WIND HIM UP

Mit SAGA service-übergreifende Transaktionen verwalten

sidion

ÜBER MICH

Thomas Müller

sidion GmbH

Senior Software-Developer

thomas.mueller@sidion.de

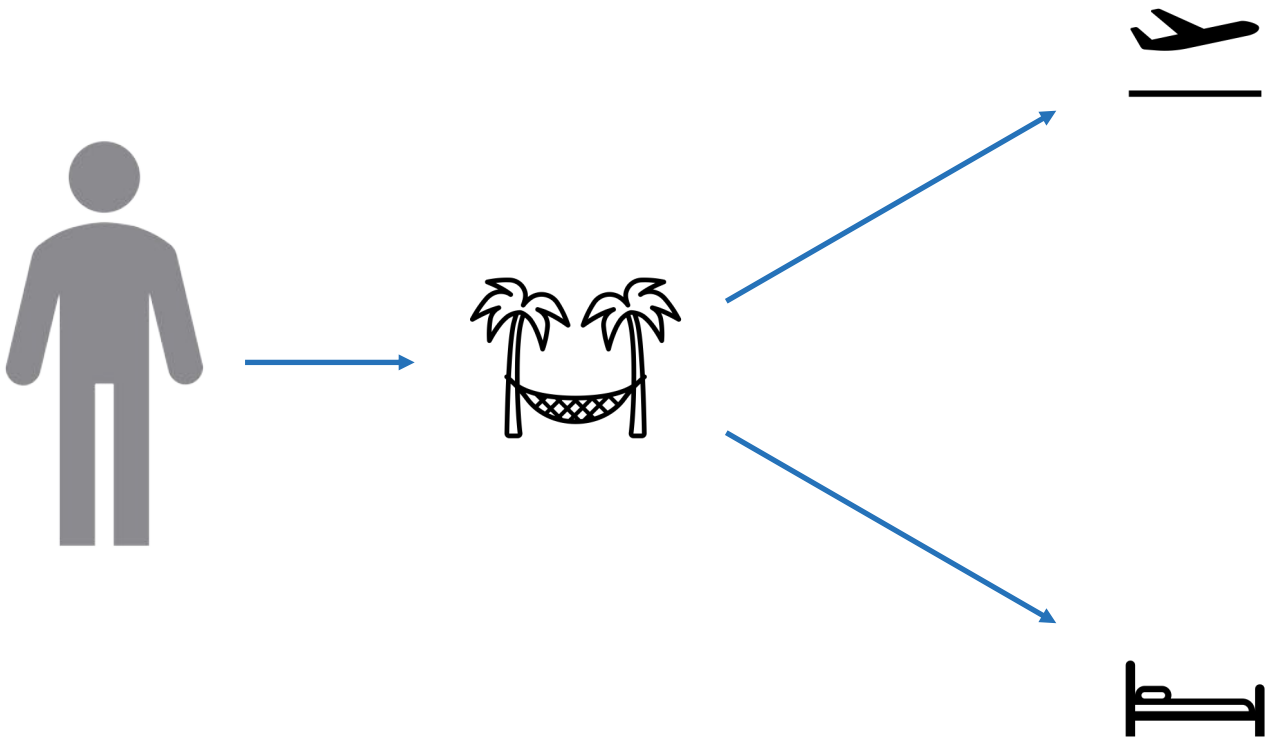
@zaroselectro

AGENDA

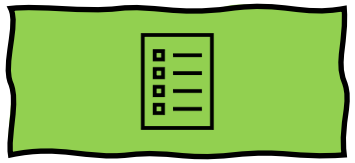
1. Szenario / Problemstellung
2. Distributet Transactions
3. Saga-Pattern
4. Choreography-based Saga
5. Orchestration-based Saga
6. Edge Cases
7. Coding

SZENARIO / PROBLEMSTELLUNG

SZENARIO



SERVICES

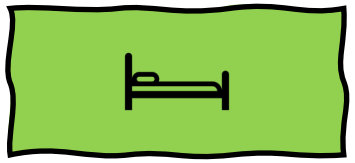


BookingService

- REST-API für Frontend
- Koordinator der Buchung

FlightBooking Service

- handelt Flugbuchung

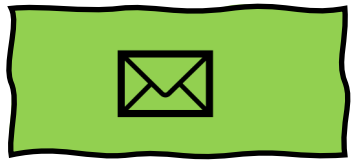


HotelBooking Service

- handelt Hotelbuchung

PaymentService

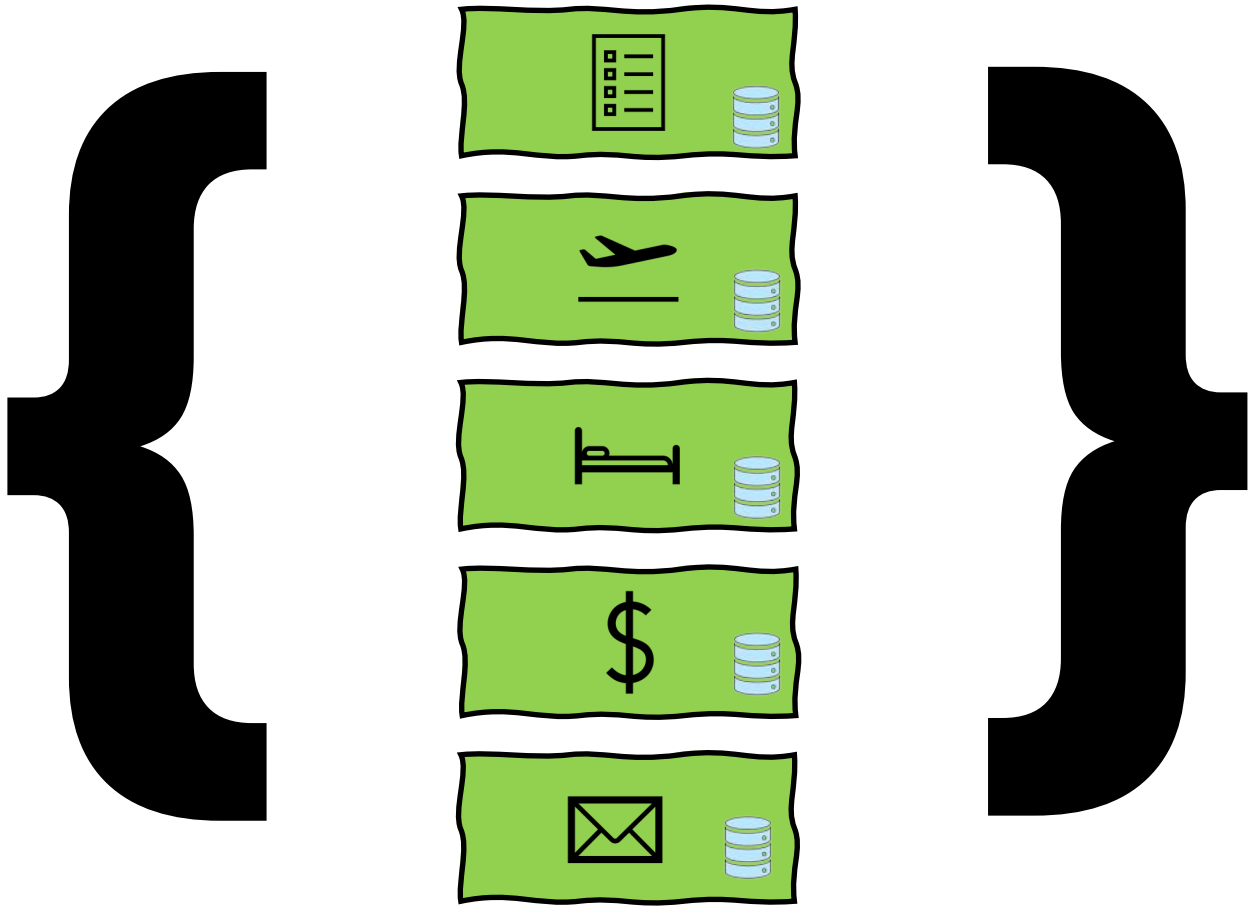
- Validiert Kreditkarte
- handelt Zahlung



