Building asynchronous micro-services that get along

王浩宇
Makara Wang
makara@wiredcraft.com

2016.09.03

Before we start...

2012, HuJS, API + Static clients

We've been exploring

Now everybody talk about...

Micro-services

so what is it?

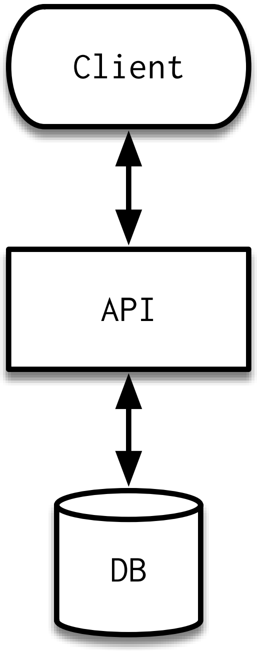
Let's start from the beginning...

RESTful API

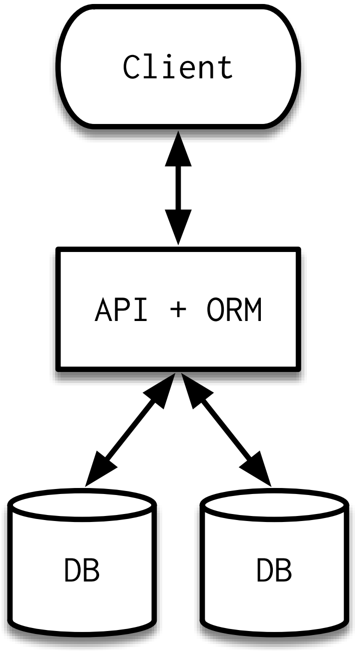
The standard, the simplest, the most well understood

We use [JSON API](1)

the simplest...



when more DB or storage involved...



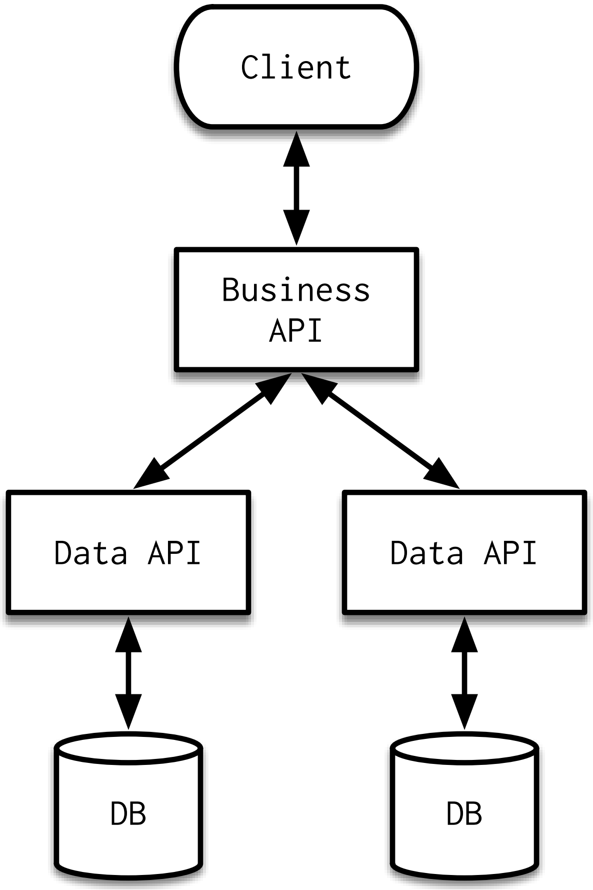
Client

API + ORM

DB

DB

scaling...



…and now we have “micro-services”

Micro-services was born for...

