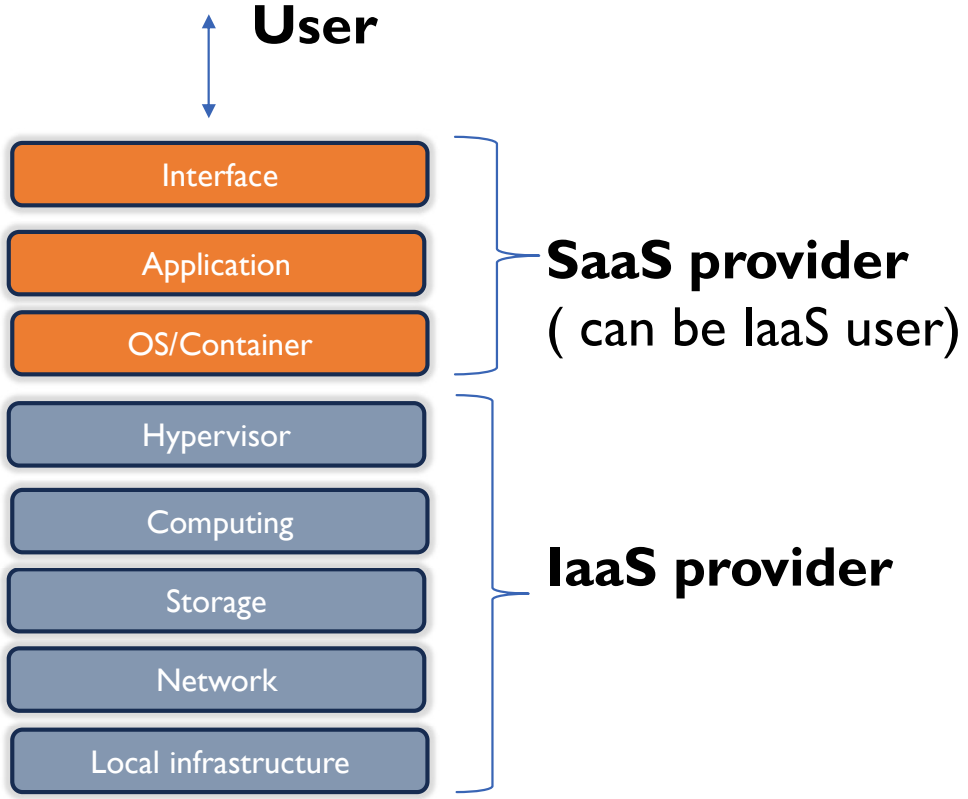
PAAS AND SAAS

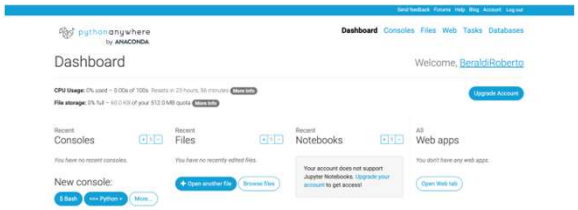
SAAS

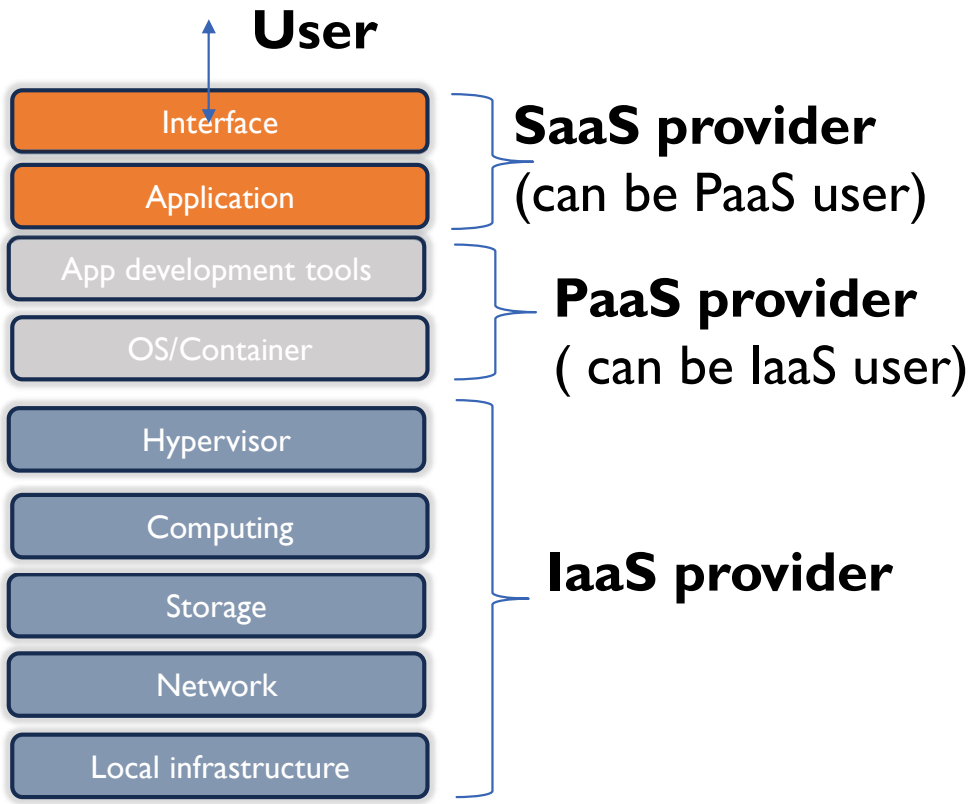
- An IaaS user can develop software that is delivered to other users.
- **SaaS** (Software as a Service) refers to software applications that are delivered as a service, that is used without managing their underlying infrastructure or platform.
- Users can be either humans or other software systems.
- In a broad sense, most applications accessed over the Internet can be considered **SaaS**



PAAS

- To facilitate application software development, the responsibility of managing the operating system and development environment is delegated to a provider, allowing developers to focus solely on writing code.
- This service is called **PaaS** (Platform as a Service).
  - For example, Jupyter



