Moving beyond request-reply: How smart APIs are different.
Software Development

Architecture and Design 2019 Q1 Graph

http://infoq.link/architecture-trends-2019

- Reactive Programming
  - Functional Programming
  - CQRS
  - Actor Model
  - "Serverless" (FaaS/BaaS/DBaaS/PaaS)
  - gRPC and HTTP/2
  - GraphQL
- Evolutionary Architecture
  - (Lightweight) workflow and decision automation platforms
- "Architect as technical leader"

- Event-Driven Architecture (and Event Sourcing)
  - Eventual Consistency
- Microservices
  - Domain-Driven Design
  - Behaviour-Driven Design
  - Test-Driven Design
  - REST

Innovators | Early Adopters | Early Majority | Late Majority
--- | --- | --- | ---
Blockchain and Distributed Ledgers | Service meshes (Envoy, Linkerd, Istio) | Correctly built distributed systems | Microservices
HTTP/3

InfoQ
Failure will happen. Accept it!

But keep it local! Be resilient.
There was an error while sending your boarding pass
Current situation

Me → Web-UI → Check-in
Current situation

Me → Web-UI → Check-in → Output Mgmt → Barcode Generator
Current situation

Me -> Web-UI

Web-UI -> Check-in

Check-in -> Output Mgmt

Output Mgmt -> Barcode Generator

Barcode Generator on fire
Current situation – the good part

Me → Web-UI → Check → Output Mgmt → Barcode Generator

Circuit breaker
Internal Server Error - Read

The server encountered an internal error or misconfiguration and was unable to complete your request.

Reference #3.1d079ccc.1519892932.9c55d68
Current situation – the bad part
Current situation - the bad part
Current situation — the bad part
We are having some technical difficulties at the moment.

Please log on again via www.easyjet.com

If that doesn’t work, please try again in five minutes.

We do actively monitor our site and will be working to resolve the issue, so there’s no need to call.
We are having some technical difficulties and cannot present you your boarding pass right away.

But we do actively retry ourselves, so lean back, relax and we will send it on time.
Possible situation — much better!
Possible situation – much better!

Me ➔ Web-UI ➔ Check-in ➔ Output Mgmt ➔ Barcode Generator ➔ Stateful Retry

Check-in

Web-UI
Warning: Contains Opinion
Bernd Ruecker
Co-founder and Chief Technologist of Camunda

http://berndruecker.io/
mail@berndruecker.io
@berndruecker

Berlin, Germany
You can use a workflow engine (=durable state machine)!

Diagram:
- Boarding pass requested
- Generate Barcode
- Send boarding pass via email
- Boarding pass sent
- Check-In
  - REST
  - Barcode
- Stateful retry

Notes:
- You can use a workflow engine (=durable state machine)
- Boarding pass requested
- Generate Barcode
- Send boarding pass via email
- Boarding pass sent
- Check-In
  - REST
  - Barcode
- Stateful retry
Want to see code?

https://github.com/berndruecker/flowing-retail
Client

has to implement

Retry

Service Provider

has to implement

Idempotency
Your transaction is in progress

It may take up to 60 seconds to complete. Please do not close this window or select the "Back" button on your browser.
Your transaction is in progress

It may take up to 60 seconds to complete. Please do not close this window or select the "Back" button on your browser.
Don't worry, it will happen safely – even if you lose connection.
Feel free to reload this page any time!
Requirement: Idempotency of services!
Requirement: Idempotency of services!

Photo by Chr.Späth, available under Public Domain.
Make every service idempotent!

Generally: create IDs as soon as possible.

Charge Credit Card
- cardNumber
- amount

Charge Credit Card
- cardNumber
- amount
- transactionId

Not idempotent

Idempotent
Distributed systems introduce complexity you have to tackle!
Distributed systems
It is impossible to differentiate certain failure scenarios.

Independent of communication style!
Distributed systems introduce complexity you have to tackle!
Distributed systems introduce complexity you have to tackle!
Being able to implement long running services is essential for smart APIs (on a technical level)
Example

Booking → Retrieve Payment → Payment ↔ Credit Card
Example

Booking → Retrieve Payment → Payment → Credit Card → Rejected
If the credit card was rejected, the customer can provide new details.
If the credit card was rejected, the customer can provide new details.