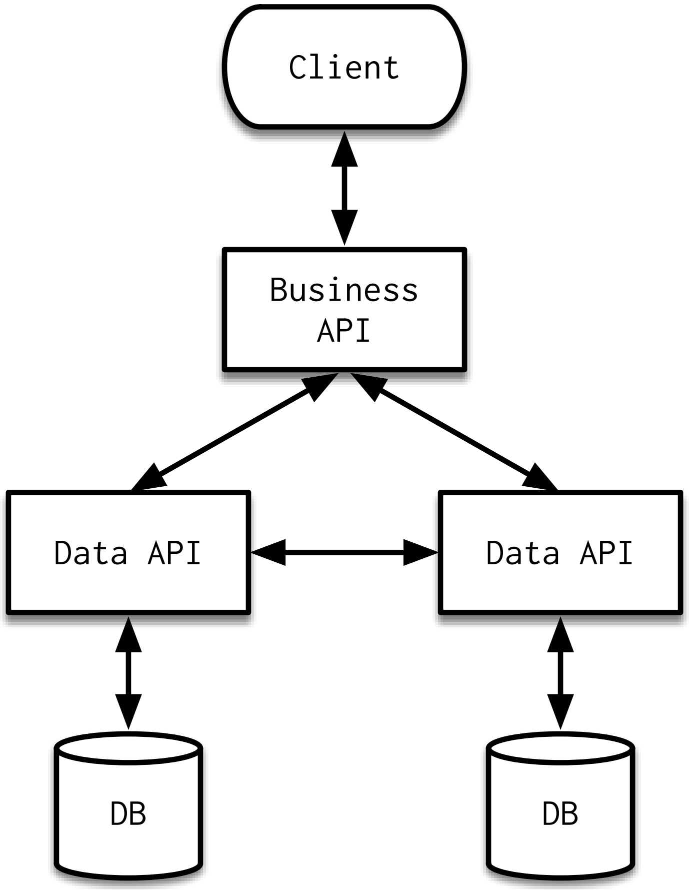
Distributed systems

Continuous delivery, DevOps...

Problem arisen

Communication between services

RESTful all the way...



However...

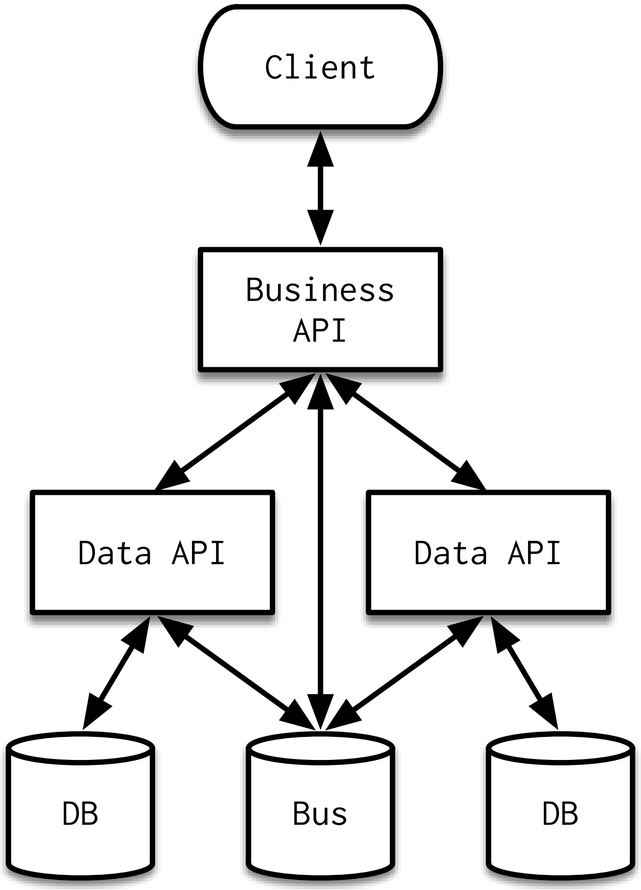
That only covers Req/Rep

That gets confusing easily

We need Pub/Sub

We need "Job" (retry/delay/...)

with a message queue...



Message Queue / Bus

Async, 1 to n

[Queues.io](2)

We use [NSQ](3) and many

Problem arisen

Requests become too heavy

Decoupling an API…

One API one thing?

Sparse fieldsets with query string?