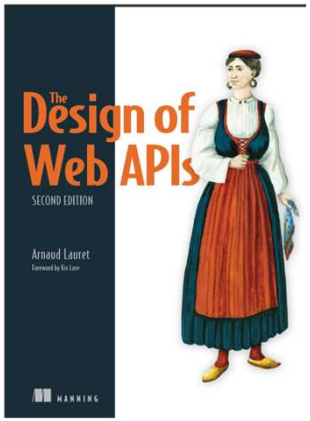


A JOURNEY AS SAAS PROVIDER

- Let's now focus on how to deliver a software as a service in case the users are other programs
- As the software is included in a larger application, this software is accessed through an API (Application Program Interface), and since they are accessed exploiting the web technologies (HTTP is the de-facto standard), sometimes they are called Web-API

EXAMPLE OF WEB API

Socnet is fiction social application



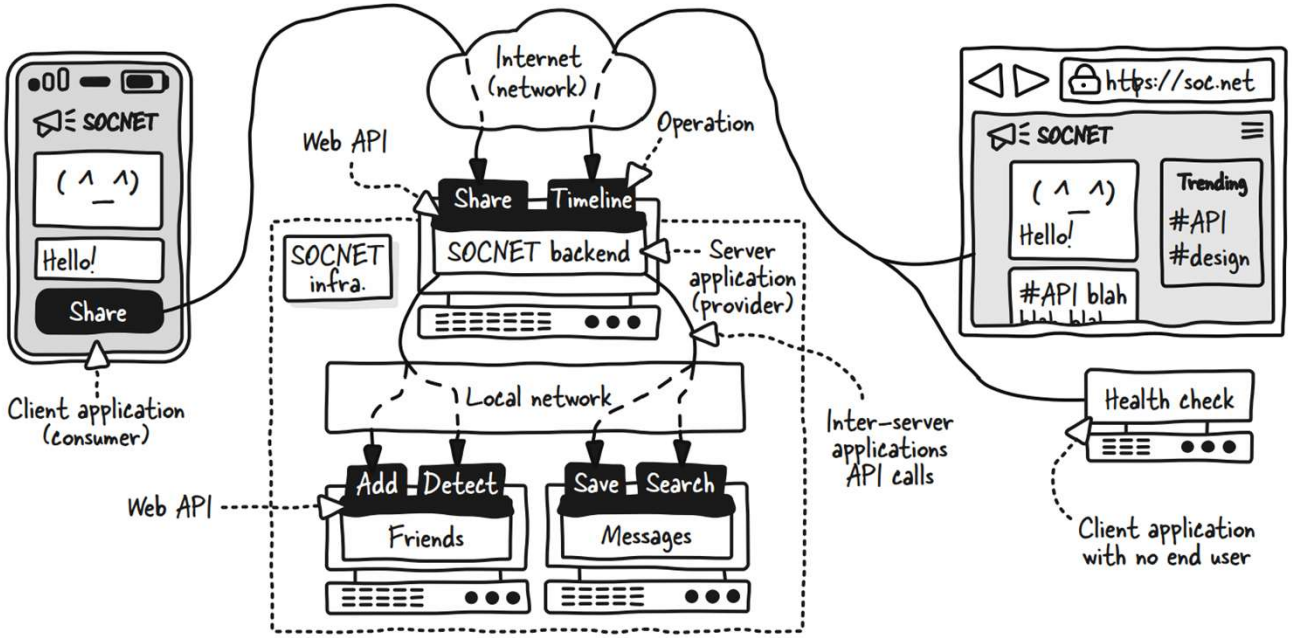


Figure 1.1 The SOCNET web API exposed by the backend has operations that can be called by any kind of application over the internet. The Friends and Messages internal applications also expose web APIs that can be called over the local network.

EXAMPLE OF OPERATION

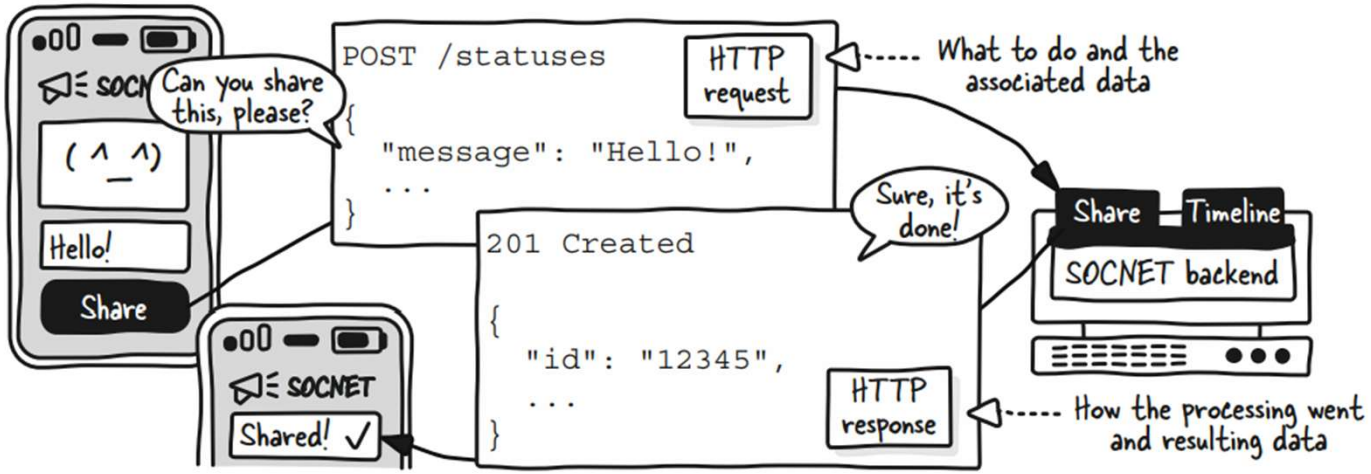
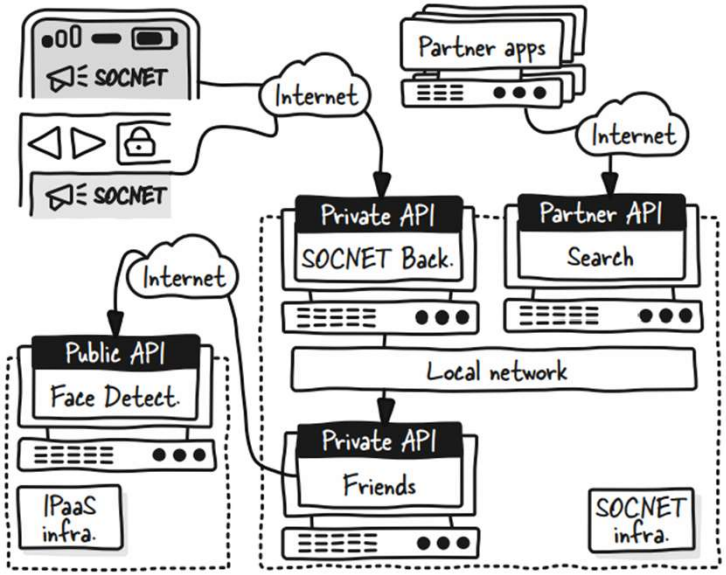


