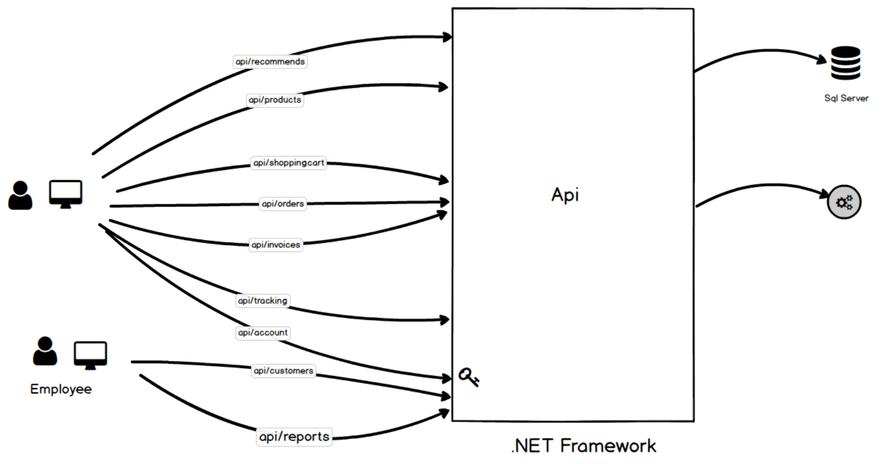
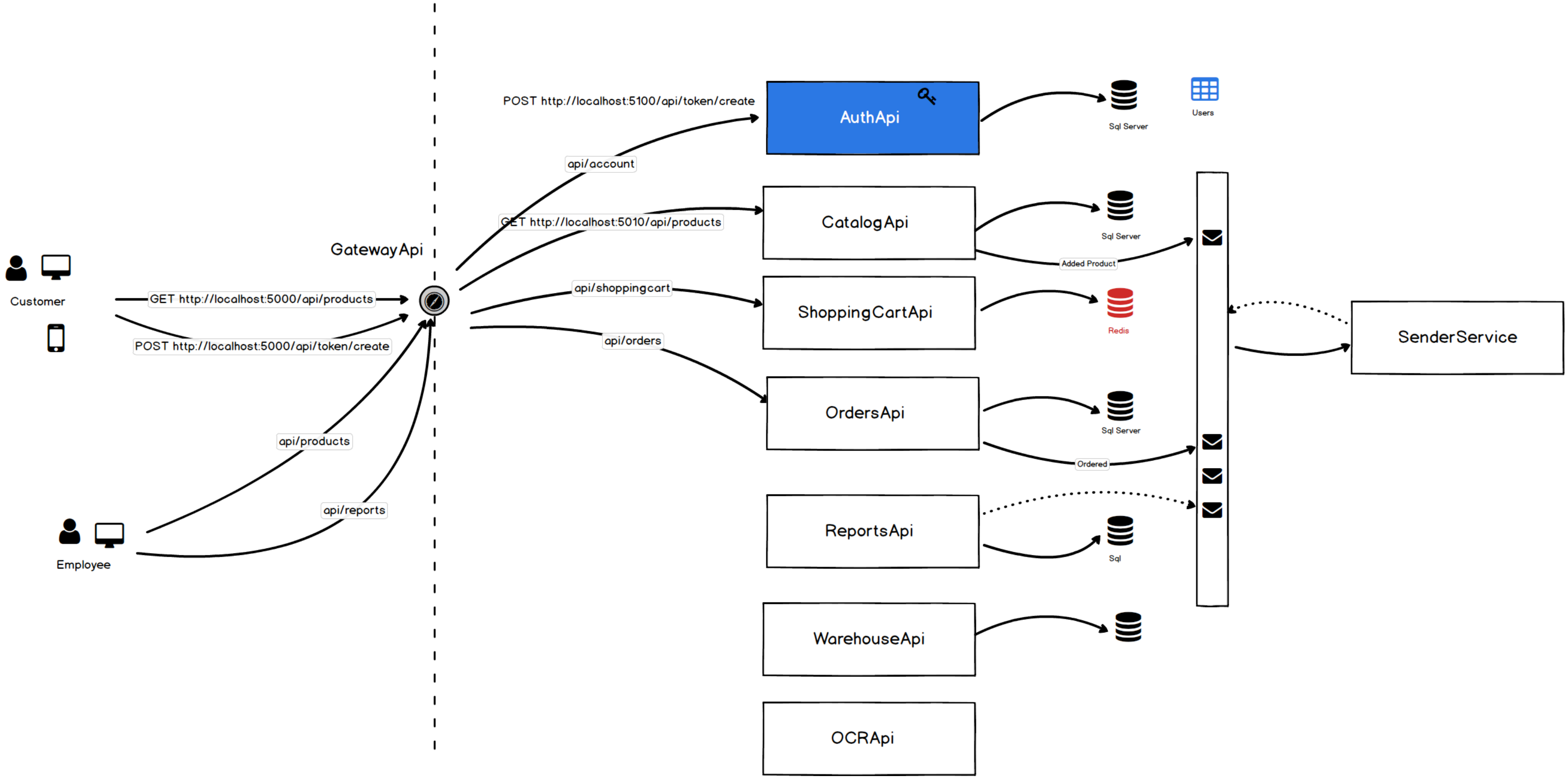
Monolit