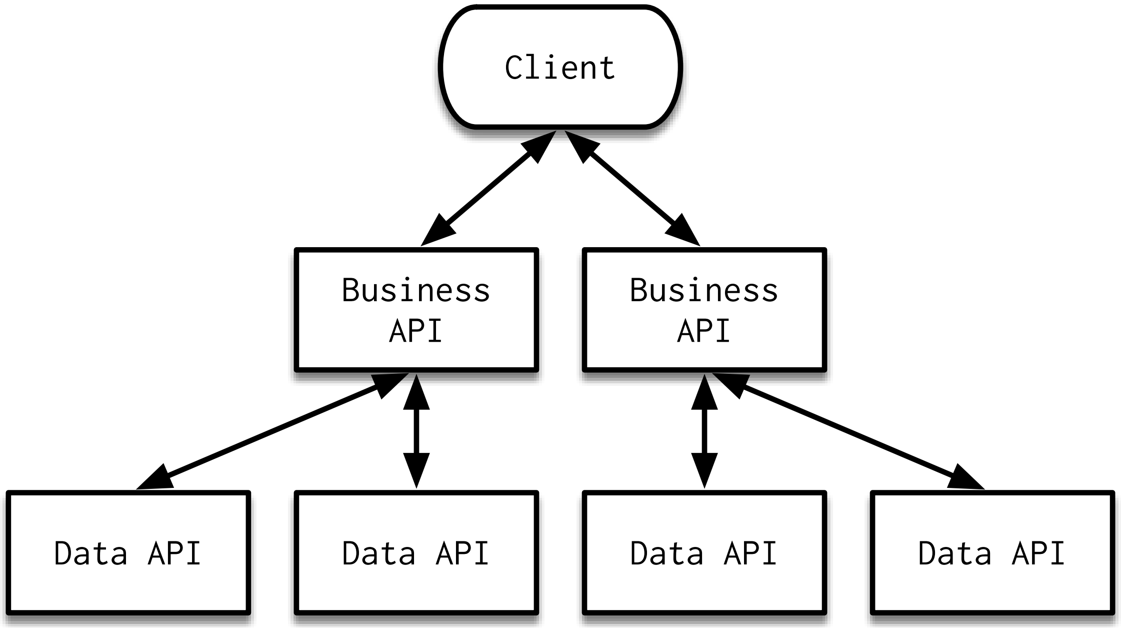
GraphQL

Shared schema

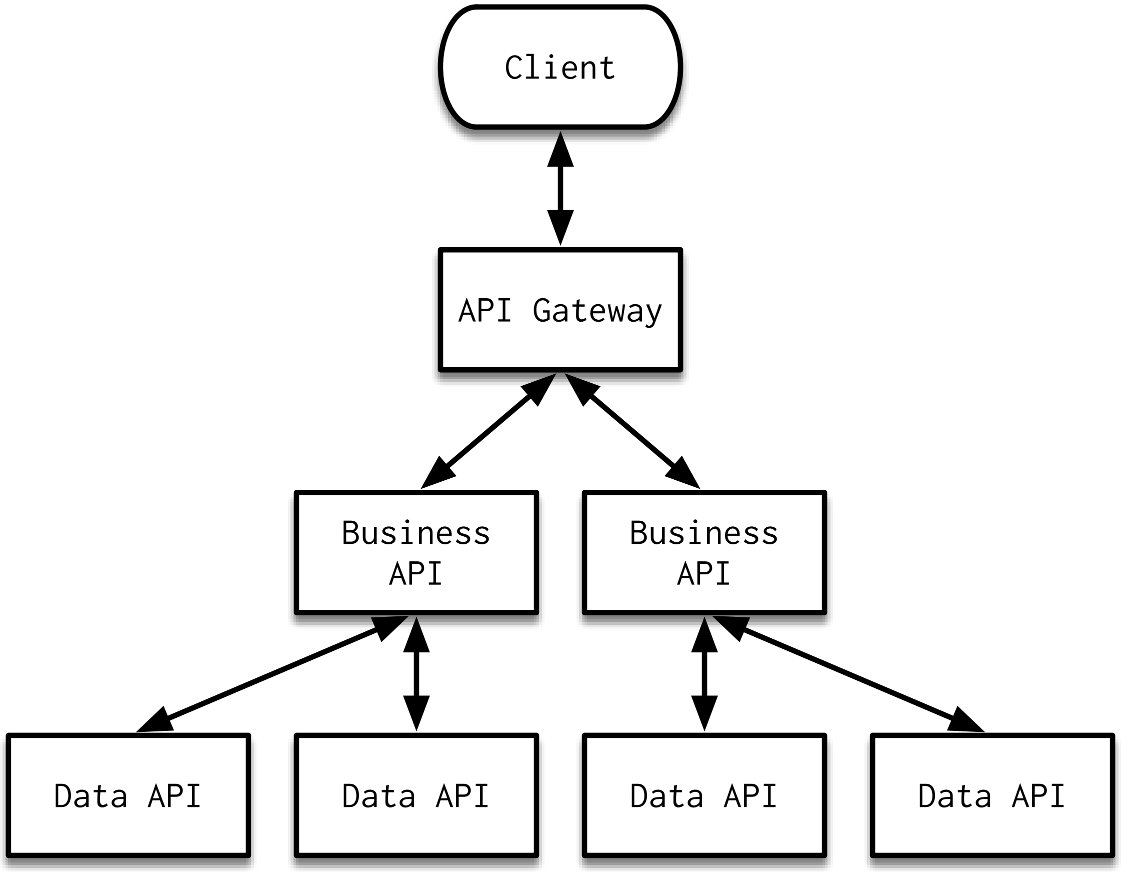
Decoupling with `resolve()`

It's becoming more…

multiple business API...



with a Gateway...



Client

API Gateway

Business API

Business API

Data API

Data API

Data API

Data API

Gateway

Single entry point

We use [Kong](4)

Take a break…

Problem arisen

Too many requests

Asynchronous it...