Figure 1.2 When calling a web API, the consumer sends an HTTP request indicating what to do and the needed data. Once the request is processed, the server returns an HTTP response indicating how the processing went and the resulting data.

ACCESS SCOPE



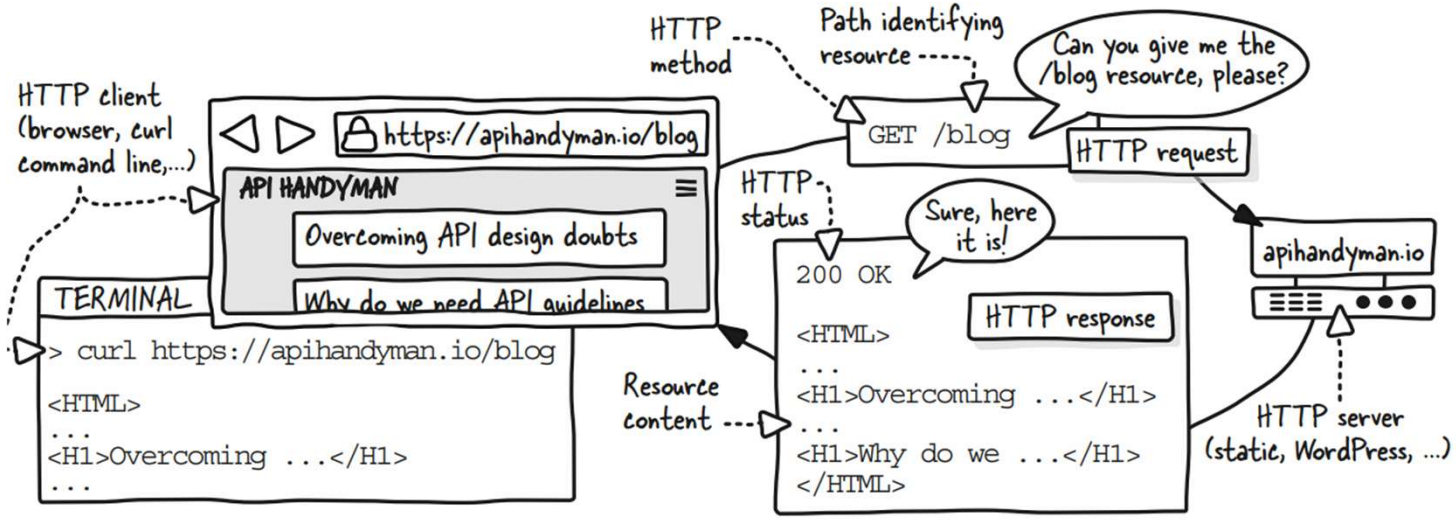
WHO CONSUMES THE API?	THEN IT IS A ...	AND ALSO AN ...
Only the provider	Private API	Internal API
Selcted third parties	Partner API	External API
Anyone*	Public API	

(* Anyone respecting the conditions of use)

WHERE IS THE API EXPOSED?	THEN IT IS AN ...
Local network, intranet	Internal API
Internet	External API

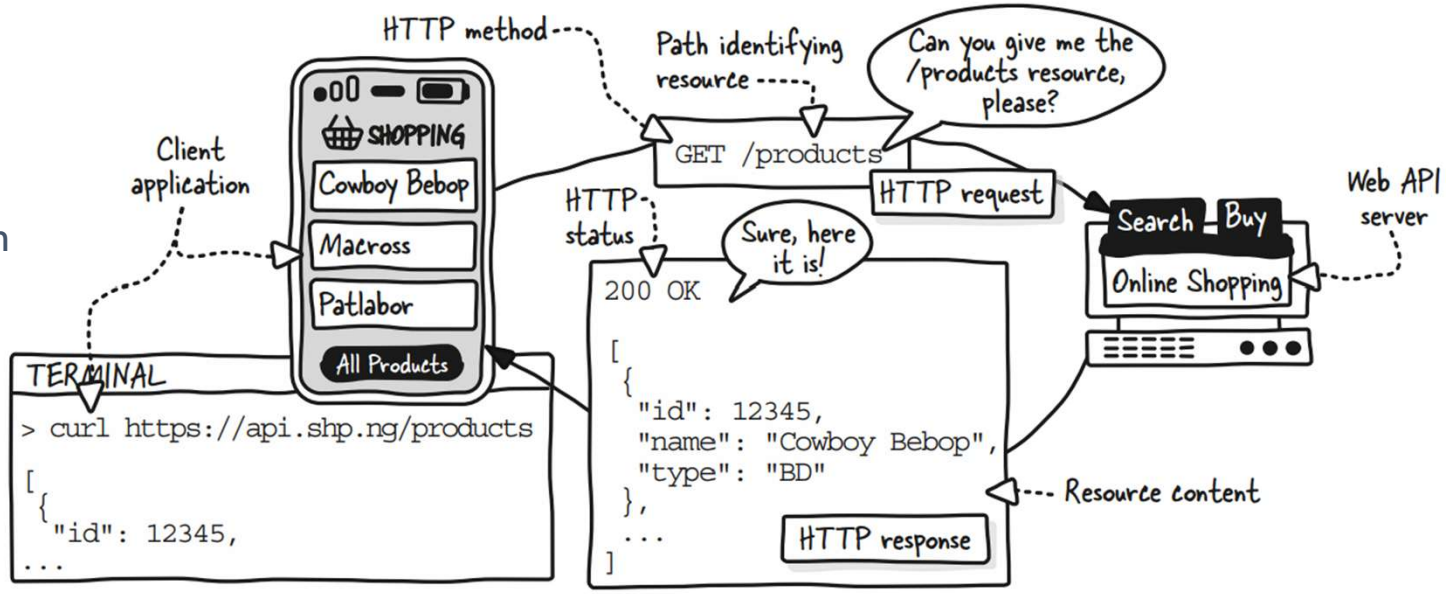
Figure 1.4 The terms *internal*, *external*, *private*, *partner*, and *public* API may need to be disambiguated so everyone involved understands each other.