A few smart god services tell anemic CRUD services what to do.

Sam Newmann
Who is responsible to deal with problems?

If the credit card was rejected, the customer can provide new details.
Long running services

Booking → Retrieve Payment
Payment received → Payment
Payment failed → Payment

Payment → Credit Card
Rejected → Credit Card

Smart endpoints are potentially long-running
Being able to implement long running services is essential for smart APIs (on a business level)
Long running services require async communication
Synchronous communication
Synchronous communication is the crystal meth of distributed programming

Todd Montgomery and Martin Thompson in “How did we end up here” at GoTo Chicago 2015
Asynchronous communication

You need to monitor timeouts

Check-in

Web-UI

Me

Output Mgmt

Barcode Generator

You need to monitor timeouts
Workflow...

1. Send generate barcode command
2. Wait for barcode
3. Resend generate barcode command every hour
4. Send boarding pass
5. Wait for mail confirmation
6. Resend boarding pass every hour
7. Check-In successful
Workflow...

1. Send generate barcode command
2. Wait for barcode
3. Send boarding pass
4. Wait for mail confirmation
5. Check-In successful
6. Resend boarding pass

Additional steps:
- Resend generate barcode command (every hour)
- Call customer to apologize and ask to visit check-in counter (4 hours before flight)
- Manually cleared
Being able to implement long running services makes it easy to get async
Can your company leverage your hipster architecture?

You need to change business processes and customer experience!
Example

Ticket and reservation

How would you like to receive your ticket?

Digital ticket
Save it as a PDF or load it to the DB Navigator app

Order and receive
5,90 EUR
Printed ticket

Ticket for another person
You have to book for this service

Our tip: reserve a seat now.
Reservation of 1 seat 4,50 EUR

Questions concerning reservations?

Onward journey
Mo, 09.09.2023
18:37 Berlin HBF (arr)
10:42 München HBF

Ticket class
1 Adult
City Ticket
30 EUR

Total
25,50 EUR

Die Bahn
OnlineTicket

Fahrkarte ICE
98 09 2023 18 37 10 42
Honigstein (Reitdiebach) (Eisenbahn-Posten)
Bremen HBF (Hauptbahnhof)
Berlin HBF (Central Station)

Preis
Ticket, class: 1 Adult
City Ticket: 30 EUR

Zahlungsbereitschaft und Preis

BGB 123 456 789
123 456 789 123

Date: 22.09.2023
Example

[Diagram showing the process of booking a seat reservation, payment, and ticket generation.]
Example

Check & book

Confirmation

Ticket and reservation

How would you like to receive your ticket?

Digital ticket
Save it as a PDF or load it in the DB Navigator app

Order and receive
5,90 EUR

Total
75,90 EUR

You aren't travelling?

Ticket for another person

Our tip: reserve a seat now.

Reservation of 1 seat

Questions concerning reservation?

Sync
Example

Ticket and reservation

How would you like to receive your ticket?

- Digital ticket: Save it as a PDF or load it in the DB Navigator app

Please note:
- Only valid for the passenger stated (and accompanying passengers if applicable), not transferable. You must present your ID to the ticket inspector on the train.

Order and receive
- 5,50 EUR
- 75,90 EUR

You aren’t travelling?

- Ticket for another person

Our tip: reserve a seat now.

- Reservation of 1 seat: 4,50 EUR

Hinweis

Sehr geehrte/r Kunde/r, sehr geehrter Kundin,

Bei der Verarbeitung Ihrer Anmeldung ist ein Fehler aufgetreten. Unverzüglich haben Sie nationalen den gleichen Button angeklickt oder eine Funktion aus dem Fenster heraus starten aufgezweigt. Bitte beachten Sie, dass Sie unser Leitungsrat aus einem virtuellen Blat Fenster heraus bedienen sollten.

Wö unschließlich aus für die Unannehmlichkeiten.

