Informational Response
- ▶ 2xx - Success
- ▶ 3xx - Redirection
- ▶ 4xx - Client Errors
- ▶ 5xx - Server Errors

## Найрозпосюдженіші коди відповідей

200 OK

400 Bad Request

500 Internal  
Server Error

201 Created

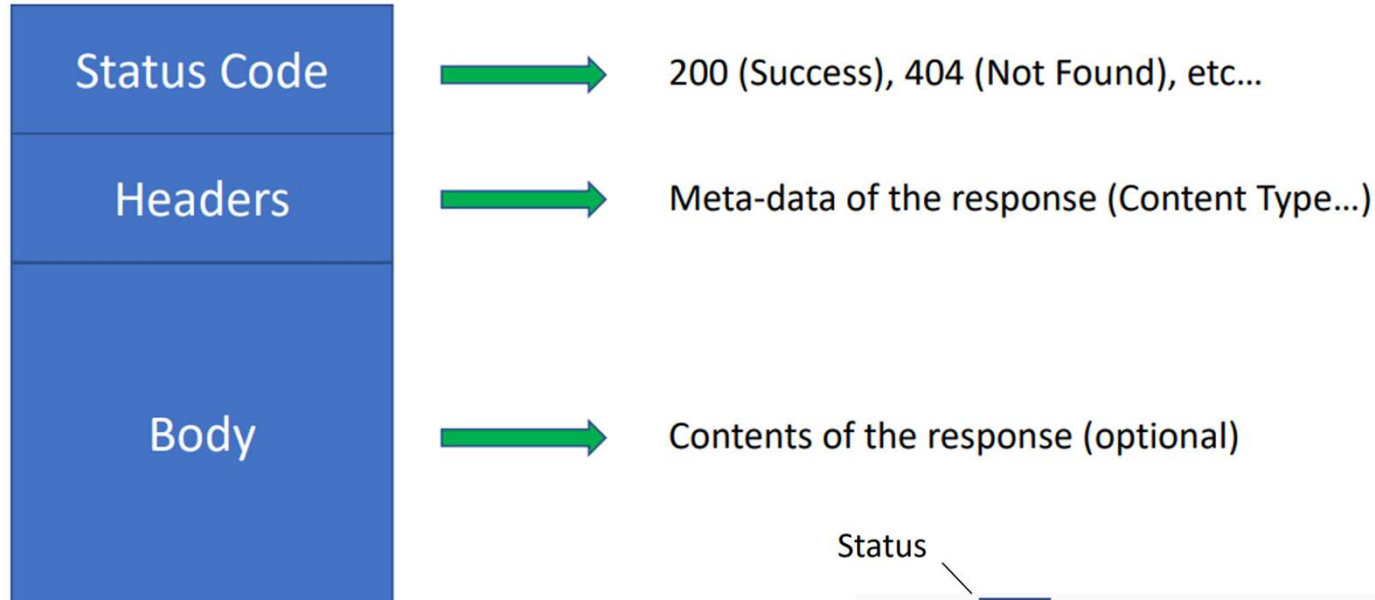
401 Unauthorized

202 Accepted

403 Forbidden

404 Not Found

# Структура REST відповіді



- ▶ Тип відповіді:
- Зазвичай JSON
- Рідко може бути XML або HTML

An example of an HTTP response is shown, with labels pointing to its components:

- Status**: Points to the status line "HTTP/1.1 200".
- Headers**: Points to the header section containing:

```
status: 200
Date: Fri, 02 Aug 2019 09:06:13 GMT
Content-Type: text/plain
Transfer-Encoding: chunked
Connection: keep-alive
access-control-allow-origin: *
Vary: Accept-Encoding
```
- Body**: Points to the JSON body:

```
{ "orderId" : "17", "orderDate" : "12.12.2018", "items" : [ "id" : "6", "id" : "56" ] }
```

# REST Висновки:

- ▶ Працює з сутностями
- ▶ На основі протоколу HTTP
- ▶ (дієслова, коди відповідей, URL)
- ▶ Парадигма Request/Response
- ▶ Надзвичайно легко реалізувати



# Недоліки REST

- ▶ Фіксована структура сутності
  - ▶ За допомогою REST клієнт не може вказати частину параметрів сутності, яку він хоче отримати
  - ▶ Формат повернених даних про сутність встановлюється розробником серверної частини та не може бути змінений на основі кожного запиту
  - ▶ Збільшує час передачі даних та затримку мережі
  - ▶ Підтримує лише шаблон запиту-відповіді (Push-повідомлення стають надзвичайно корисними)



# Історія GraphQL

- ▶ Facebook розробив GraphQL як заміну REST API для внутрішнього використання
- ▶ Вперше використано у 2012 році
- ▶ Специфікація стала відкритою у 2015 році
- ▶ Зараз керує GraphQL Foundation



# Graph QL - це

1. Специфікація
2. Визначає структуру даних відповіді
3. На основі JSON
4. Базується на схемі
5. 3 види операцій
6. Кросплатформенність

```
type Character {  
  name: String!  
  appearsIn: [Episode!]!  
}
```

# Graph QL Висновки

- Працює з сутностями
- Є мовою запитів
- Запит / Відповідь + підписка
- Не так легко реалізувати





# Проблеми REST, що призвели до створення gRPC

## 1. Продуктивність

- REST використовує текстові повідомлення та протоколи
- HTTP 1.1 — це текстовий протокол
- JSON — це текстовий формат обміну повідомленнями

## 2. Лише шаблон запит-відповідь

- Складність реалізації push-повідомлень

## 3. Обмежений синтаксис

- Не підходить для запуску дій
  - Приклади:
  - Почати новий процес
  - Виконати вхід
  - Вимкнути пристрій