HTTP


HTTP client (browser, curl command line,...)
https://apihandyman.io/blog
API HANDYMAN
Overcoming API design doubts
Why do we need API guidelines
TERMINAL
> curl https://apihandyman.io/blog
<HTML>
...
<H1>Overcoming ...</H1>
...
HTTP method
Path identifying resource
Can you give me the /blog resource, please?
GET /blog
HTTP request
HTTP status
Sure, here it is!
200 OK
HTTP response
<HTML>
...
<H1>Overcoming ...</H1>
...
<H1>Why do we ...</H1>
</HTML>
Resource content
apihandyman.io
HTTP server (static, WordPress, ...)

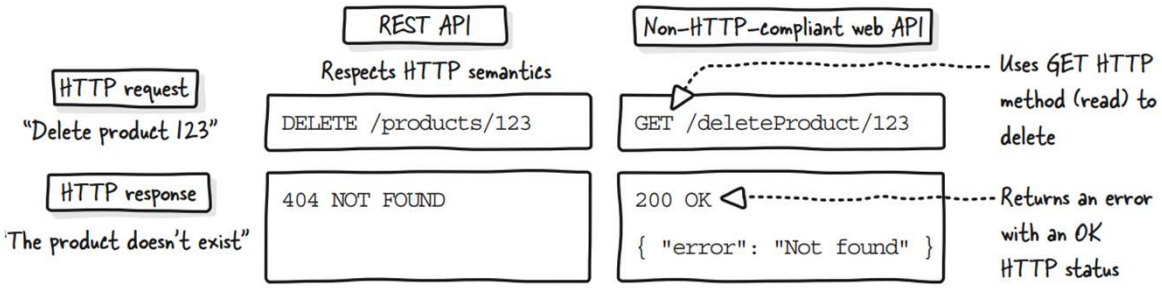
REST API

- In a rest API a client application calls the API in the same way a web browser loads/ an HTML page or interact with it.
- However, the reply is structured data (JSON or XML) rather than HTML./CSS.
- The data model is a resource which represents an entity or concept ("products" in this example related to "Online Shopping").



CONTRASTING REST WITH NON-HTTP-COMPLIANT WEB APIS

- Although all web APIs use HTTP, not all web APIs respect HTTP semantics like REST APIs.
- In this example deleting a nonexistent product with a REST and non-HTTP compliant web API.
- If the product doesn't exist, the server responds with 404 Not Found, like a nonexistent web page request.



RESTFUL API

EXAMPLE OF API CAPABILITY

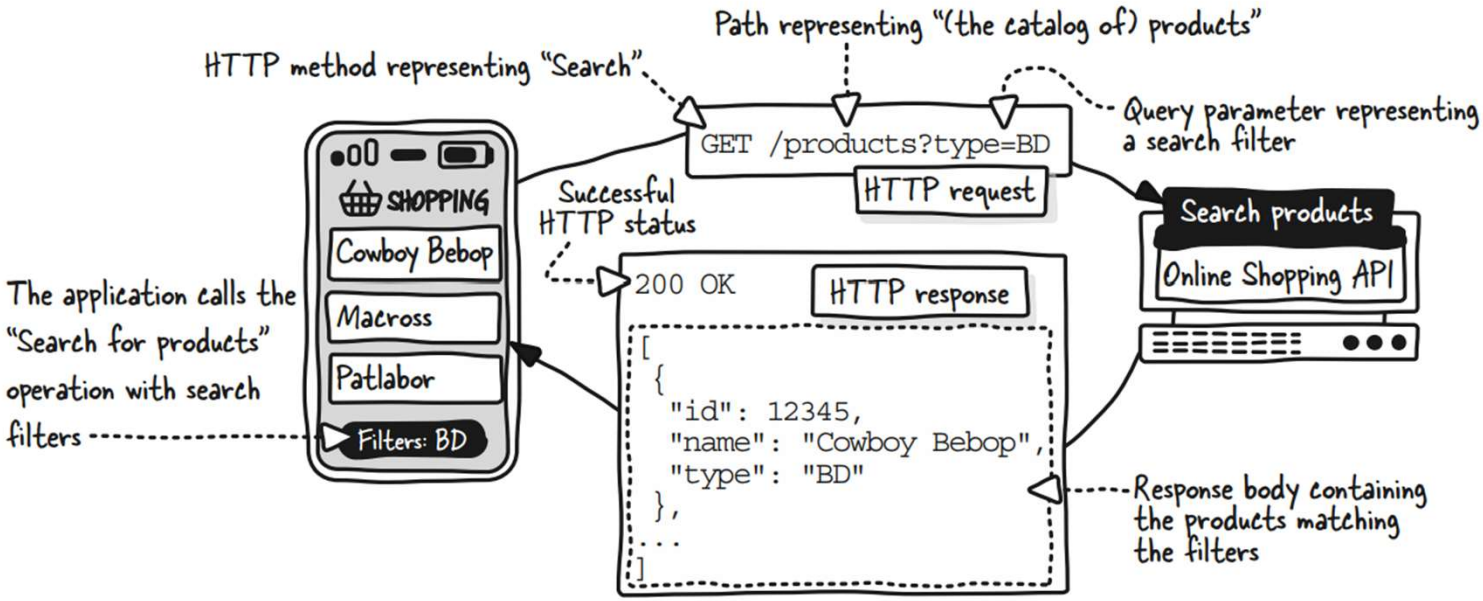
Capabilites identified by analyzing needs