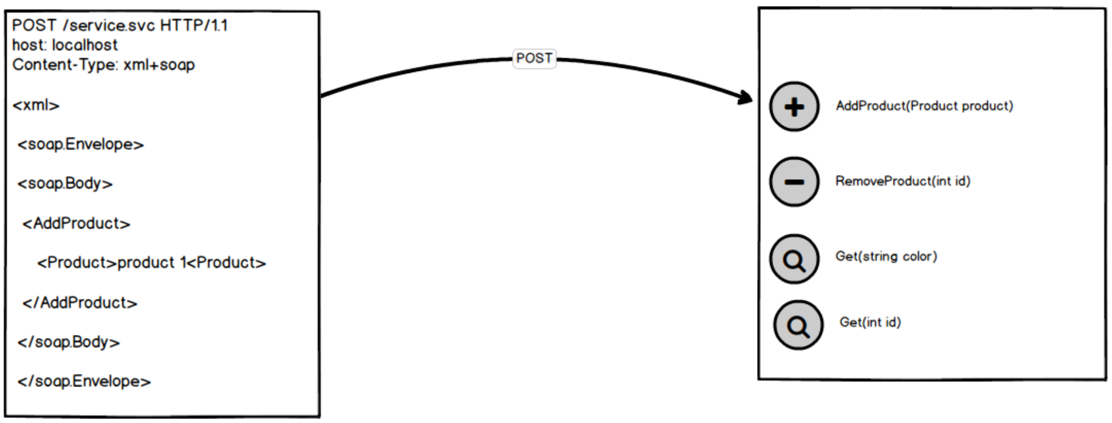


Microservices



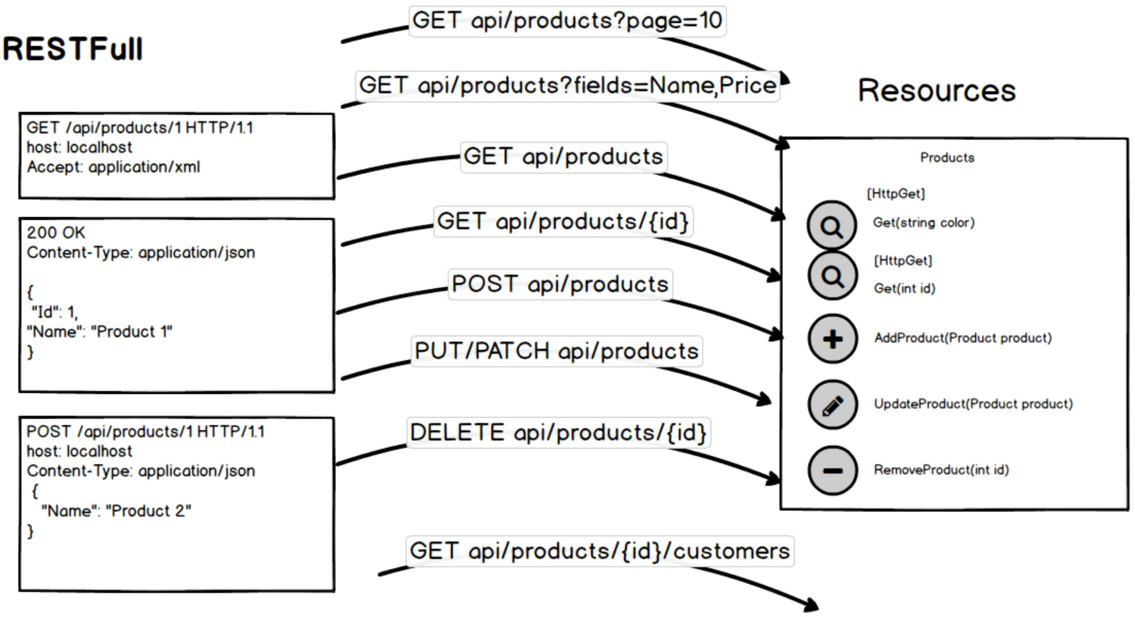
SOAP

RPC



```
POST /service.svc HTTP/1.1
host: localhost
Content-Type: xml+soap

<xml>

 <soap.Envelope>

 <soap.Body>

  <AddProduct>

     <Product>product 1<Product>

  </AddProduct>

 </soap.Body>

 </soap.Envelope>
```

RESTFull

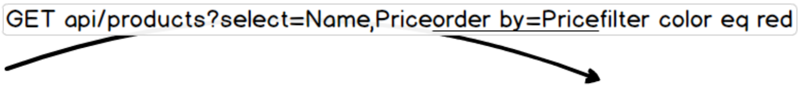


```
GET /api/products/1 HTTP/1.1
host: localhost
Accept: application/xml
```

```
200 OK
Content-Type: application/json

{
 "Id": 1,
"Name": "Product 1"
}
```

```
POST /api/products/1 HTTP/1.1
host: localhost
Content-Type: application/json
 {
   "Name": "Product 2"
}
```