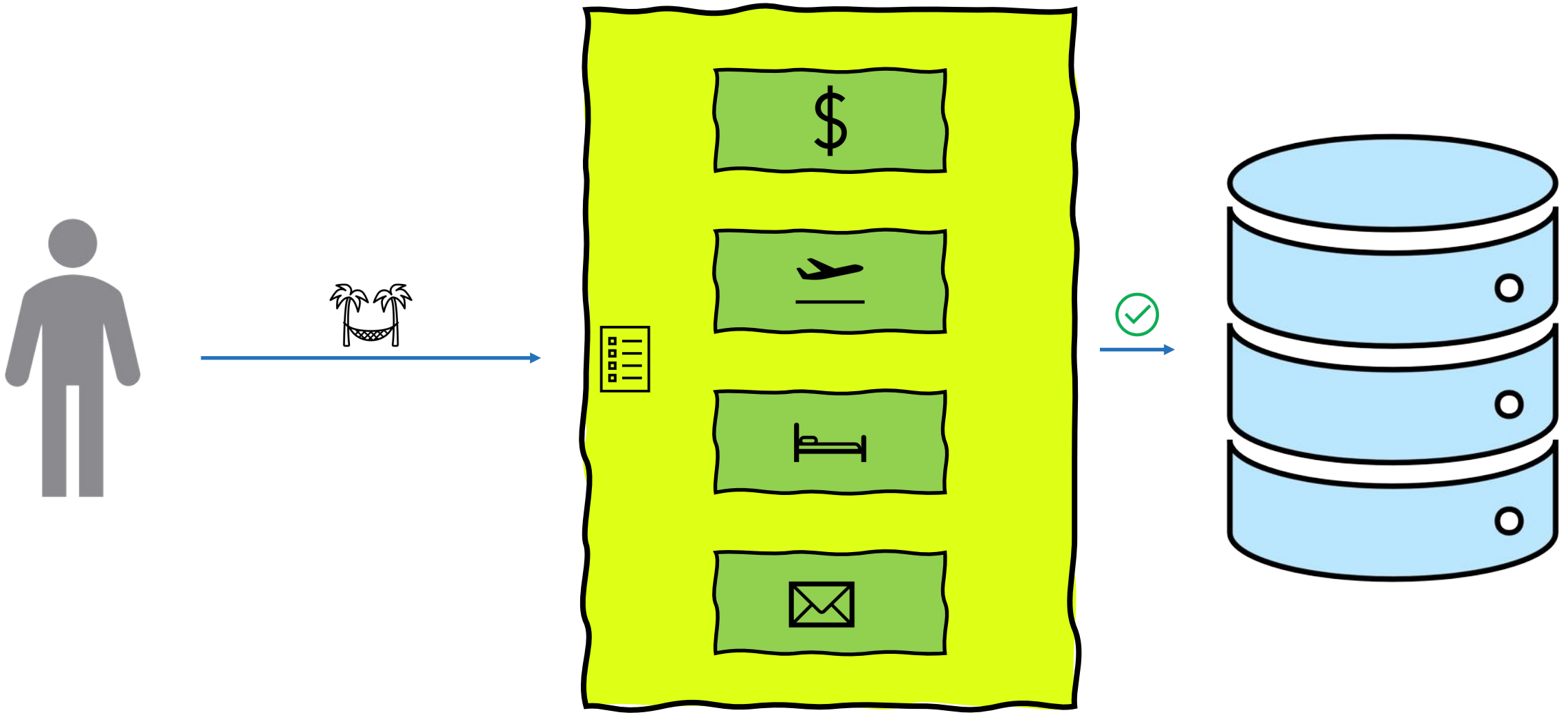
Confirmation Service

- Bestätigungs-Email

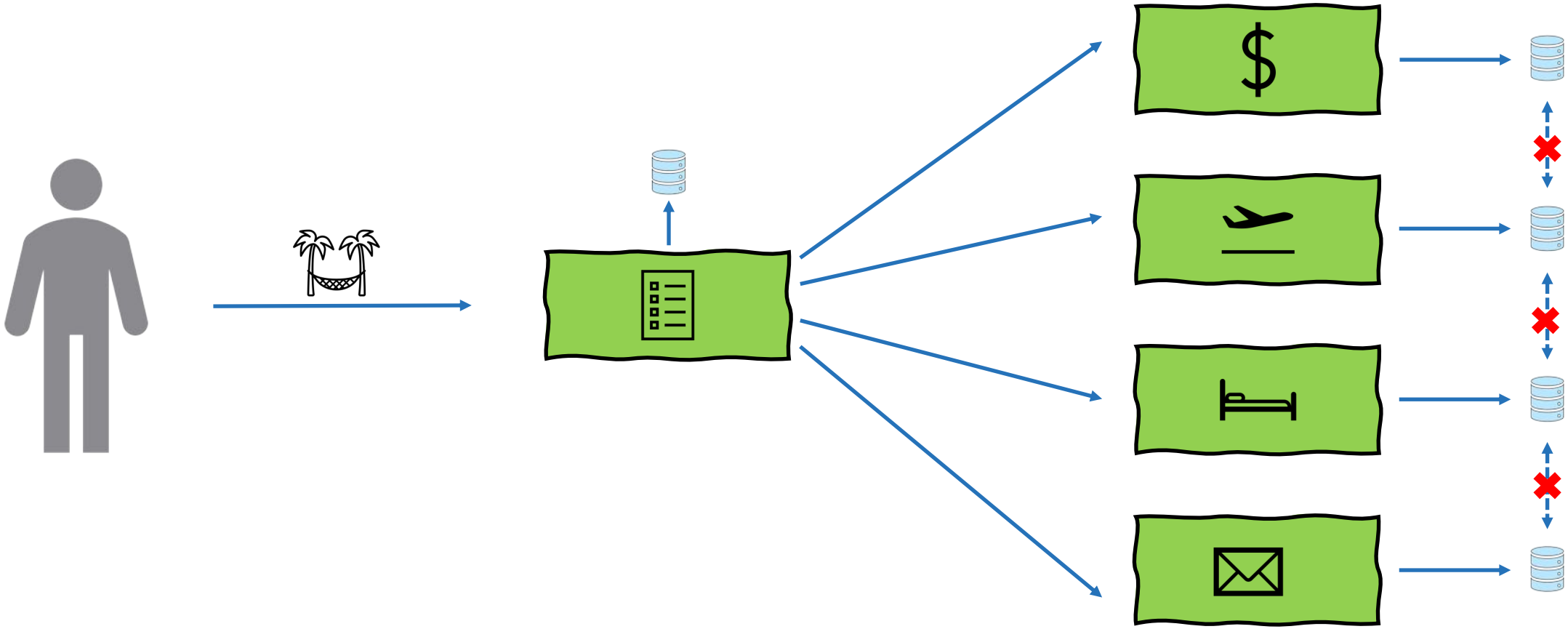
TRANSAKTION



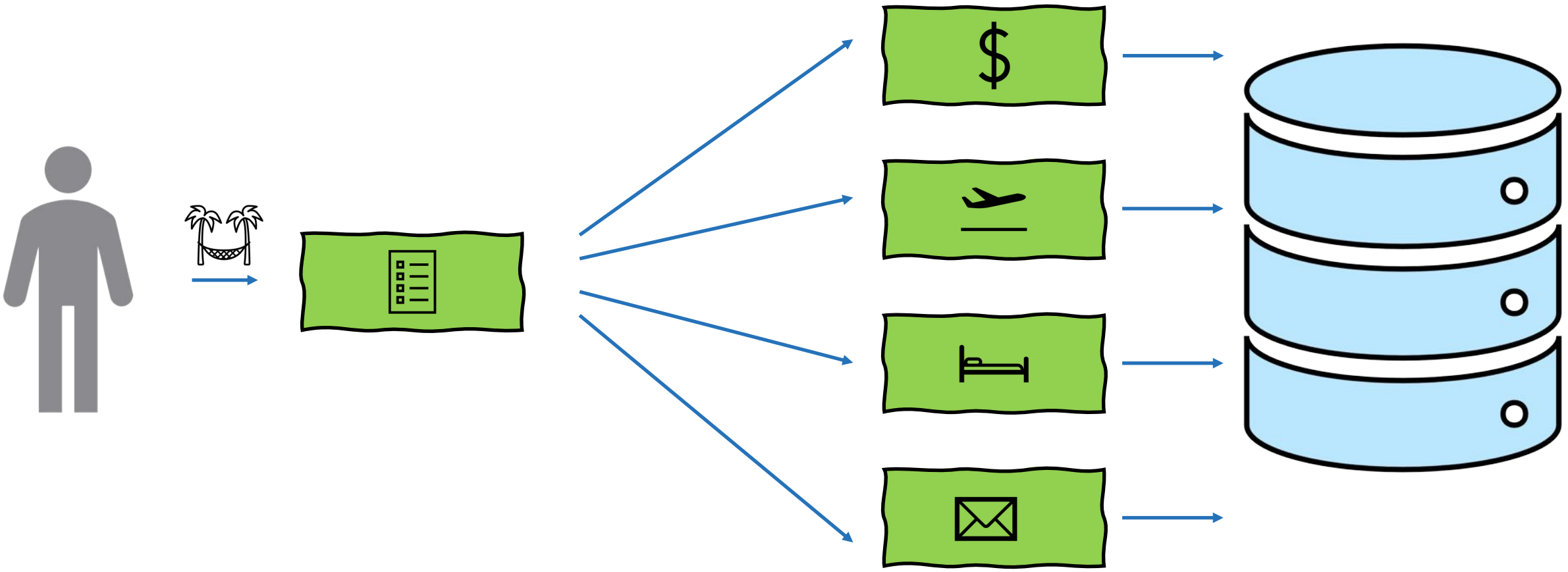
MONOLITH



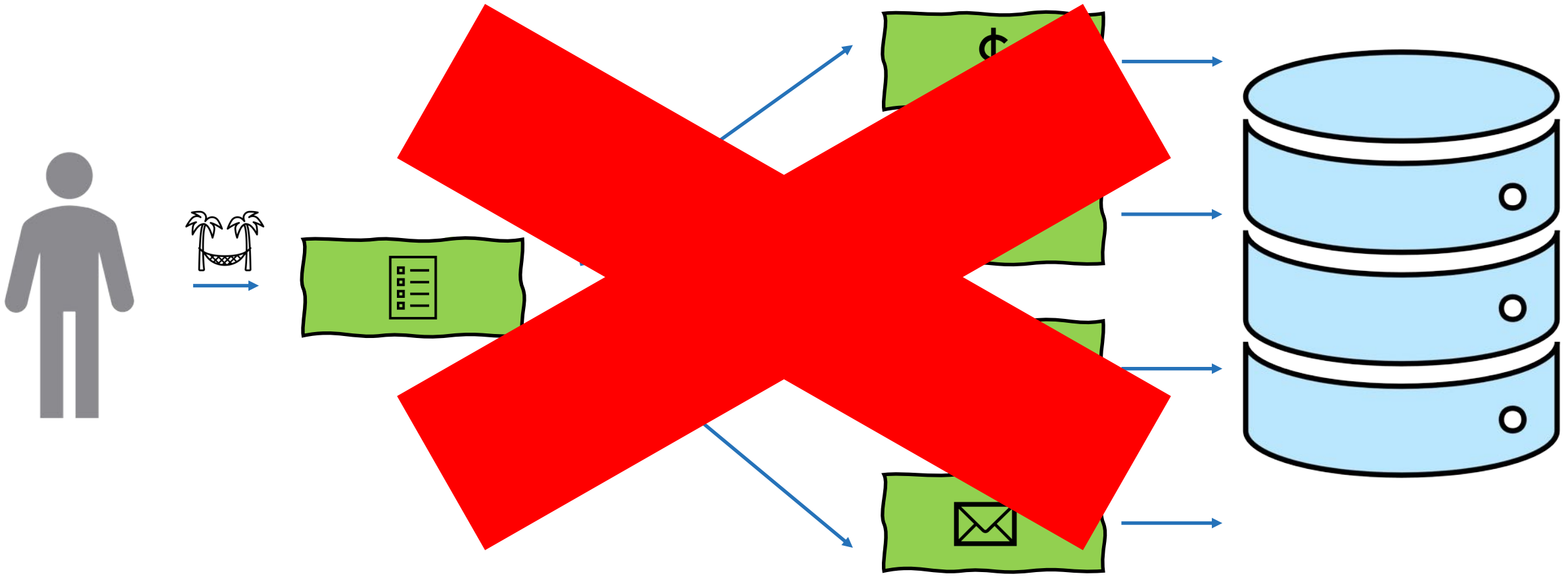
MICROSERVICE-ANSATZ



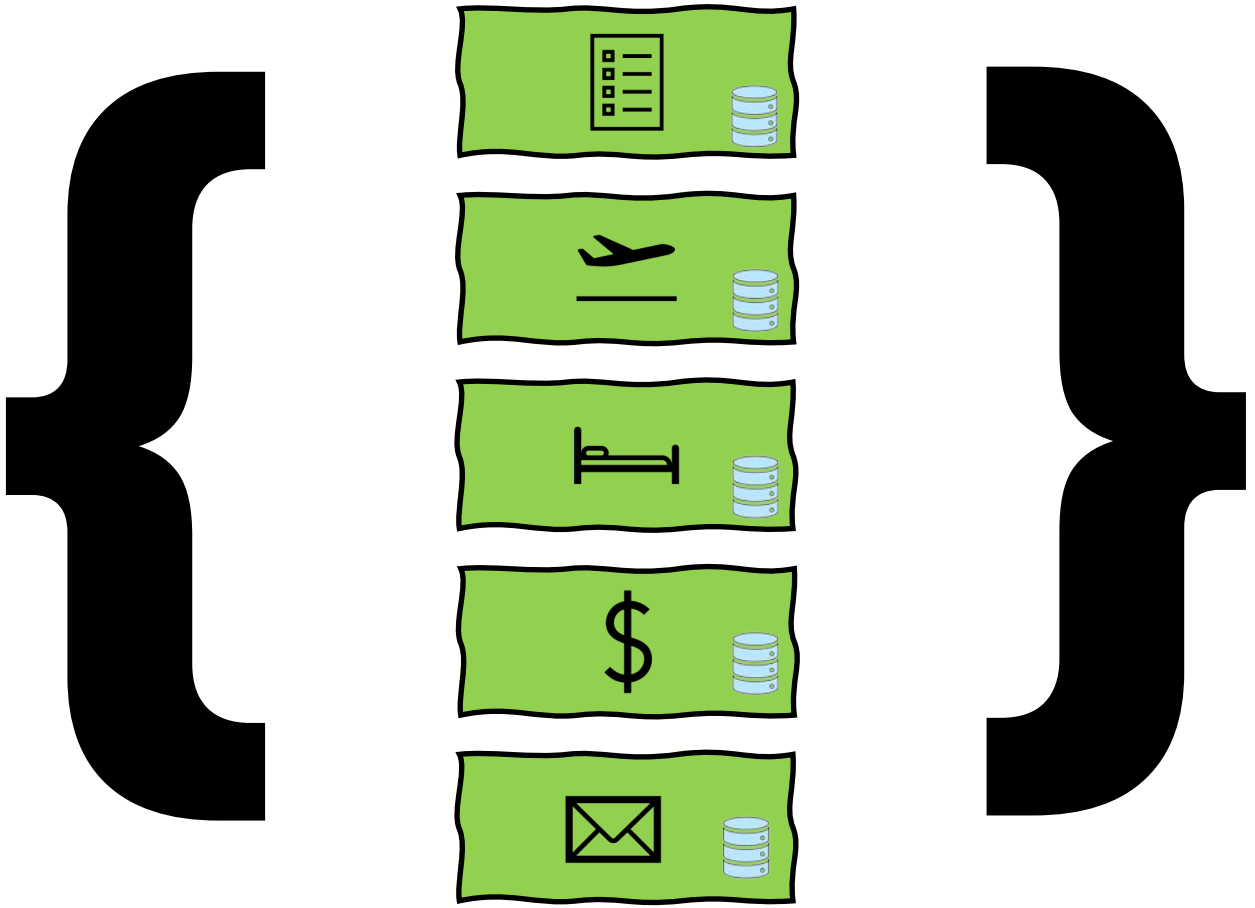
MICROSERVICE-ANSATZ



MICROSERVICE-ANSATZ



TRANSAKTION



A – C – I - D

A tomicity

C onsistency

I solation

D urability

A – C – I - D

A tomicity

C

I

D

Ausführung aller oder keiner Informationsteile einer Transaktion

A – C – I - D

A

C onsistency

I

D

Transaktionen erzeugen einen gültigen Zustand oder fallen in den alten Zustand zurück