Information collected when observing the operation capabilities from the REST angle

OPERATION	RESOURCE	ACTION	INPUT	OUTPUT		
				Description	Type	Data
Add a product to the catalog	Catalog	Add	Product info	Product added to the catalog	Success	Product info.
				Wrong product information	Error	
Search for products	Contains many	Search	Filters	Products matching filters found	Success	Products info.
				No products matching filters	Success	
Get product details	Product	Get	Product reference	Product found	Success	Product info.
				No product found	Error	
Modify a product		Modify	Product reference, Modified product info	Product modified	Success	
				No product found	Error	
Remove a product from the catalog		Remove	Product reference	Product removed	Success	
				No product found	Error	

- To map the capabilities of the API to REST paradigm one must identify resources, the actions that apply to them, and their inputs and outputs.
- A resource is represented by a path

EXAMPLE


HTTP method representing "Search"
Path representing "(the catalog of) products"
Query parameter representing a search filter
GET /products?type=BD
HTTP request
Search products
Online Shopping API
SHOPPING
Cowboy Bebop
Macross
Patlabor
Filters: BD
Successful HTTP status
200 OK
HTTP response
[
{
"id": 12345,
"name": "Cowboy Bebop",
"type": "BD"
},
...
]
The application calls the "Search for products" operation with search filters
Response body containing the products matching the filters

RELASHIONSHIP AMONG RESOURCES

- Hierarchical Relationship among resources should be mapped to a path hierarchy for easier interpretation..
- For example:
- /merchants/{merchant id}/products/ {product reference} is more readable than
- /products/merchant/{product reference}/ {merchant id}

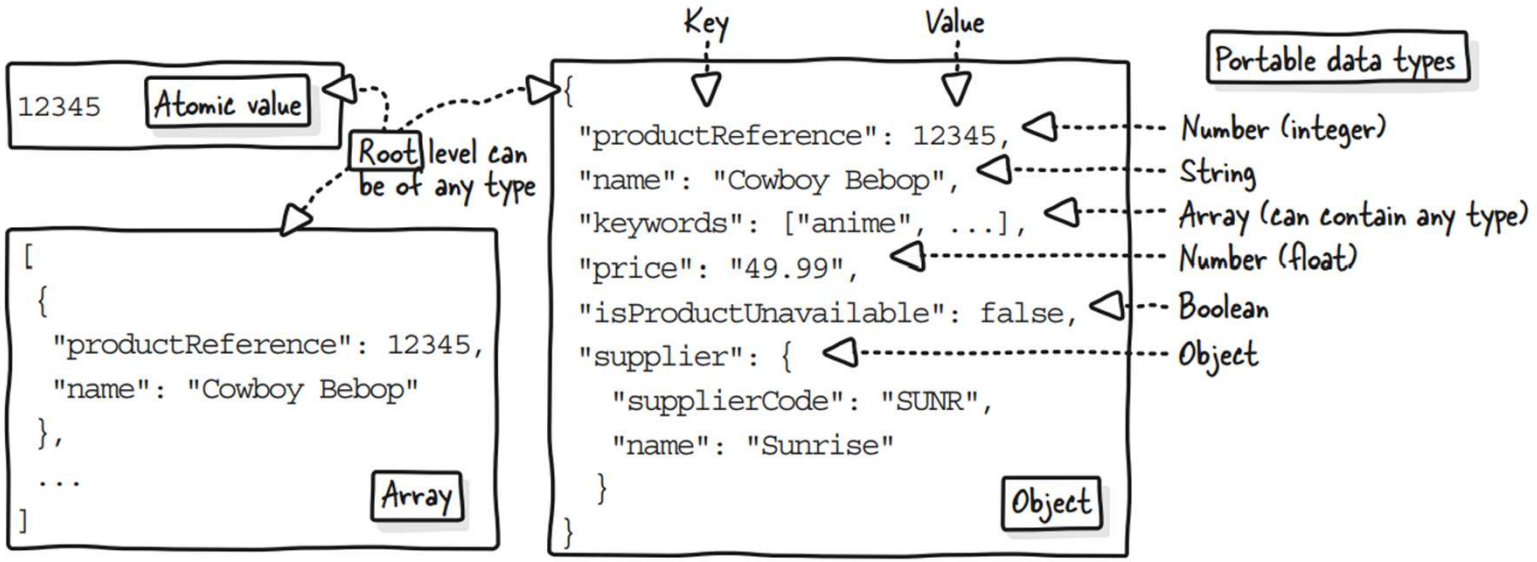
REST INTERFACE

OPERATION	RESOURCE	ACTION	HTTP METHOD	INPUT Description	INPUT	Location
Add a product to the catalog	Catalog	Add	POST	Product information	Resource representation	body
Search for products		Search	GET	Filters	Resource modifier	query
Get product details	Product	Get	GET	Product reference	Resource identifier	path
Modify a product		Modify	PUT or PATCH	Product reference		path
				Modified product info	Resource representation	body
Remove a product from the catalog		Remove	DELETE	Product reference	Resource identifier	path