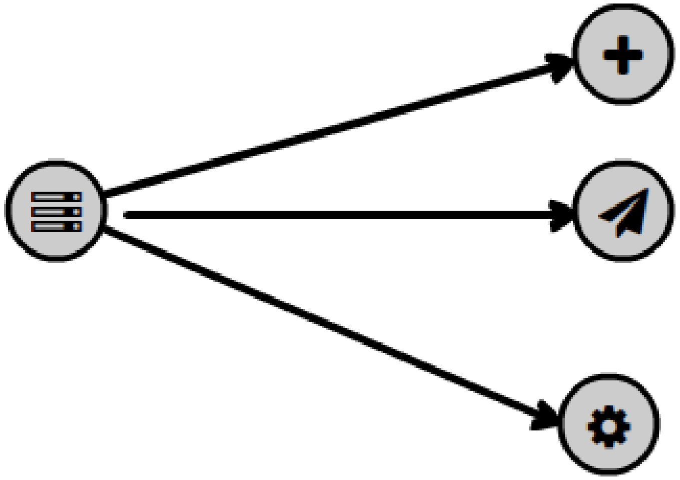
OData



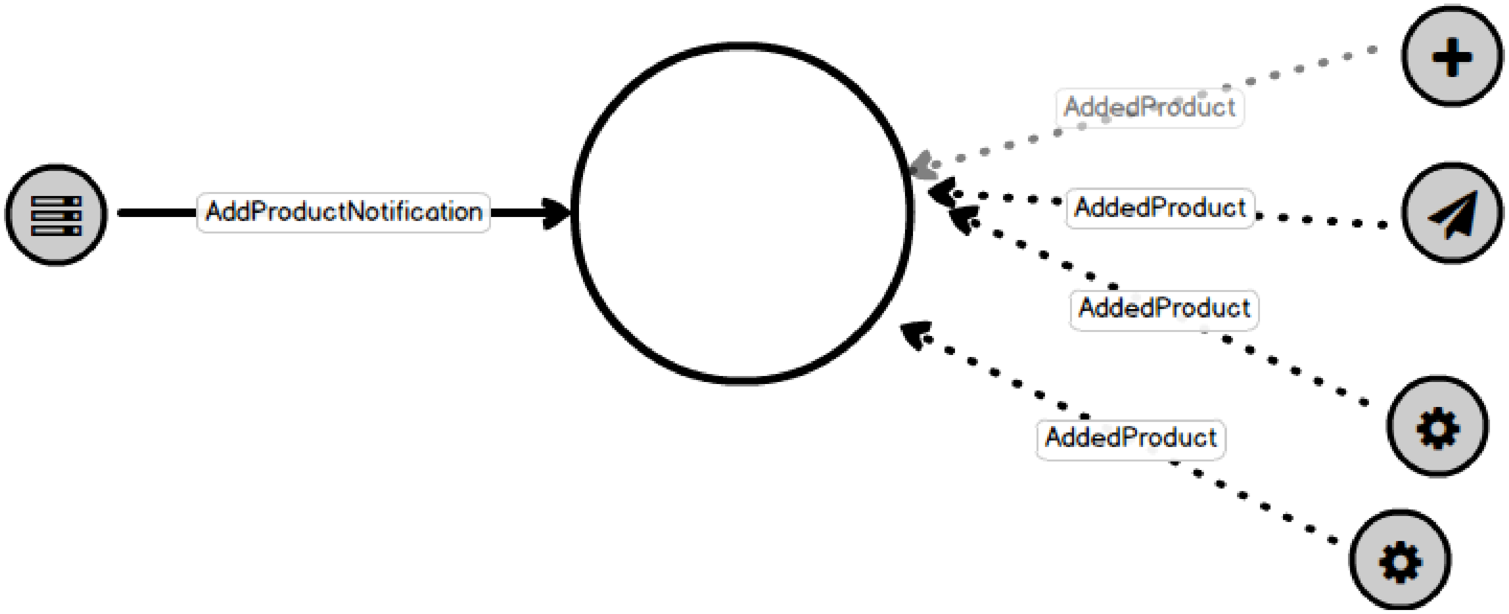
GET api/products?select=Name,Priceorder by=Pricefilter color eq red

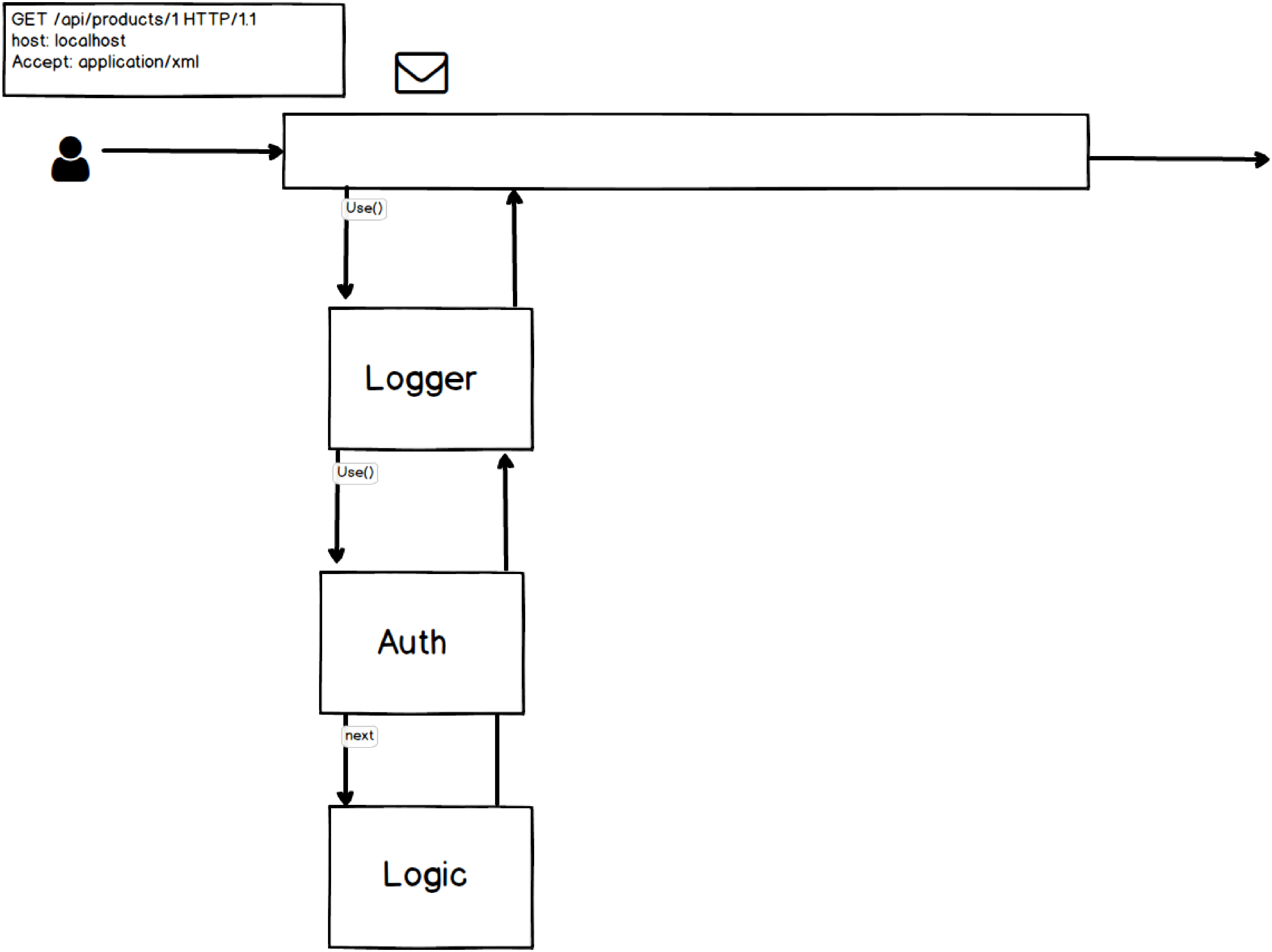
GraphQL

```
POST/graphql HTTP/1.1
host: localhost

{
   "Products" :  { name, price,
Customers }



}
```