A – C – I - D

A

C

I solation

D

Transaktionen verschiedener Anwender oder Prozesse bleiben voneinander isoliert

A – C – I - D

A

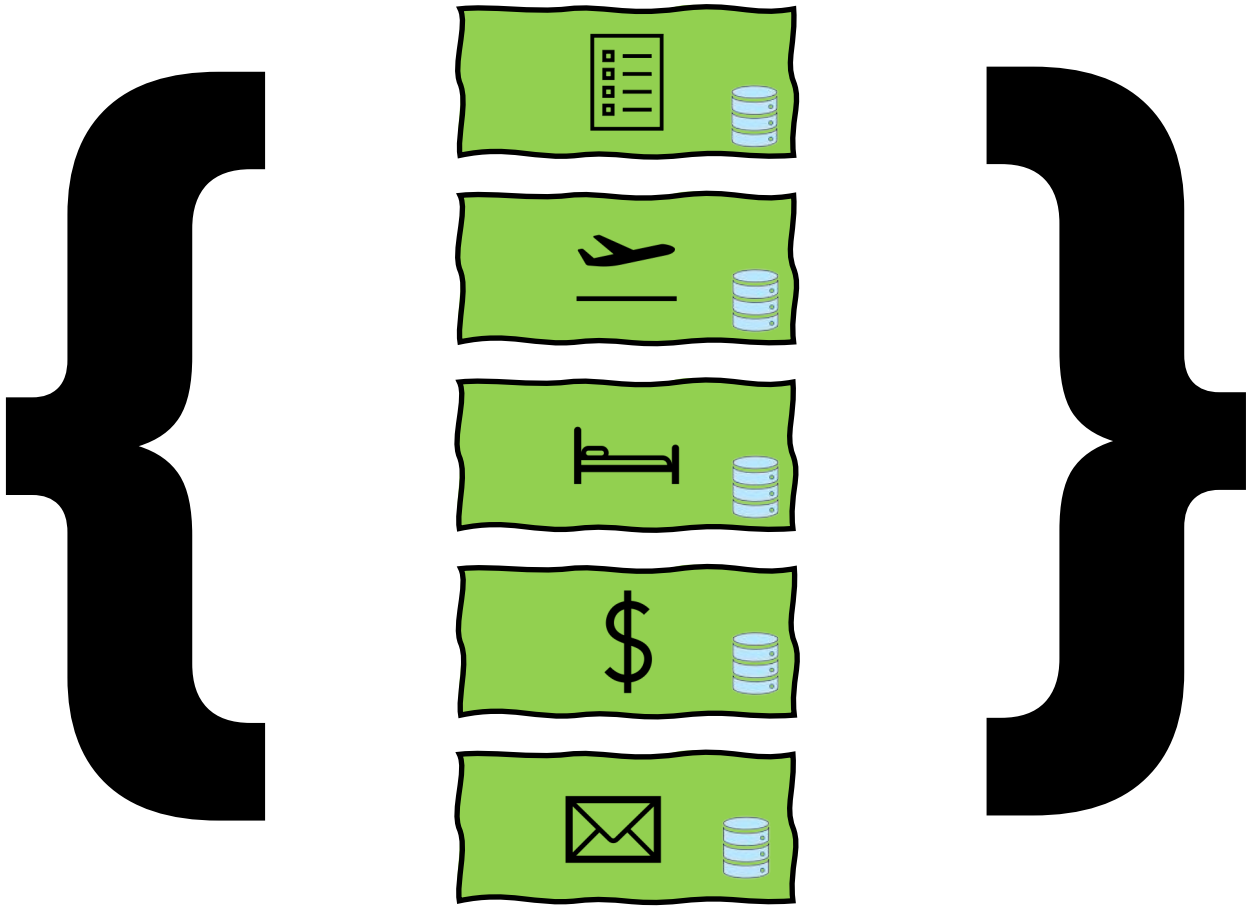
C

I

D urability

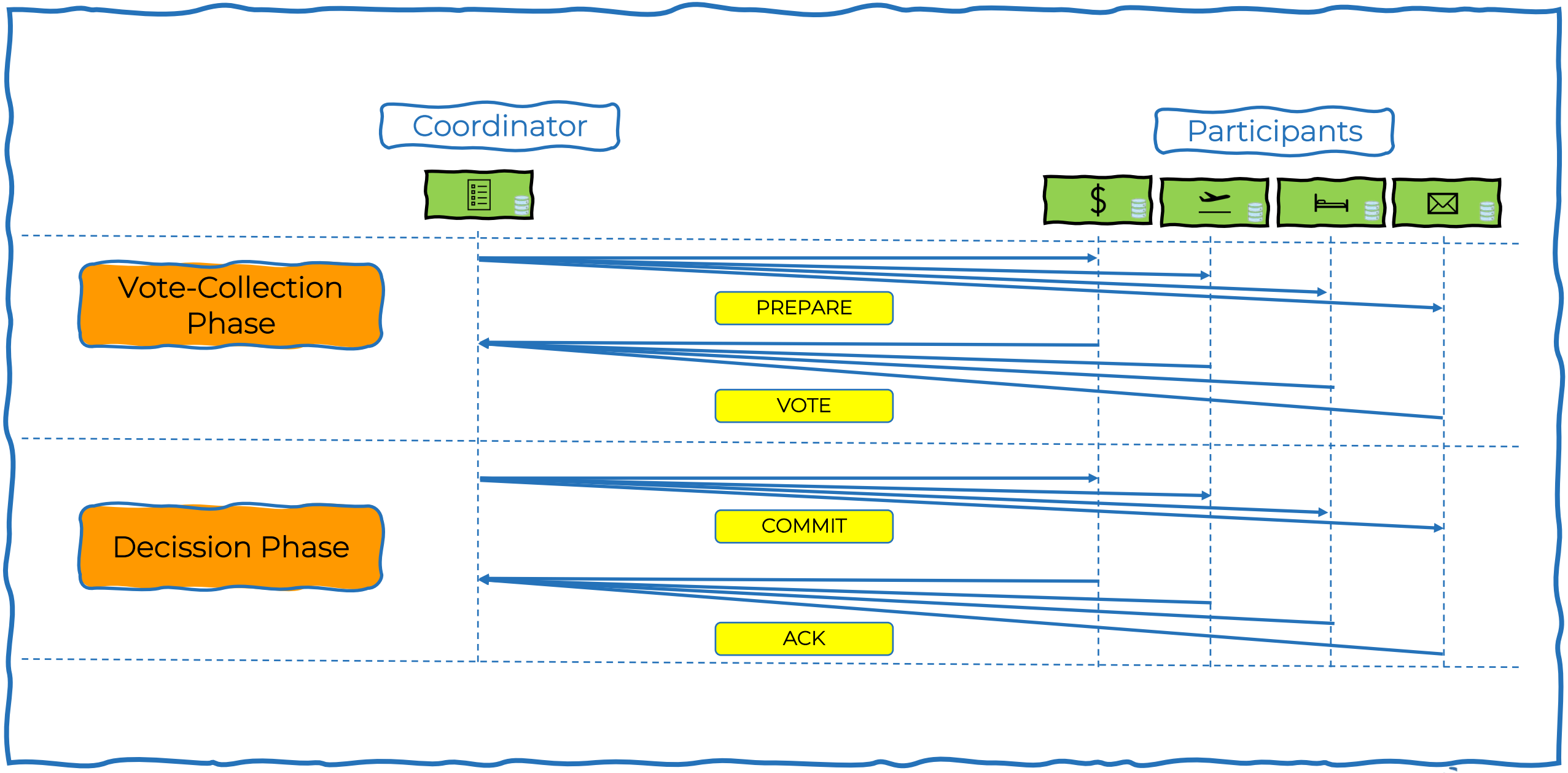
Nach einer erfolgreichen Transaktion bleiben die Daten dauerhaft gespeichert