Push services

Websockets

HTTP/2

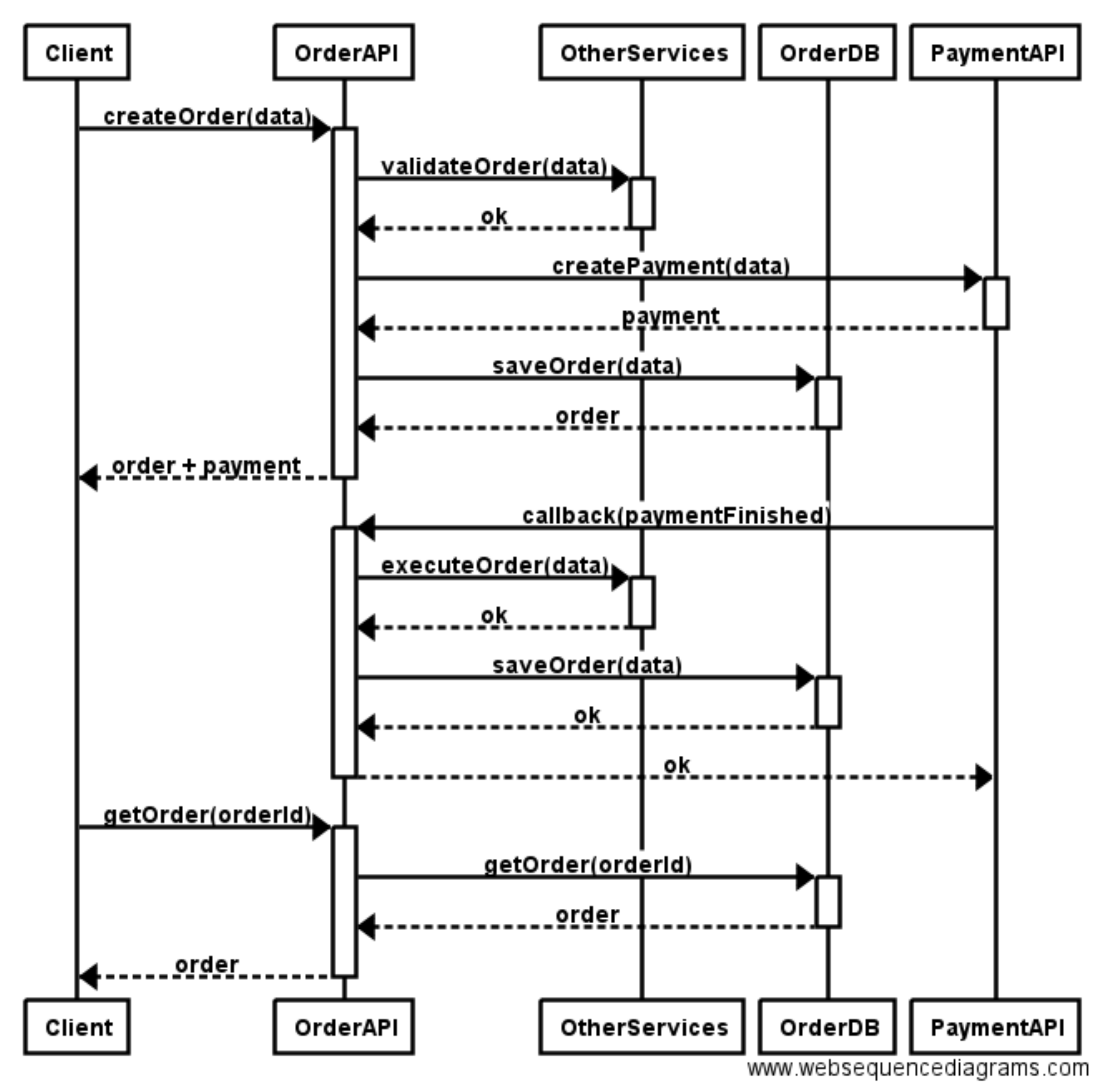
Problem arisen

Transaction processing

Race condition

Example

Order and payment


Client
OrderAPI
OtherServices
OrderDB
PaymentAPI
createOrder(data)
validateOrder(data)
ok
createPayment(data)
payment
saveOrder(data)
order
order + payment
callback(paymentFinished)
executeOrder(data)
ok
saveOrder(data)
ok
ok
getOrder(orderId)
getOrder(orderId)
order
order
Client
OrderAPI
OtherServices
OrderDB
PaymentAPI
www.websequencediagrams.com

Traditionally...

DB transaction and rollback

DB lock/revision

However...