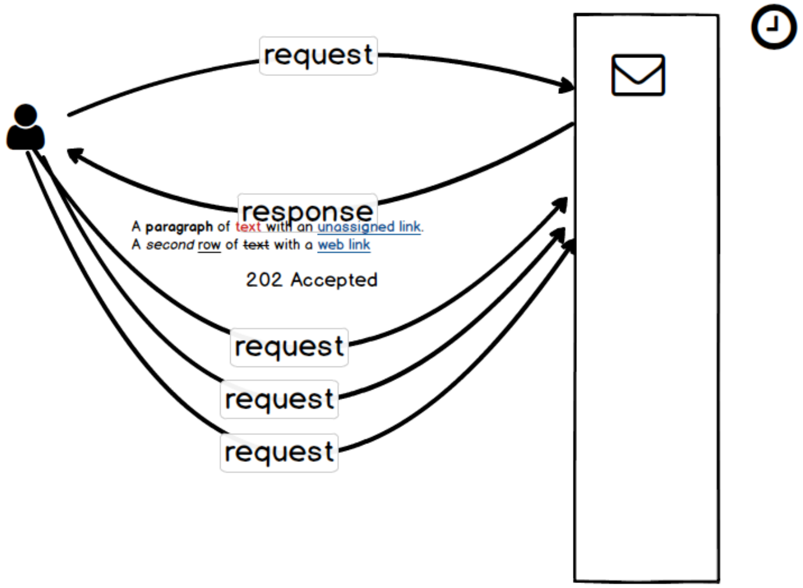


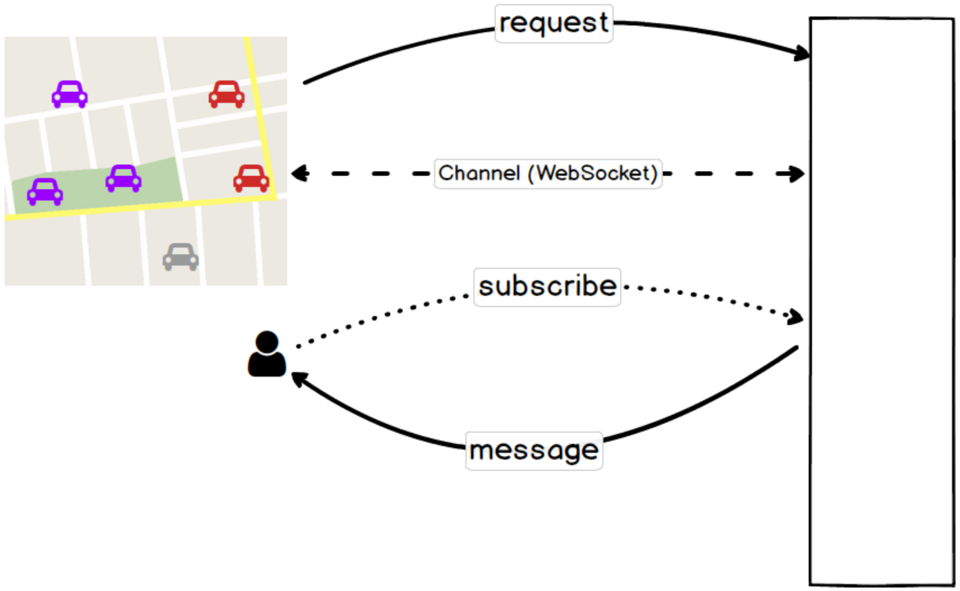
Mediator


AddProductNotification
AddedProduct
AddedProduct
AddedProduct
AddedProduct

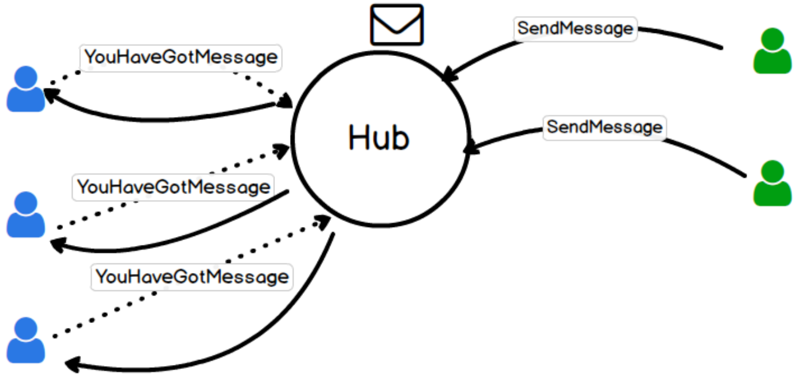


```
GET /api/products/1 HTTP/1.1
host: localhost
Accept: application/xml
```