TRANSAKTION



DISTRIBUTED TRANSACTIONS

2-PHASE-COMMIT (2PC)

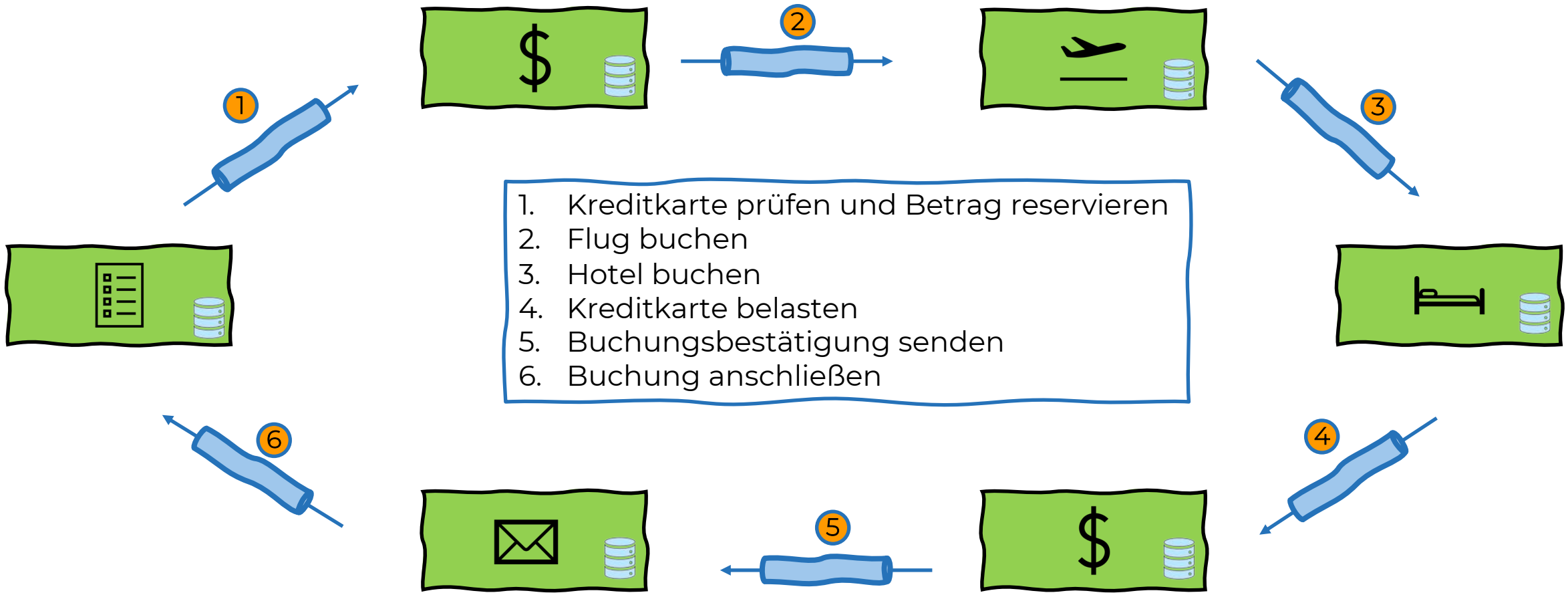


SAGA-PATTERN

SAGA PATTERN

- SAGA is a sequence of local transactions that are coordinate using messaging
- Each local transaction updates data in a single service
- On failure due to the violation of a business rule, it must execute compensation transactions to explicitly undo changes

SAGA PATTERN

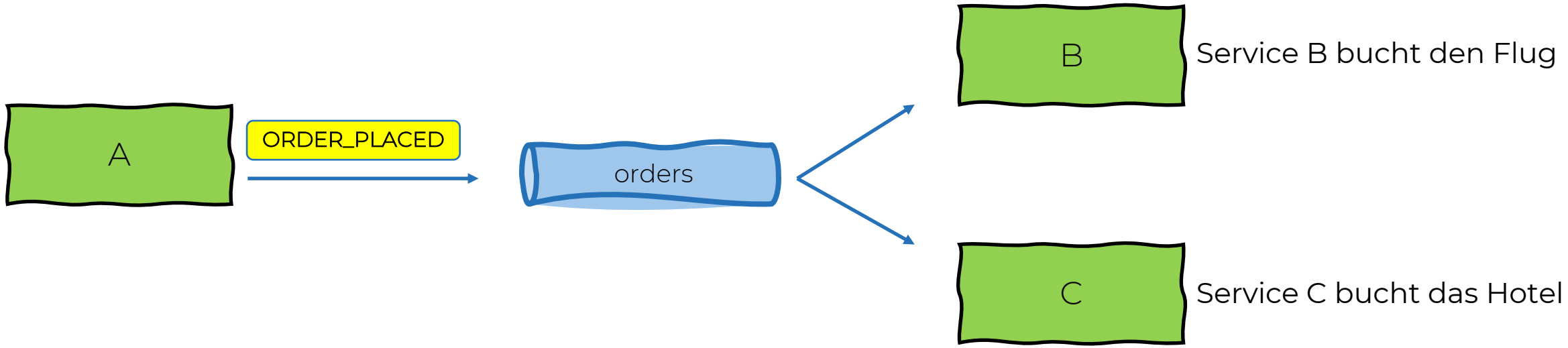


COMMAND-
VS.
EVENT-BASED

COMMAND- VS. EVENT-BASED

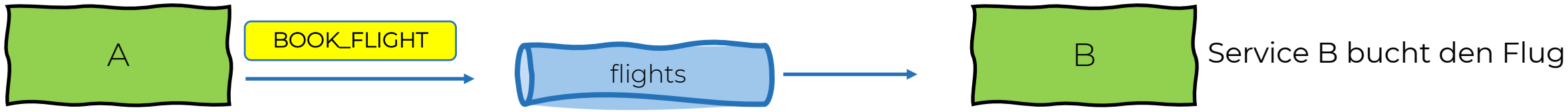
EVENT

- Fakt der bereits passiert ist
- Sender kann nicht kontrollieren was damit passiert
- Sender weiß nicht wer das Event aufnimmt



COMMAND- VS. EVENT-BASED

- COMMAND
  - Sender will das etwas passiert
  - Absicht, Aufforderung
  - kann nicht ignoriert werden
  - Empfänger weiß nicht wer das Kommando beauftragt hat