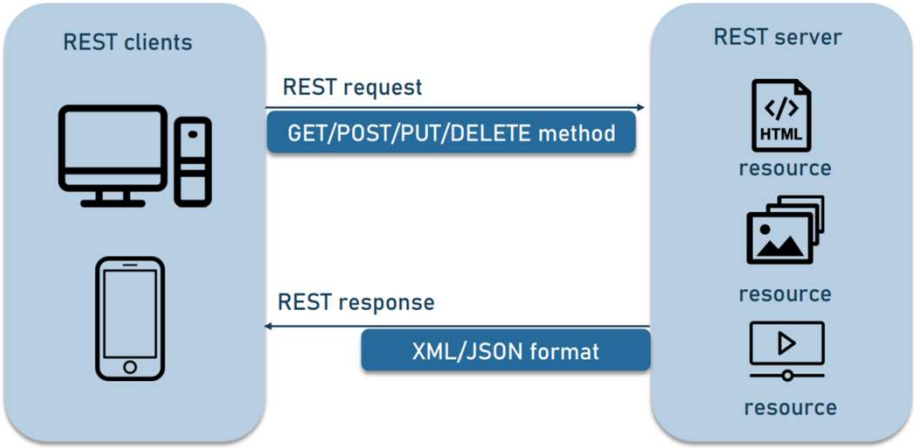
Location is influenced by HTTP method and input data nature

- Just processing data (like do something on the data) is not a CRUD operation.
- However, one can map the processing operation to a resource (and hence to a path), process the data input without creating any resource
- For example: POST /check-out

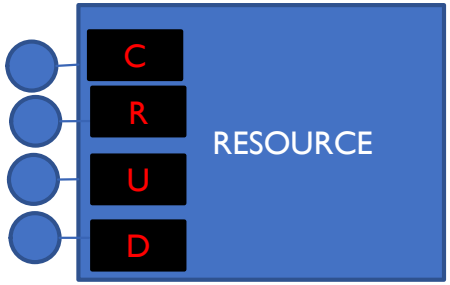
DATA REPRESENTATION

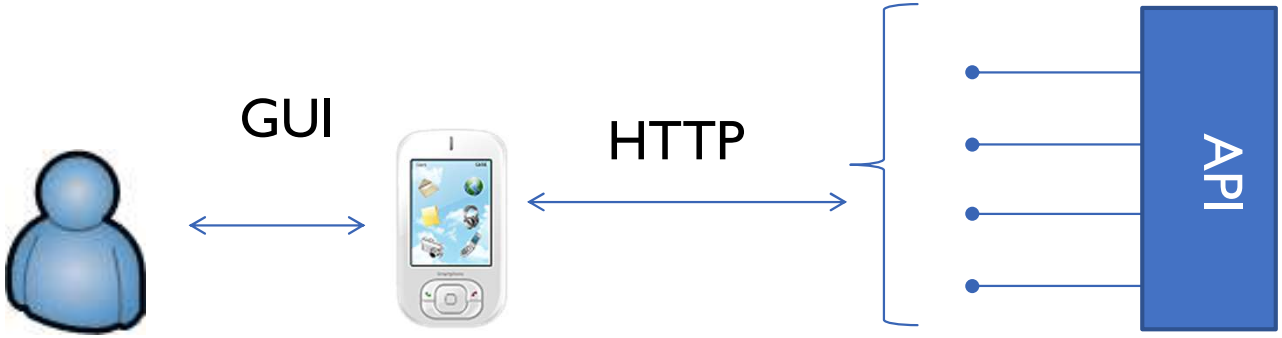


SUMMARY



- GET /USER
- GET /USER/42
- POST /USER
- PUT /USER/42
- DELETE /USER/42





RPC API (OR ACTION – BASED)

- Differently than REST, in the Action-oriented API there is only one URL for all the operations, e.g. /api
- Operations not restricted to CRUD.
- An operation gets input parameters and provides a result, just like any function (procedure)

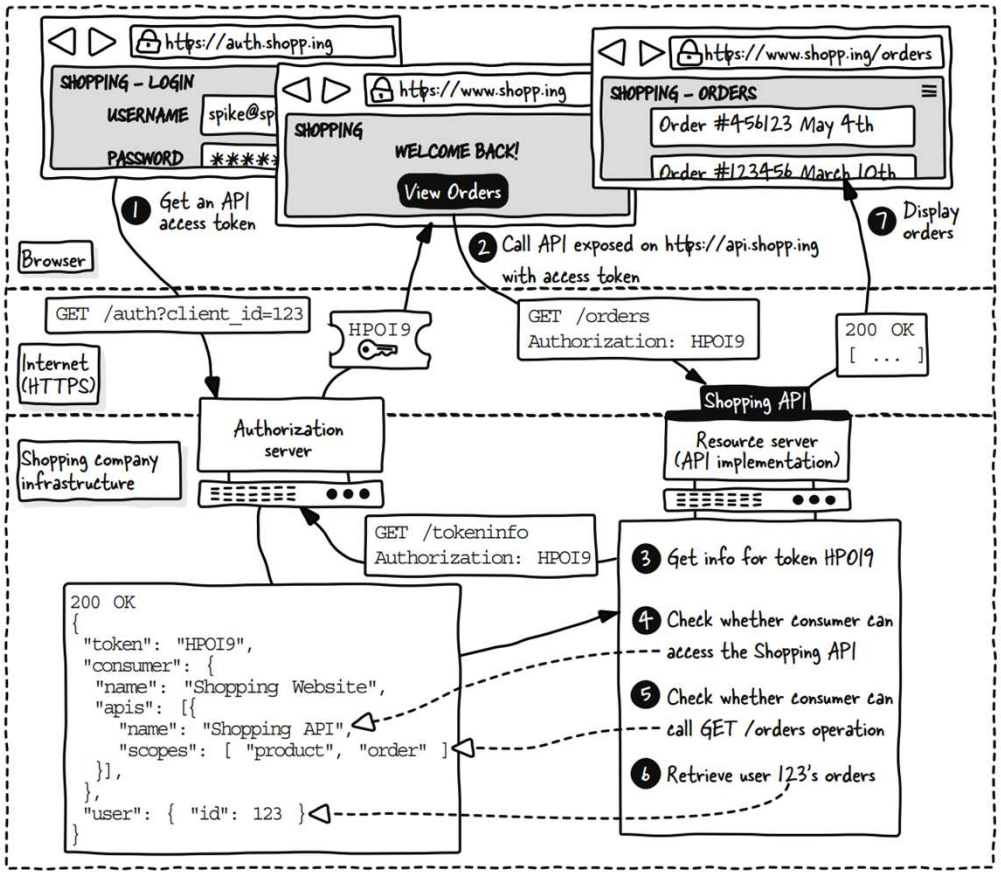
```
POST /api
{
  "action": "createUser",
  "params": {
    "name": "Alice",
    "email": "alice@example.com"
  }
}
```