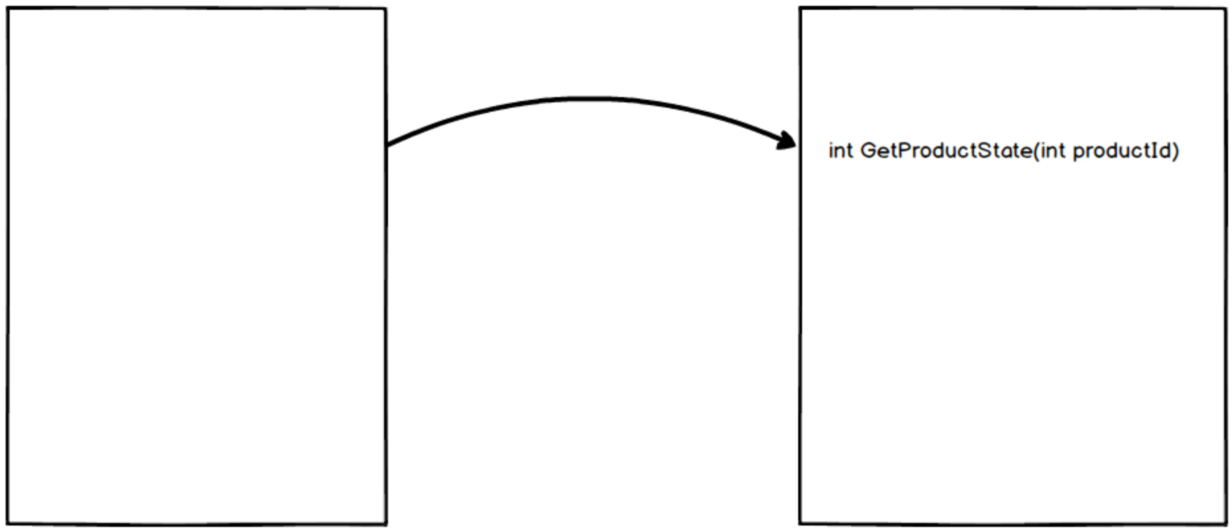

request
response
A paragraph of text with an unassigned link.
A second row of ~~text~~ with a web link
202 Accepted
request
request
request