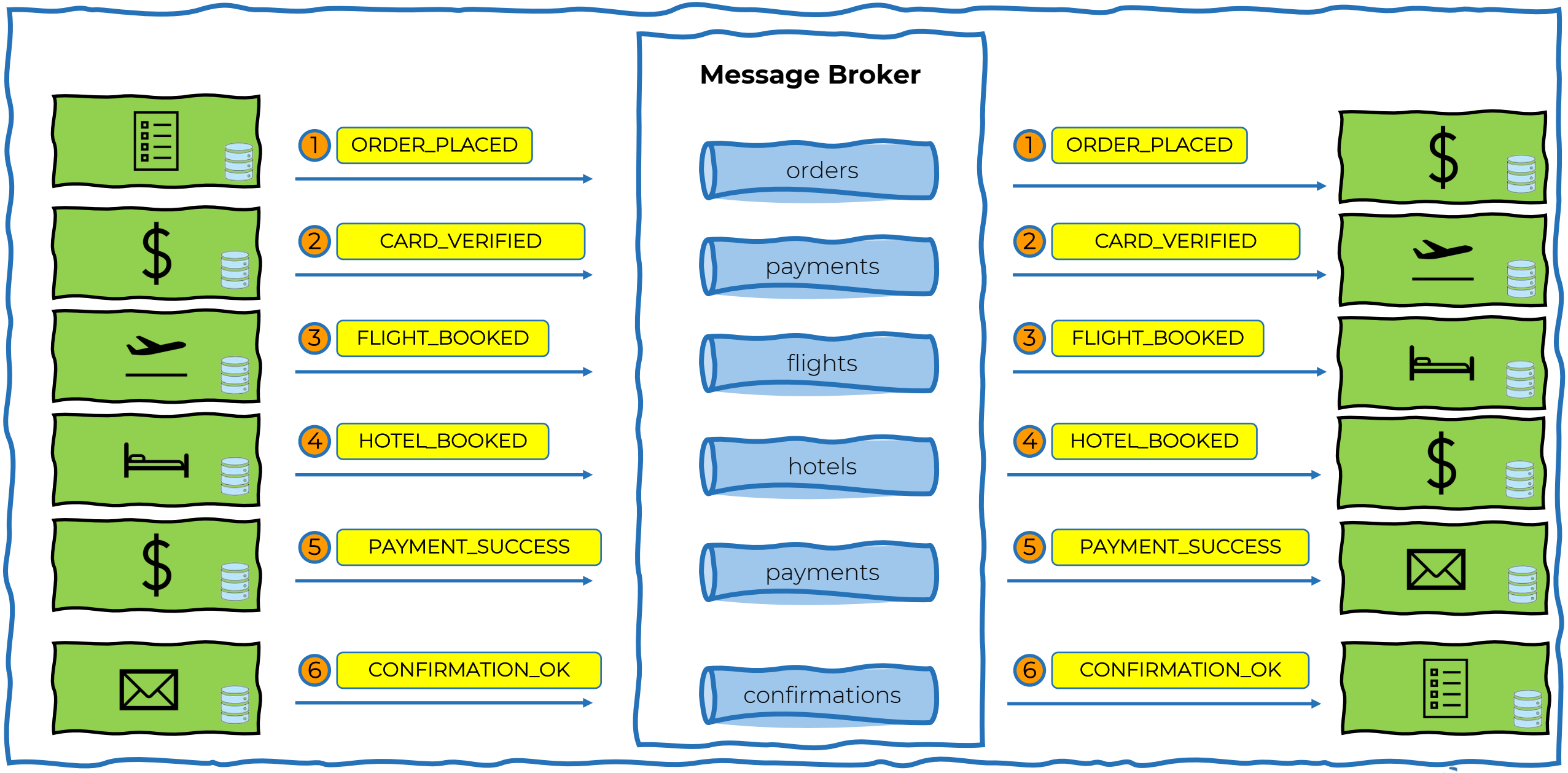


CHOREOGRAPHY BASED SAGA

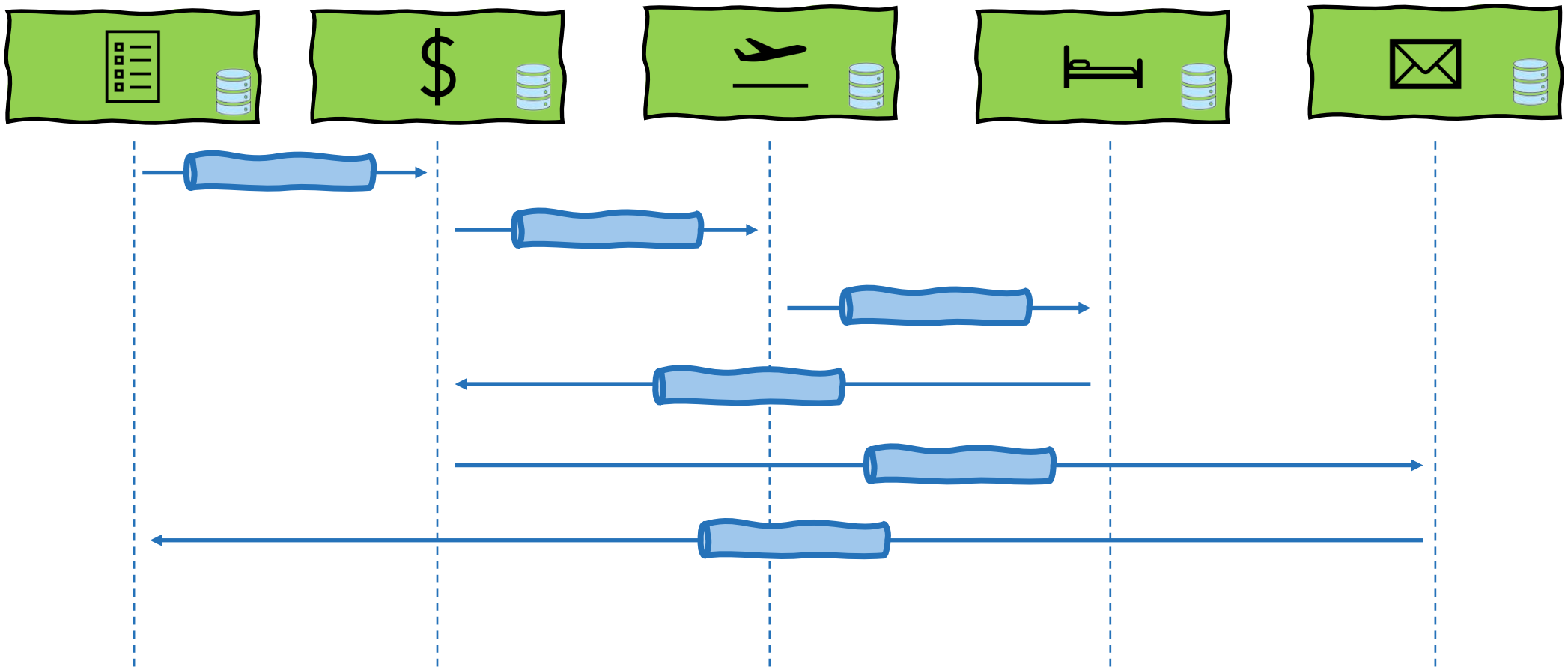
CHOREOGRAPHY-BASED SAGA

- einzelne Transaktionen die in einer bestimmten Reihenfolge ausgeführt werden
- jeder Teilnehmer der Saga weiß nach welchem vorherigen Schritt er dran ist
- Einsatz nur bei sehr wenige Teilnehmer einer Saga
- Event-Based-Communication