We are distributed

...and sometimes asynchronous

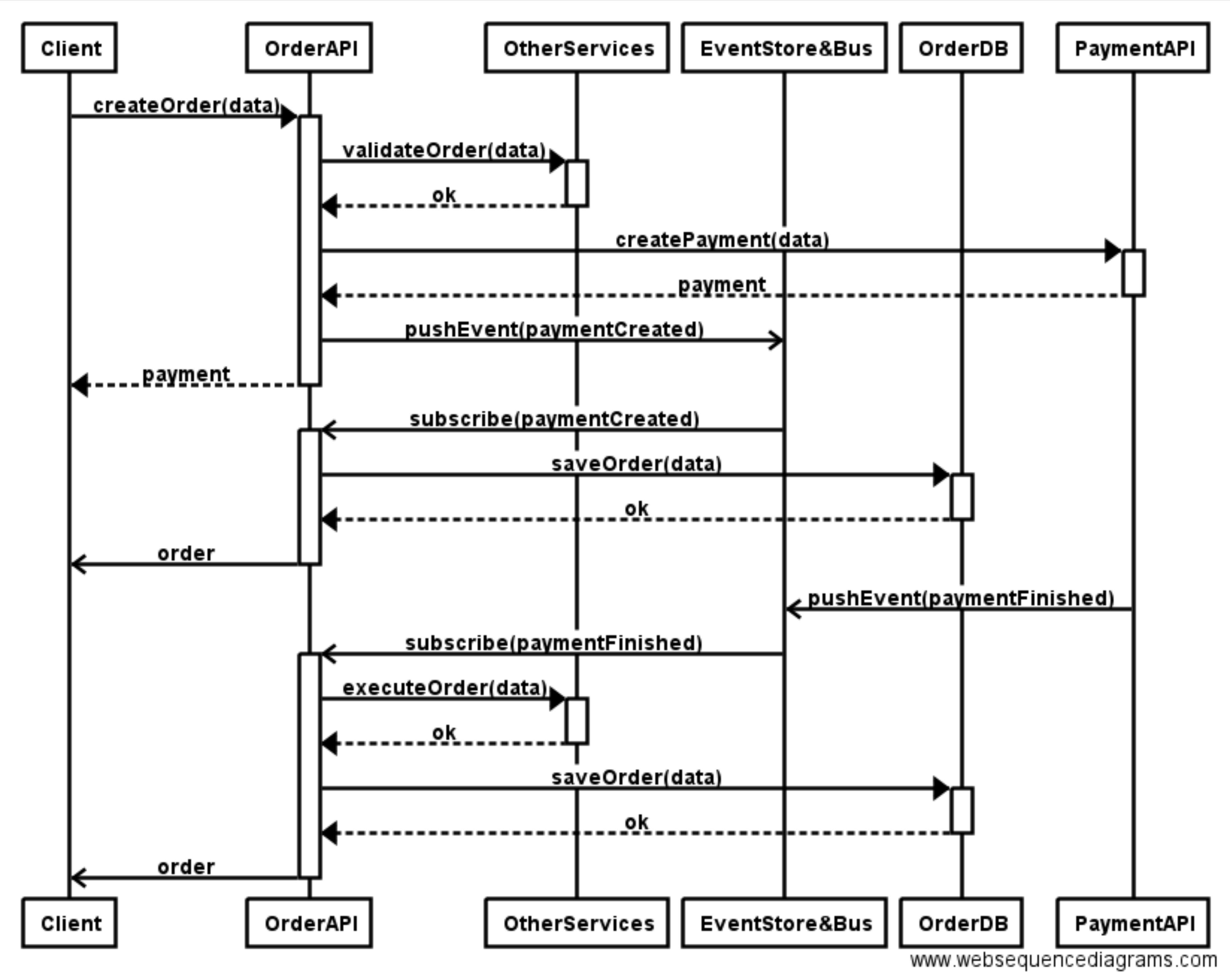
A solution…

Multiple steps to handle one request

An event/job queue

…and now you have…

CQRS + Event Sourcing


Client
OrderAPI
OtherServices
EventStore&Bus
OrderDB
PaymentAPI
createOrder(data)
validateOrder(data)
ok
createPayment(data)
payment
pushEvent(paymentCreated)
payment
subscribe(paymentCreated)
saveOrder(data)
ok
order
pushEvent(paymentFinished)
subscribe(paymentFinished)
executeOrder(data)
ok
saveOrder(data)
ok
order
Client
OrderAPI
OtherServices
EventStore&Bus
OrderDB
PaymentAPI
www.websequencediagrams.com

CQRS (usually with Event Sourcing)