request
Channel (WebSocket)
subscribe
message

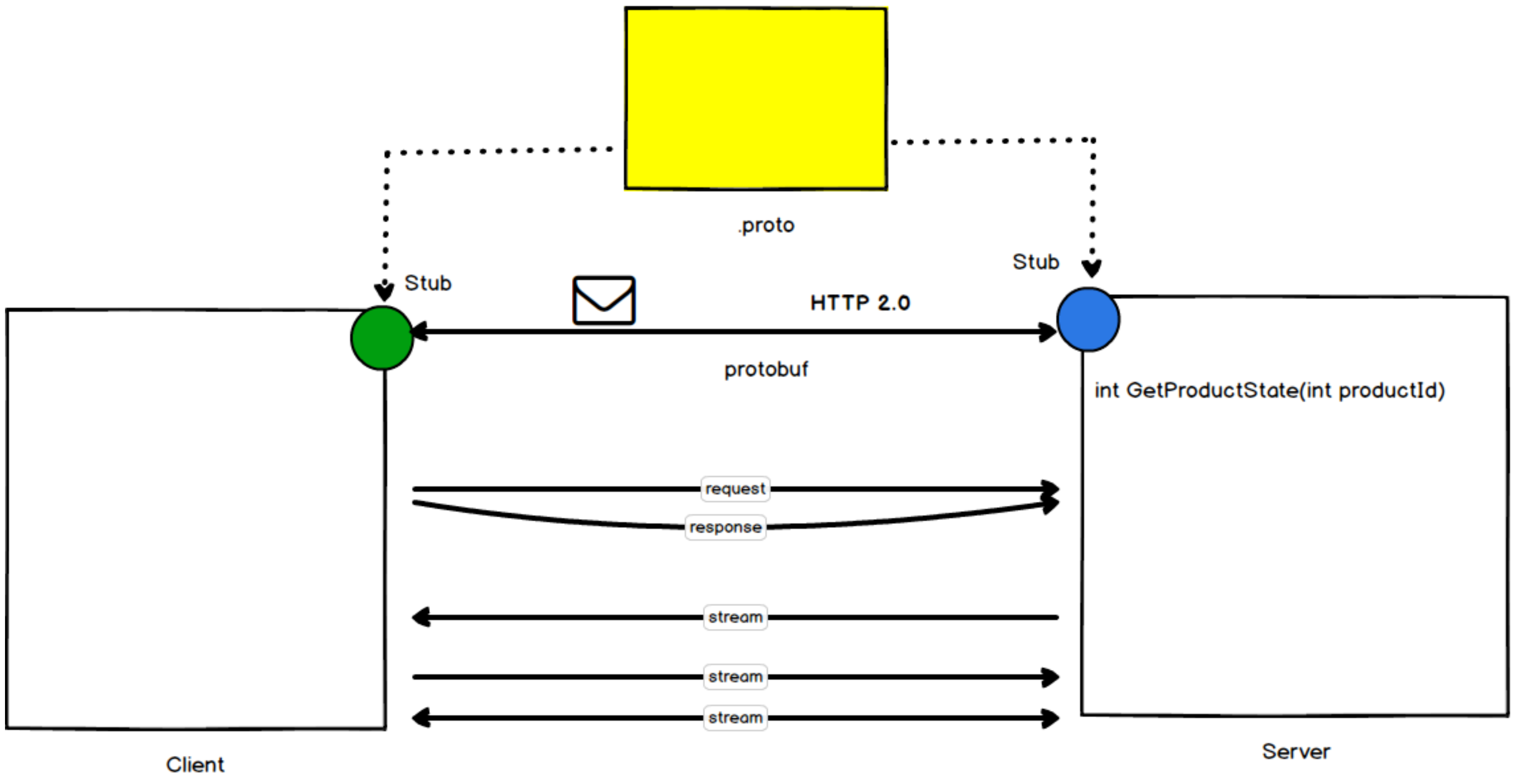
Signal-R

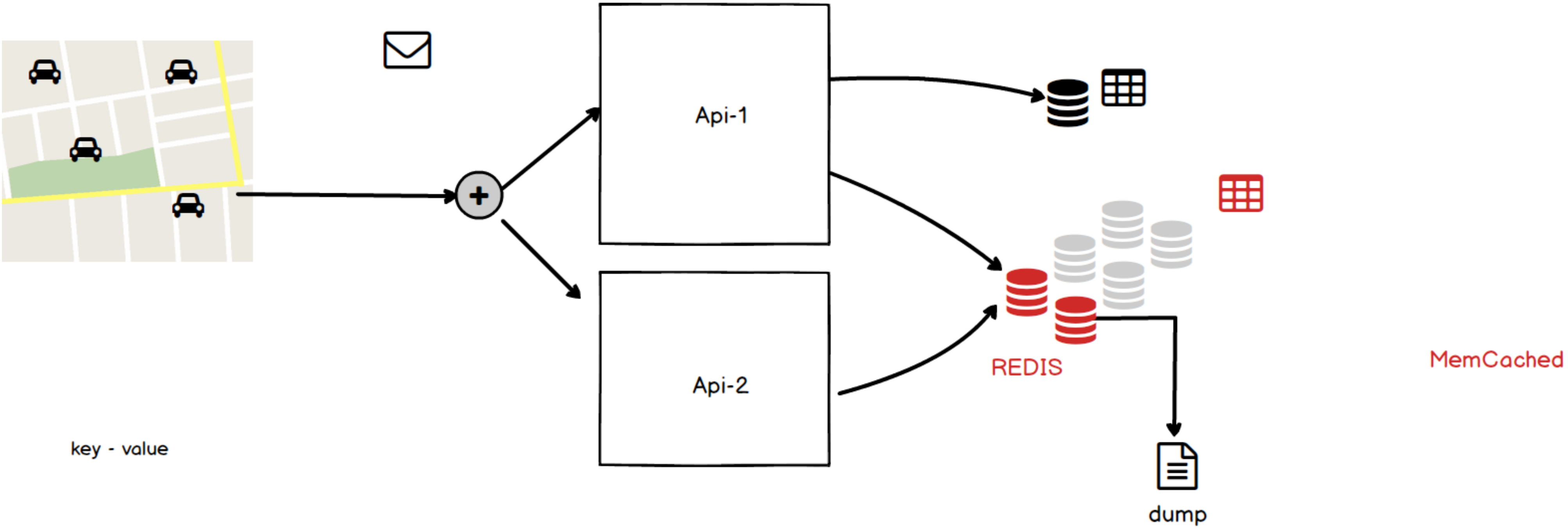

YouHaveGotMessage
SendMessage
Hub
YouHaveGotMessage
SendMessage
YouHaveGotMessage