API SECURITY

- API providers must ensure that only relevant operations and data are accessible to authorized consumers and end users.
- The simplest form of security is by an **API-key** that is added in any API call
- Higher control is implemented using **tokens**.
- A token can regulate (deny or permit) the call to an API (access control).
- A token can define the **scope** of a call (which data can be accessed).
- A token can be used delegated indentity management to other applications
  - OAuth 2.0 / OpenID Connect

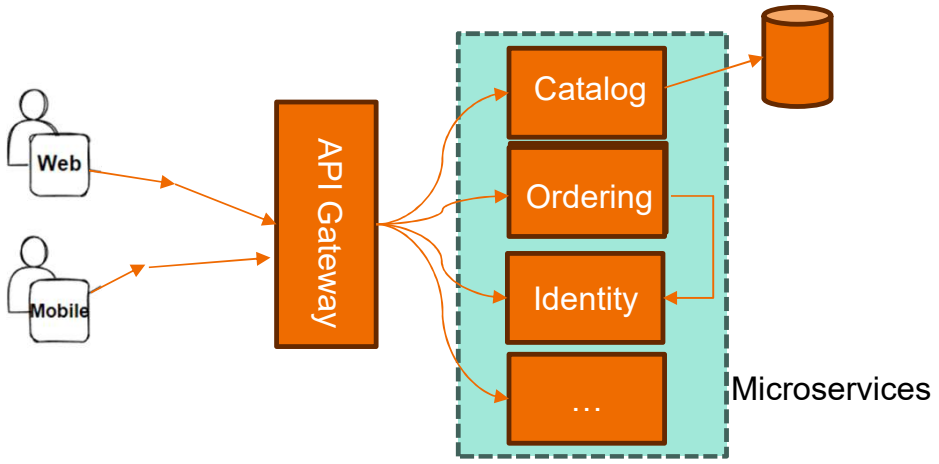
```
GET /api/user/profile
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

EXAMPLE OF ACCESS TOKEN



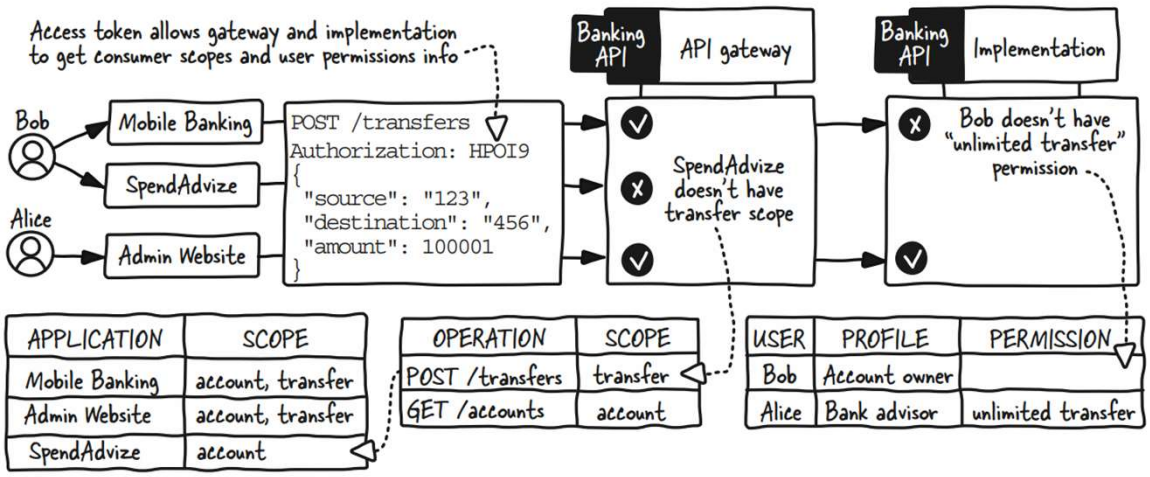
API GATEWAY

- The API Gateway is a single-entry point for a set of microservices.
- It's like a 'smart' proxy that receives requests from the client for services and before routing the request to the specific service, may apply
- Authorization control..
- Rate limiting
- Transformation
- Monitoring and logging …
- Whitelisting and blacklisting
- ..

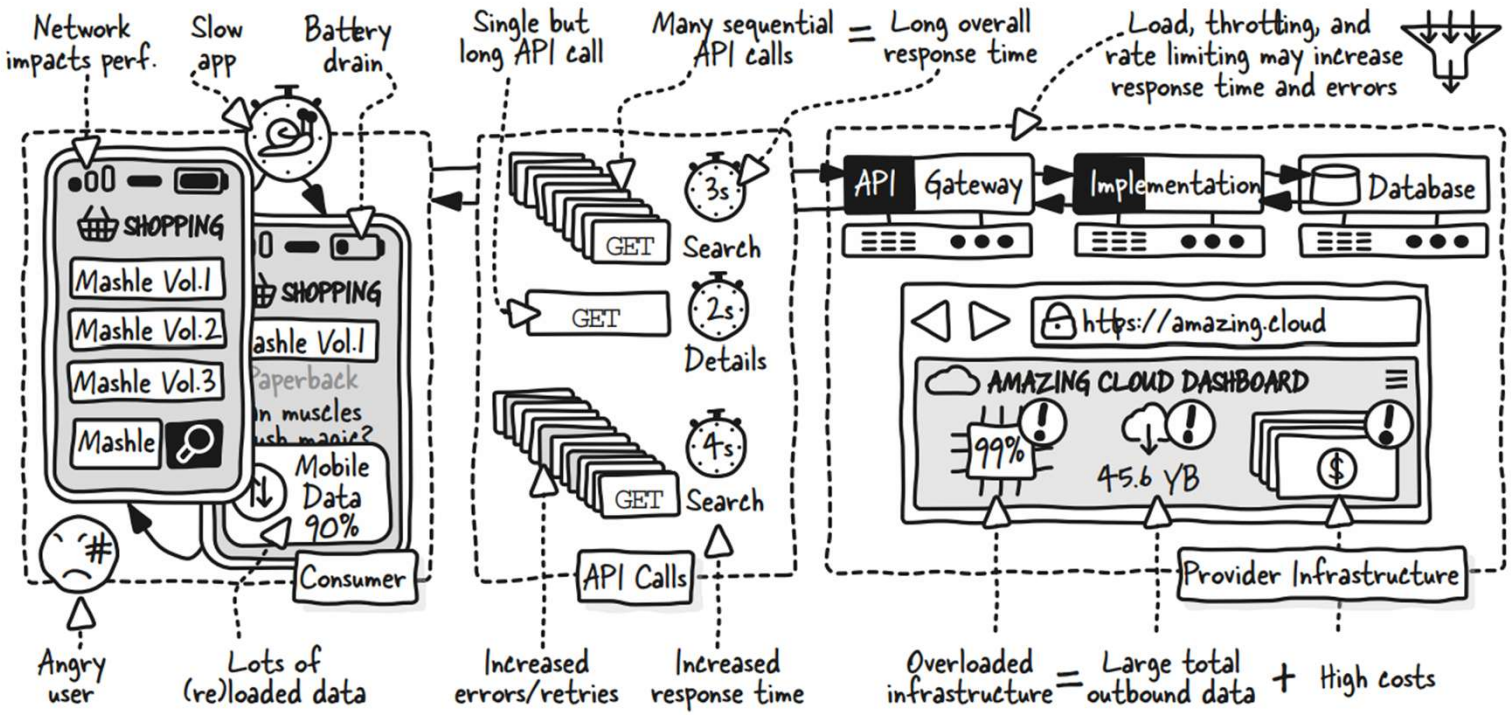


EXAMPLE OF SCOPE CONTROL

- Alice and Bob want see their account an transfer money,
- Bob can't transfer more than €10000
- Alice can transfer any amount she wants
- SpendAdvize can only see information on accounts



API EFFICIENCY



OTHER TYPES OF API

- **gRPC** (Google Remote Procedure Call) is an open-source framework developed by Google for high-performance communication between services.
- It fully uses HTTP/2's possibilities, including bidirectional streaming. Messages use the Protobuf binary format, which is more compact and has more efficient serialization than JSON.
- **graphQL**, developed internally at Facebook in 2012 to address issues with over-fetching and under-fetching data in their mobile apps. It was publicly released in 2015 as an open standard.