Deutsche Bahn AG
Weaknesses

REST → Booking → Seat Reservation
Booking → Payment
Booking → Ticket Generation
Weaknesses: Latency creep

- Payment: 1150 + x ms
- Seat Reservation: 600 ms
- Payment: 300 ms
- Ticket Generation: 250 ms

Diagram:
Weaknesses: Availability erosion

- Booking: 96% uptime
- Seat Reservation: 99% uptime
- Payment: 99% uptime
- Ticket Generation: 99% uptime
And it is even hard to implement
And it is even hard to implement
Typical pattern

Simulate synchronicity by waiting (callback or polling)
Redesign your business process accordingly!

Sync in happy case

Or some interface to poll for status

Async response
Redesign your business process accordingly!
Your business processes need to be more reactive!

https://www.reactivemanifesto.org/
Yeah!
Let’s go reactive.
We were suffering from Pinball machine Architecture
“What the hell just happened?”
Example: order fulfillment via dash button

Photo by 0xF2, available under Creative Commons BY-ND 2.0 license. https://www.flickr.com/photos/0xf2/29873149904/
Three steps...

1. Pay item
2. Fetch item
3. Ship item

@berndruecker
(Micro-)services

- Checkout
- Payment
- Inventory
- Shipment
Event-driven architecture

Checkout

Payment

Inventory

Shipment

Order Placed
Payment Received
Goods Fetched

Notification
Peer-to-peer event chains

- Order placed
- Payment received
- Goods fetched
- Goods shipped
- Checkout
- Inventory
- Shipment

Events:
- Pay item
- Fetch item
- Ship item

@berndruecker
Peer-to-peer event chains

Order placed

Checkout

Payment

Goods fetched

Goods shipped

Payment received

Inventory

Shipment

@berndruecker
The danger is that it’s very easy to make nicely decoupled systems with event notification, without realizing that you’re losing sight of that larger-scale flow, and thus set yourself up for trouble in future years.

https://martinfowler.com/articles/201701-event-driven.html
The danger is that it’s very easy to make nicely decoupled systems with event notification, without realizing that you’re losing sight of that larger-scale flow, and thus set yourself up for trouble in future years.

https://martinfowler.com/articles/201701-event-driven.html
The danger is that it’s very easy to make nicely decoupled systems with event notification, without realizing that you’re losing sight of that larger-scale flow, and thus set yourself up for trouble in future years.

https://martinfowler.com/articles/201701-event-driven.html
Peer-to-peer event chains

- **Order placed**
  - **Checkout**
  - **Payment**
    - **Goods fetched**
    - **Inventory**
    - **Goods shipped**
    - **Shipment**
  - **Payment received**

- **Fetch the goods before the payment**
Peer-to-peer event chains

Order placed → Checkout

Payment received → Inventory

Goods fetched → Goods shipped

Payment received → Shipment

Fetch the goods before the payment
What we wanted

vs. what we got

Photo by Lijian Zhang, available under Creative Commons SA 2.0 License and P..19 / CC BY-SA 4.0
Extract the end-to-end responsibility
Events & Commands

Order placed

Checkout

Order

Retrieve payment

Payment received

Payment

Shipment

Inventory

Event

Fact, happened in the past, immutable

Command

Intend, Want s.th. to happen
It is not about the protocol!

It can still be messaging!
It is about where to decide about the coupling!

Order decides

- to listen to the event
- to issue the command
Extract Orchestration logic
Workflows live inside service boundaries
Your services or applications

Your IT architecture

Choreography

Monolith

or applications

Process Monitoring

Chaos

Monolith

or applications

Process Monitoring

Chaos
Balance choreography and orchestration

Your IT architecture

Your services or applications

Process Monitoring

Choreography

Orchestration

Monolith

Chaos
Distributed systems are complex. At-least-once, retries and idempotency are here to stay. Embrace async!

- Long-running services make your life easier and your API smarter.
- Change business processes and customer experience accordingly
- Use commands + events = balance choreography and orchestration
Contact:  mail@berndruecker.io  @berndruecker

Slides:  https://berndruecker.io

Blog:  https://medium.com/berndruecker

Code:  https://github.com/berndruecker

https://www.infoworld.com/article/3254777/application-development/3-common-pitfalls-of-microservices-integrationand-how-to-avoid-them.html

https://www.infoq.com/articles/events-workflow-automation

https://thenewstack.io/5-workflow-automation-use-cases-you-might-not-have-considered/