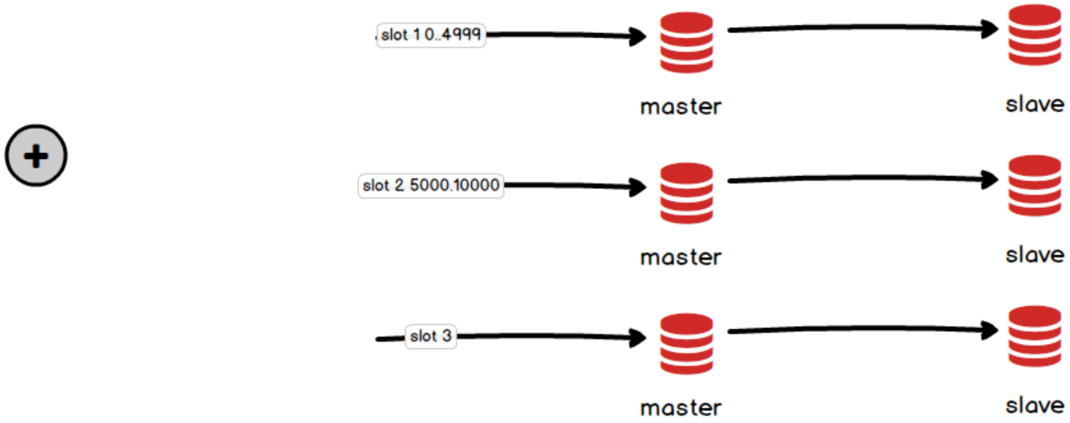
RPC

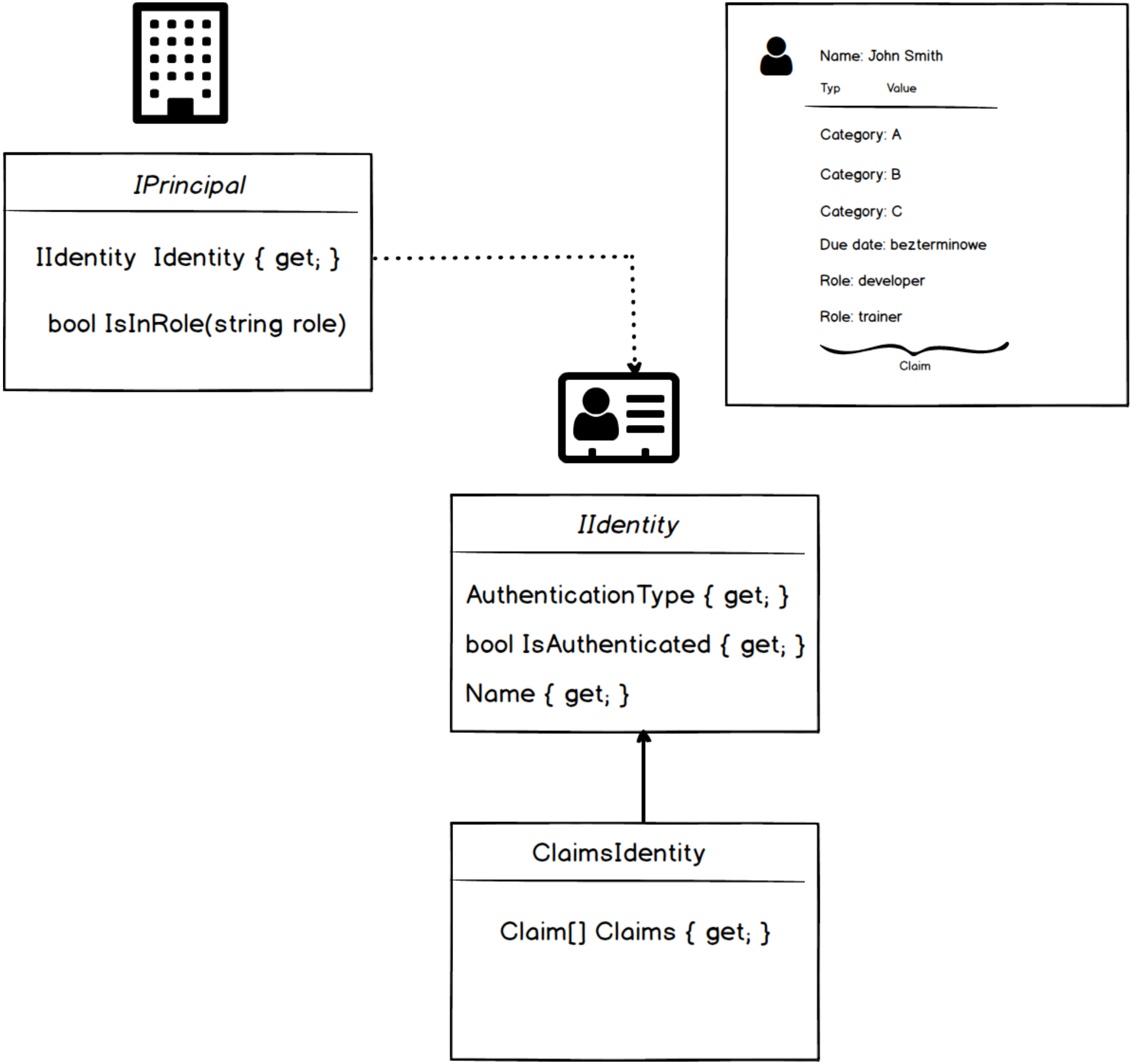

int GetProductState(int productId)

gRPC


.proto
Stub
Stub
HTTP 2.0
protobuf
int GetProductState(int productId)
request
response
stream
stream
stream
Client
Server


Api-1
Api-2
REDIS
MemCached
dump
key - value


slot 1 0..4999
master
slave
slot 2 5000.10000
master
slave
slot 3
master
slave



IPrincipal

IIdentity Identity { get; }

bool IsInRole(string role)

Name: John Smith

Typ	Value
Category:	A
Category:	B
Category:	C
Due date:	bezterminowe
Role:	developer
Role:	trainer

Claim

IIdentity

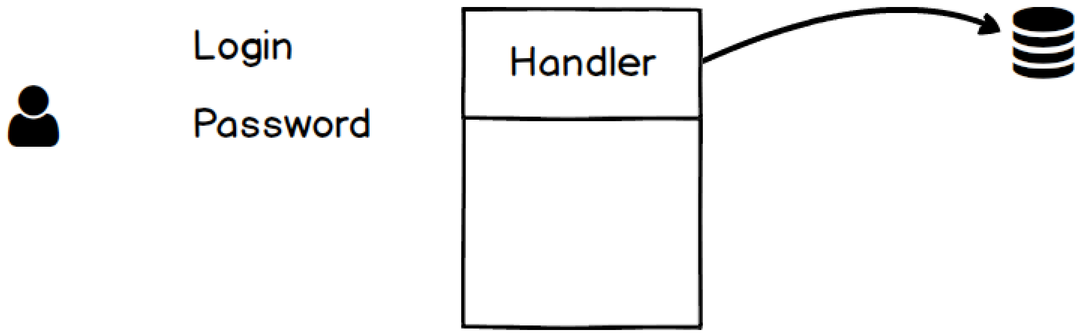
AuthenticationType { get; }

bool IsAuthenticated { get; }

Name { get; }

ClaimsIdentity

Claim[] Claims { get; }



Login

Password

Handler

Token

ACCCCC

exp:

BCCCC
XCCCC
YCCCC
ZCCCC
QCCCC

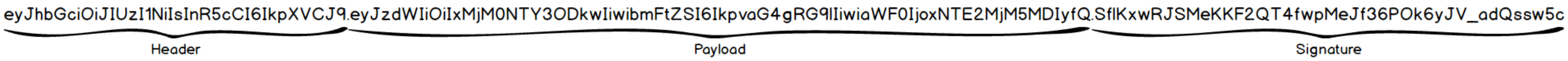
```
POST /service.svc HTTP/1.1
host: localhost
Content-Type: application/json
Authorization: Basic Base64({login}:{password})
```

```
GET /service.svc HTTP/1.1
host: localhost
Content-Type: application/json
Authorization: Basic Base64({login}:{password})
```

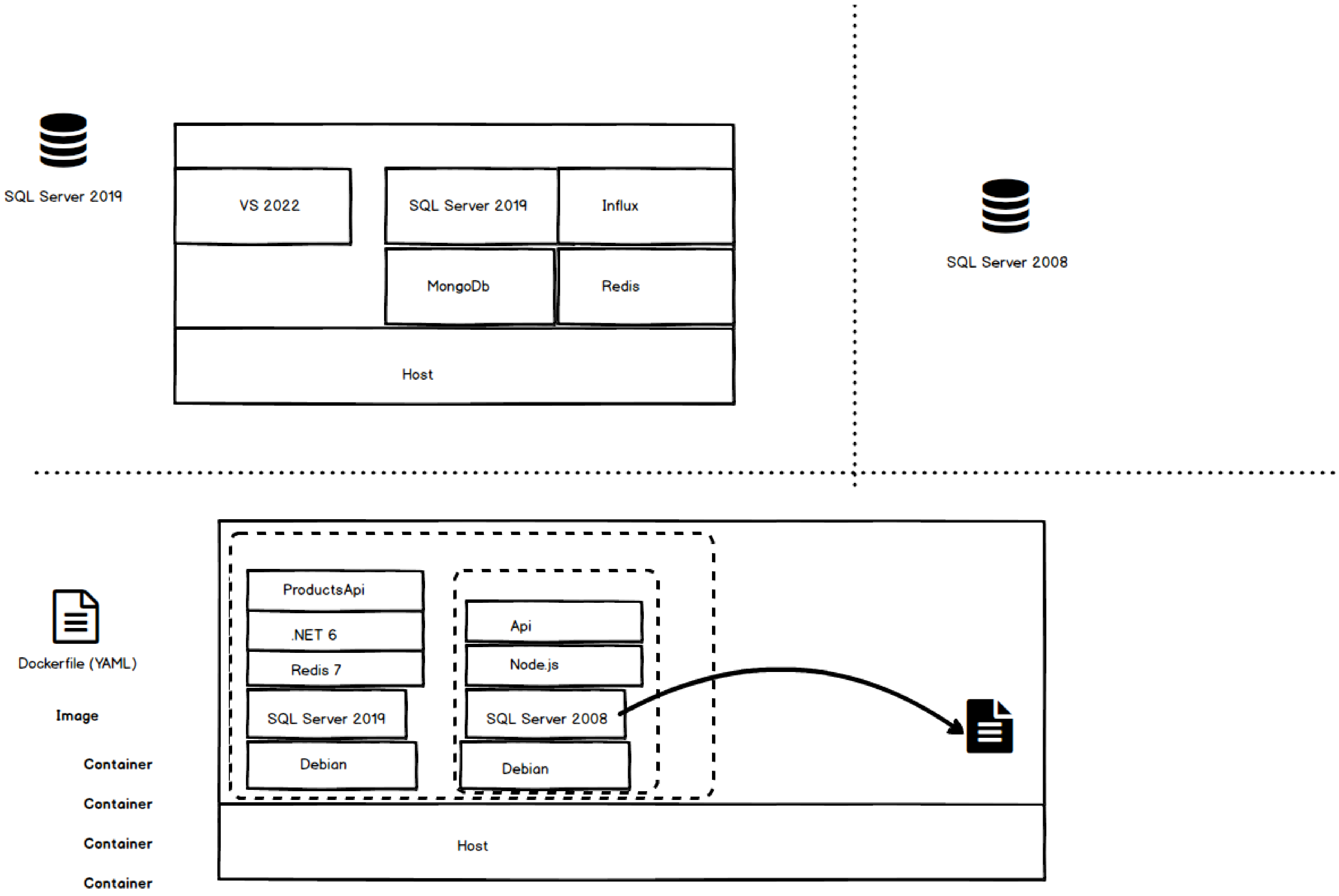
```
GET /service.svc HTTP/1.1
host: localhost
Content-Type: application/json
Authorization: Bearer {token}
Cookies: {value}
```

```
Authorization: {scheme} {parameter}
```

JWT (Json Web Token)



eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c



SQL Server 2019

VS 2022

SQL Server 2019

Influx

MongoDb

Redis

Host

SQL Server 2008

Dockerfile (YAML)

Image

Container

Container

Container

Container

ProductsApi

.NET 6

Redis 7

SQL Server 2019

Debian

Api

Node.js

SQL Server 2008

Debian

Host

DockerCompose compose.yml

> DockerCompose up