- GraphQL allows clients to specify **exactly** which fields they want in a query. This reduces over-fetching and under-fetching of data compared to traditional REST endpoints.
- **Callback API**: An API can be used asynchronously, which means that the replies is set back to the client (much) later than the request. Avoids to block the client
- Client registers a callback endpoint that the server uses to notify the reply

TOOLS TO DESIGN AN API

- **OpenAPI Specification** (formerly known as Swagger Specification) is an open-source format for describing and documenting APIs.

API CATALOG

- https://apislist.com/

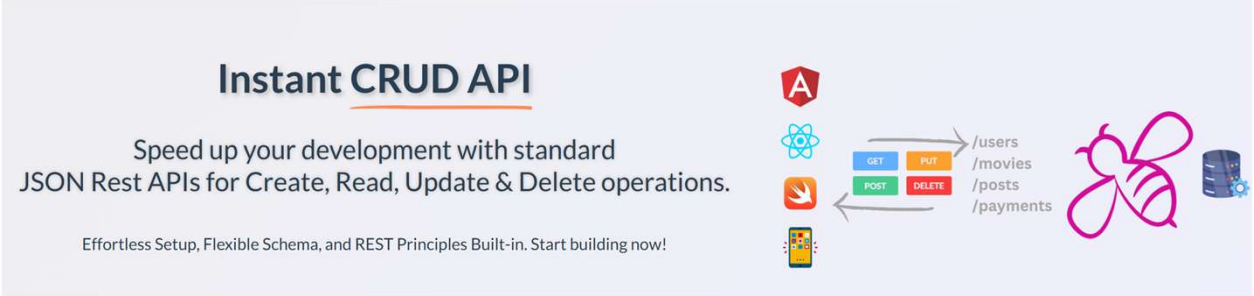
SOME EXAMPLE OF WEB-API

- Many providers sell weather related information through a single URL (http get) and api key authorization
- Free subscription plan or pro
- Information are about:
- Current weather
- Forecast
  - Hourly forecast 4 days
  - Daily forecast 16 days
- Solar irradiance forescast
- Solar panel prediction
- Air pollution
- ..

JSON format API response example


```
{
   "coord": {
      "lon": 7.367,
      "lat": 45.133
   },
   "weather": [
      {
         "id": 501,
         "main": "Rain",
         "description": "moderate rain",
         "icon": "10d"
      }
   ],
   "base": "stations",
   "main": {
      "temp": 284.2,
      "feels_like": 282.93,
      "temp_min": 283.06,
      "temp_max": 286.82,
      "pressure": 1021,
      "humidity": 60,
      "sea_level": 1021,
      "grnd_level": 910
   },
```

EXAMPLE OF CRUD-API



- Good for rapid prototype and demo

FB API

- Main interface for accessing Facebook data programmatically.
- Represents data as a graph: nodes (objects) and edges (connections).
- Used by apps and integrations to interact with Facebook's social graph.