CHOREOGRAPHY-BASED SAGA



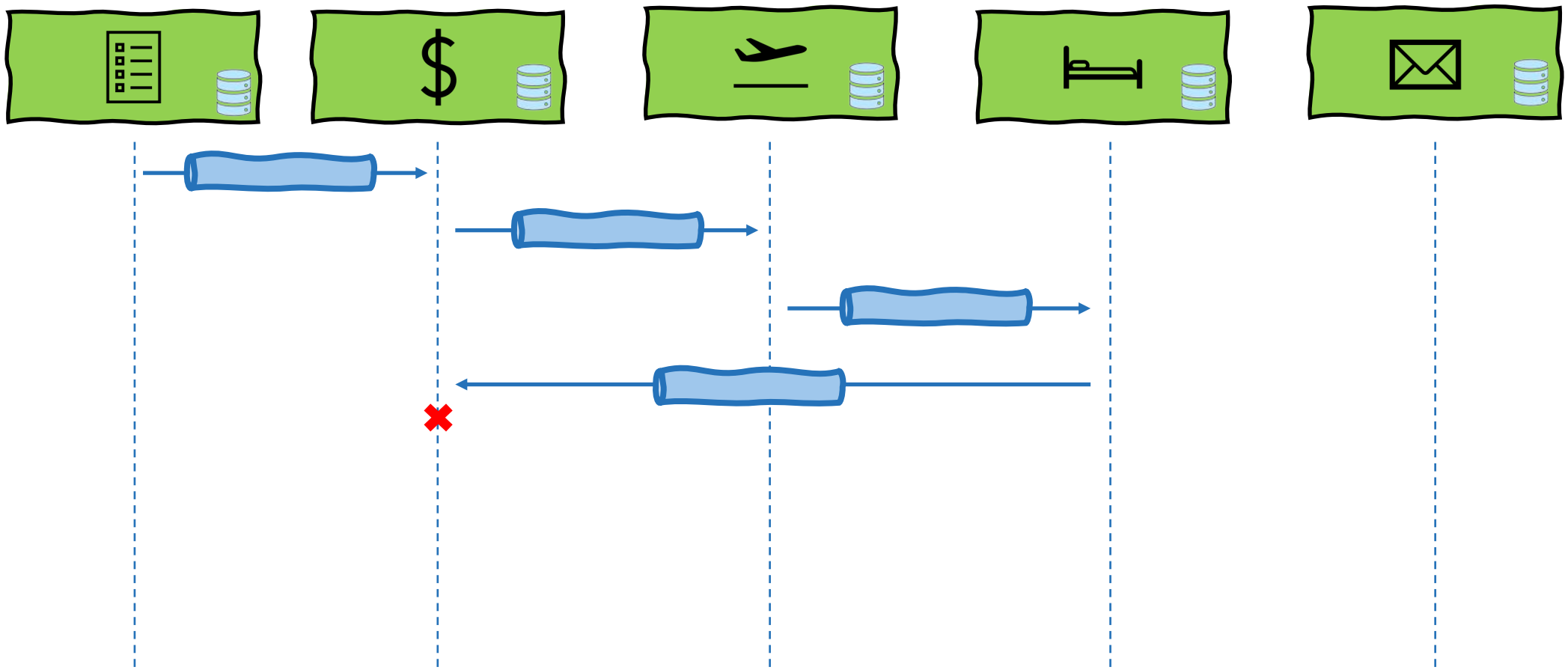
CHOREOGRAPHY-BASED SAGA



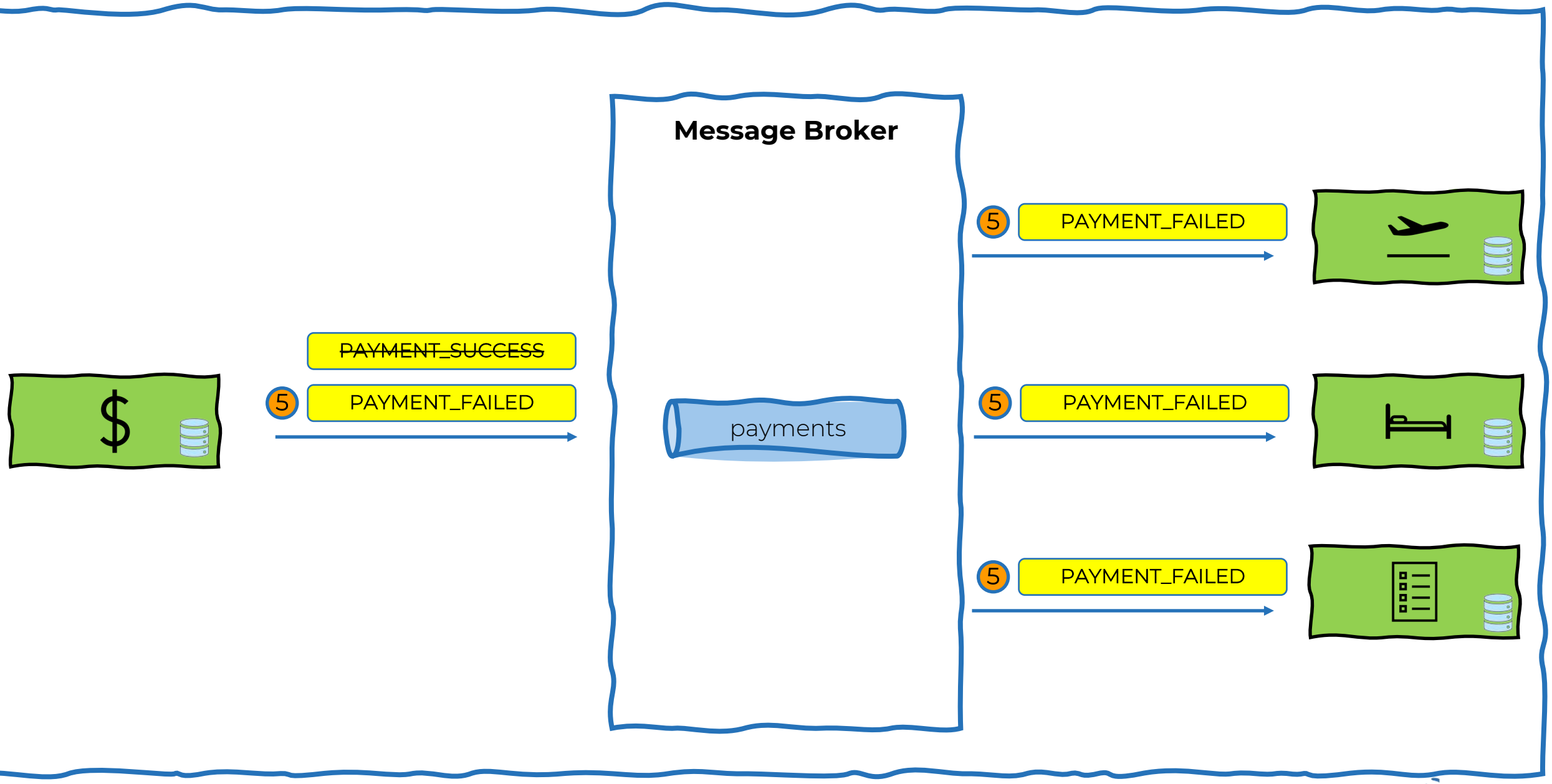
SAGA PATTERN

- SAGA is a sequence of local transactions that are coordinate using messaging
- Each local transaction updates data in a single service
- On failure due to the violation of a business rule, it must execute compensation transactions to explicitly undo changes