“Command Query Responsibility Segregation”

“Command” for create/update/delete

“Query” for read/query

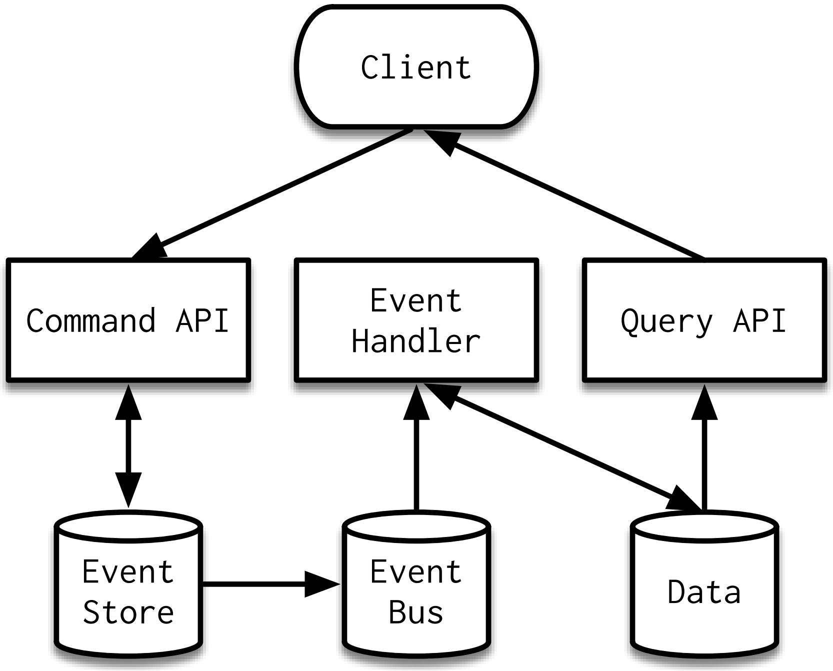
Event Sourcing

Persist each business entity as a sequence of events

Reply events to get the current state

“Eventually consistent”

CQRS + Event Sourcing...



Looking forward…

"Why this and that…"

It solves our problems (for now)

We are always exploring

Thanks!

~~Questions?~~

References

- (1) http://jsonapi.org/
- (2) http://queues.io/
- (3) http://nsq.io/
- (4) https://getkong.org/

References

- https://en.wikipedia.org/wiki/Microservices
- http://microservices.io/patterns/index.html
- http://graphql.org/