# Історія gRPC

- ▶ Google розробив новий стандарт веб-API для внутрішнього використання
  - Називався Stubby
- ▶ • У 2015 році Google вирішив створити наступну версію Stubby та зробити його відкритим кодом
- ▶ • Він отримав назву gRPC

# Основні принципи gRPC

- ▶ • Розповсюджує повідомлення та сервіси, а необ'єкти та посилання
- ▶ • Доступний на всіх популярних платформах
- ▶ • Безкоштовний та відкритий
- ▶ • Продуктивний
- ▶ • Дозволяє потокове передавання

# Основні принципи gRPC

- ▶ Web API
- ▶ На основі HTTP/2
- ▶ Стиль RPC
- ▶ Використовує Protobuf як корисне навантаження
- ▶ Декілька стилів спілкування

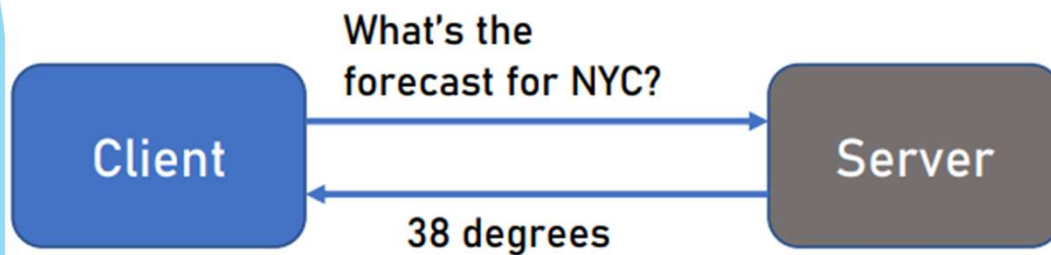


# Основні принципи gRPC

## Based on HTTP/2

### HTTP/1.1

- Connection based on the Request/Response model



### HTTP/2

- Allows streaming from both sides
- In addition to Request/Response
- Opens new possibilities such as push notifications and more

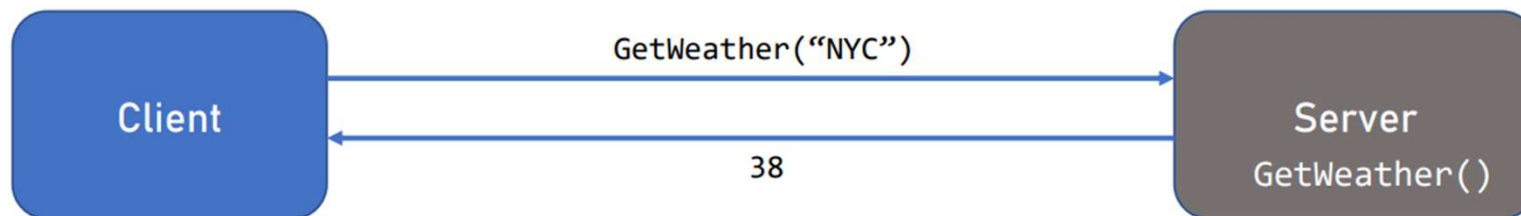


# Основні принципи gRPC

- ▶ Через складну обробку HTTP/2 у gRPC не можна використовувати з браузерів
- ▶ Використовується в основному з програм для мобільних пристроїв і для серверної частини

# Основні принципи gRPC

- ▶ Remote Procedure Call
- ▶ Виклик клієнтом метода на сервері



# Основні принципи gRPC

## Стили спілкування:

- ▶ Unary
  - ▶ Request - response model
- ▶ Client Streaming
  - ▶ Клієнт відкриває підключення до сервера
  - ▶ Безперервно надсилає повідомлення на сервер
  - ▶ Чудово підходить для телеметрії, чатів тощо
- ▶ Server Streaming
  - ▶ Клієнт відкриває підключення до сервера
  - ▶ Сервер надсилає безперервні повідомлення за допомогою з'єднання
  - ▶ Чудово підходить для сповіщень, чату тощо.
- ▶ Bi-directional
  - ▶ Клієнт відкриває підключення до сервера
  - ▶ І клієнт, і сервер надсилають безперервні повідомлення за допомогою з'єднання
  - ▶ Чудово підходить для телеметрії, чату, даних у реальному часі тощо.



# Protobuf



- ▶ Формат даних, який використовує gRPC
- ▶ Теоретично gRPC може використовувати інші формати, але цього не відбувається
- ▶ Двійковий формат
- ▶ Оголошує формат повідомлення у файлі .proto
- ▶ Генерує клієнтський код підтримуваними мовами

# gRPC Висновки

- ▶ Працює з діями
- ▶ Не працює з браузерами
- ▶ Запит/Відповідь + потокове передавання
- ▶ Не легко реалізувати



## Висновки

	REST	 GraphQL	 gRPC
Simplicity	✓	✗	✗
Flexibility	✗*	✓	✗
Performance	✓✗	✗	✓
Push notifications	✗	✓	✓
Native browser support	✓	✓	✗
Semantics	Entities	Entities	Actions

# Корисні пізнавальні відео про технології:

- ▶ REST [https://youtu.be/-mN3VyJuCjM?si=3\\_P\\_yG\\_qjDj5hrNK](https://youtu.be/-mN3VyJuCjM?si=3_P_yG_qjDj5hrNK)
- ▶ GraphQL [https://youtu.be/yWzKJPw\\_VzM?si=-b0RVDNkilz0oBC9](https://youtu.be/yWzKJPw_VzM?si=-b0RVDNkilz0oBC9)
- ▶ gRPC <https://youtu.be/gnchfOojMk4?si=DO8ZBRVBYAquSyhG>