CHOREOGRAPHY-BASED SAGA



5. PAYMENT_FAILED



VORTEILE / NACHTEILE

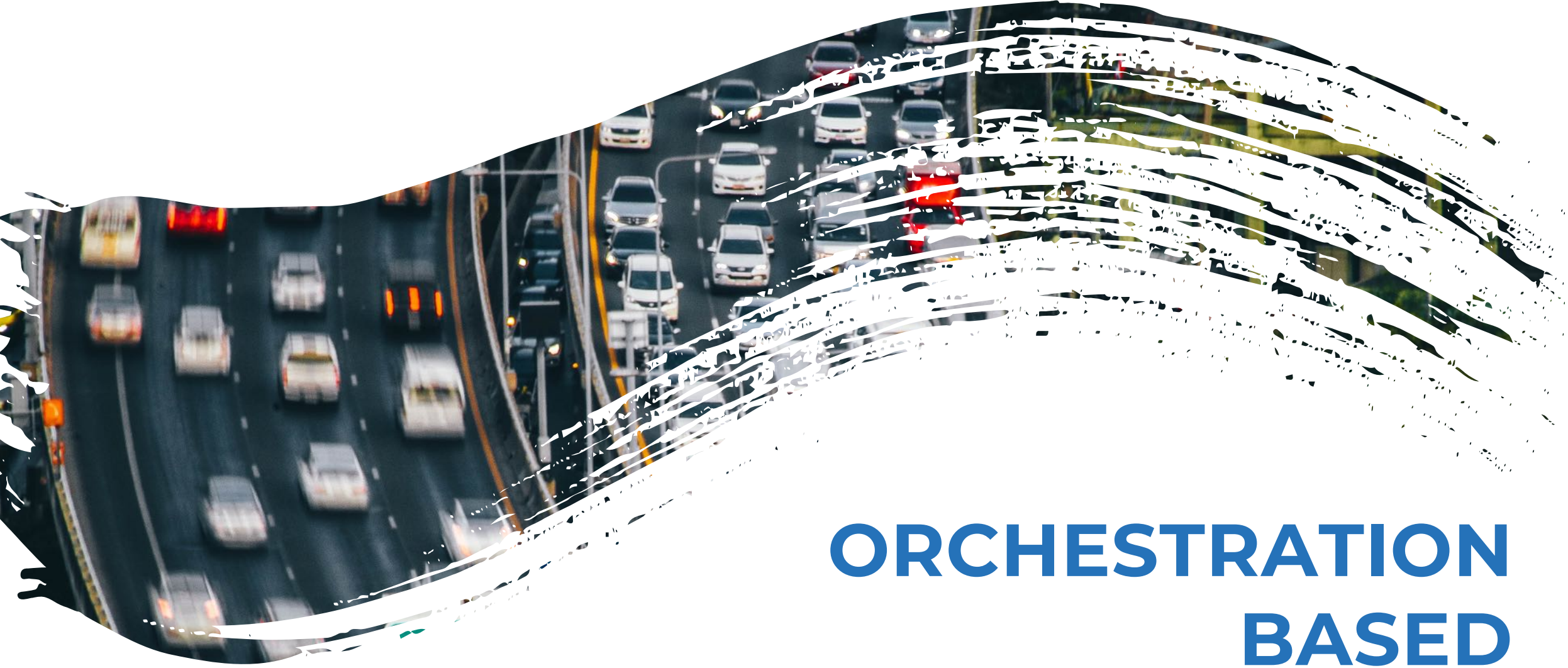
Einfachheit der Schritte

Loose Koppelung

Schwieriger zu verstehen

Zyklische Abhängigkeiten zwischen den Services

Risiko der zu engen Koppelung

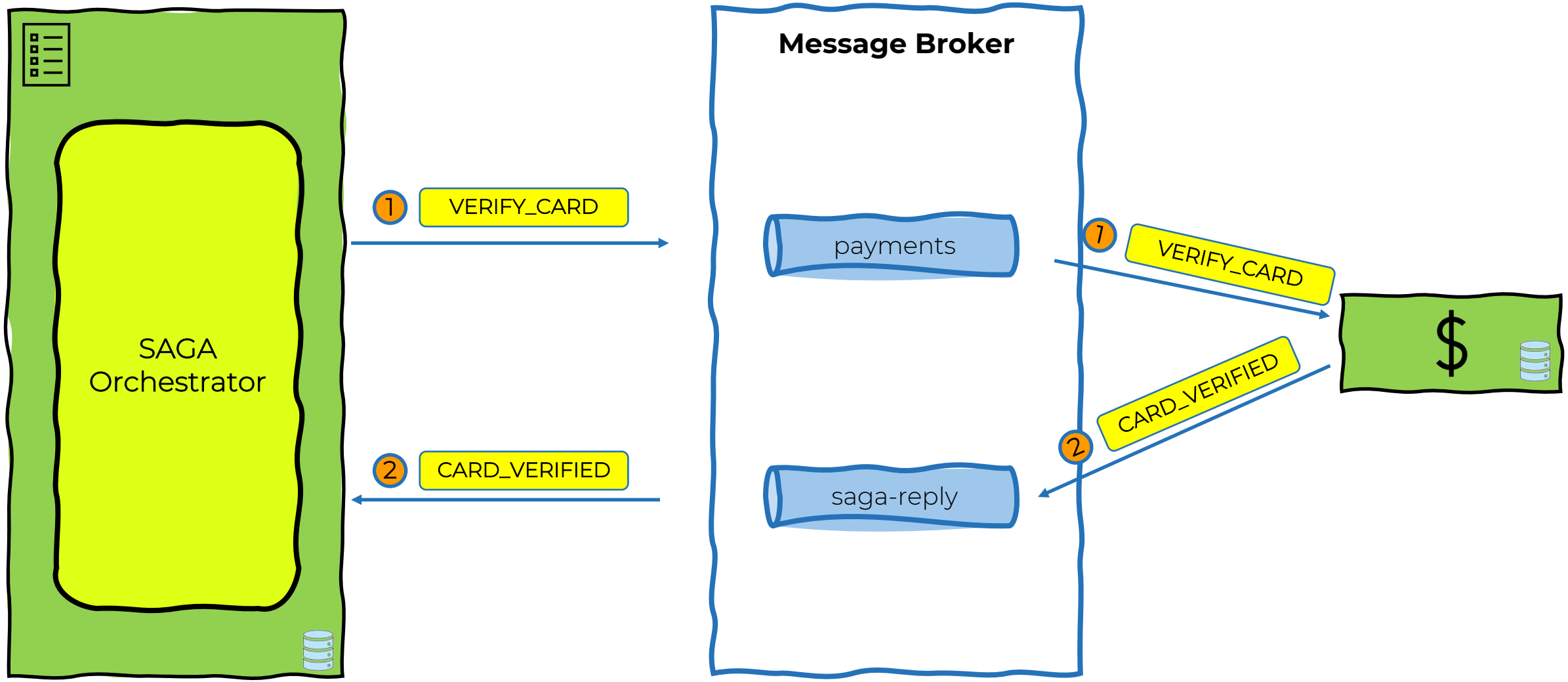


ORCHESTRATION BASED SAGA

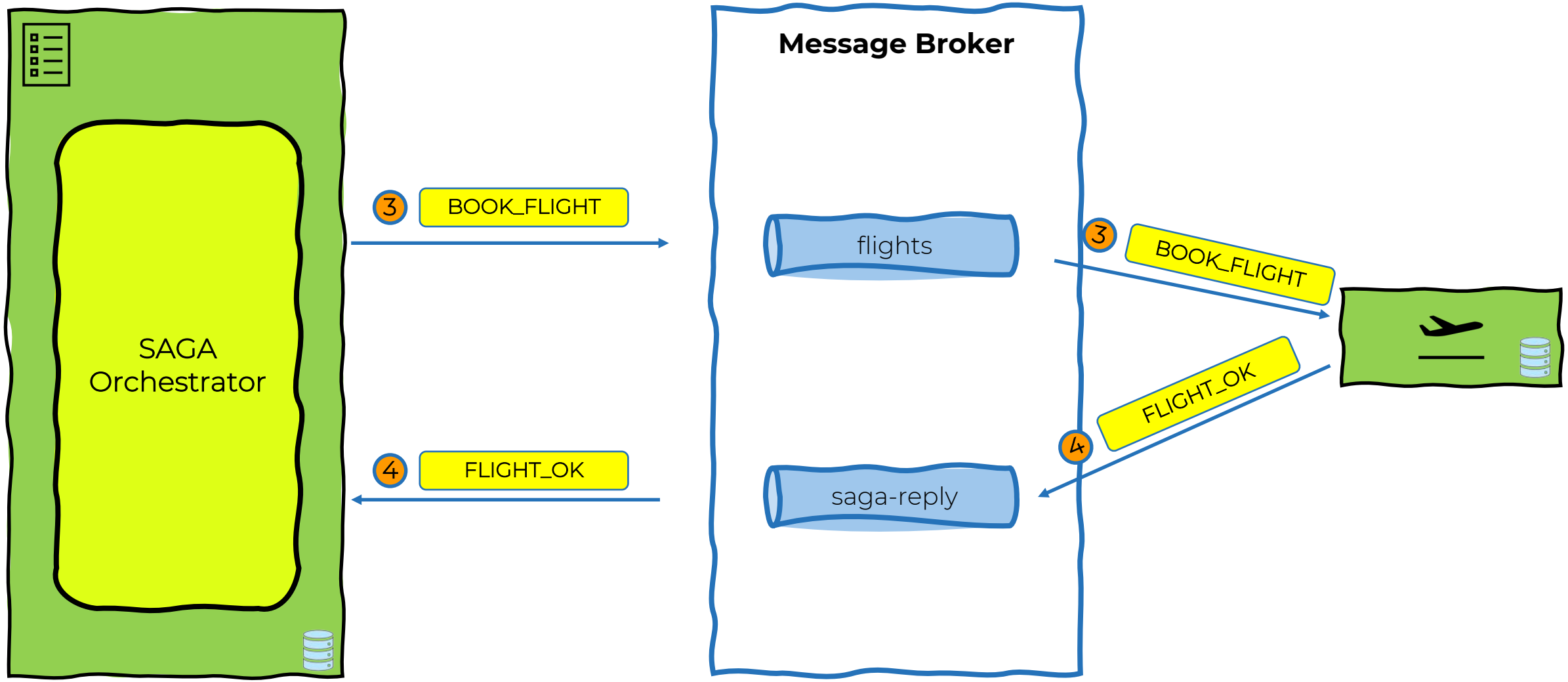
ORCHESTRATION-BASED SAGA

- eigenständige Orchestrator-Klasse mit der einzigen Verantwortung die Saga-Teilnehmer ihre Aufgaben zuzuweisen
- Command / Async-Reply-Style-Interaction

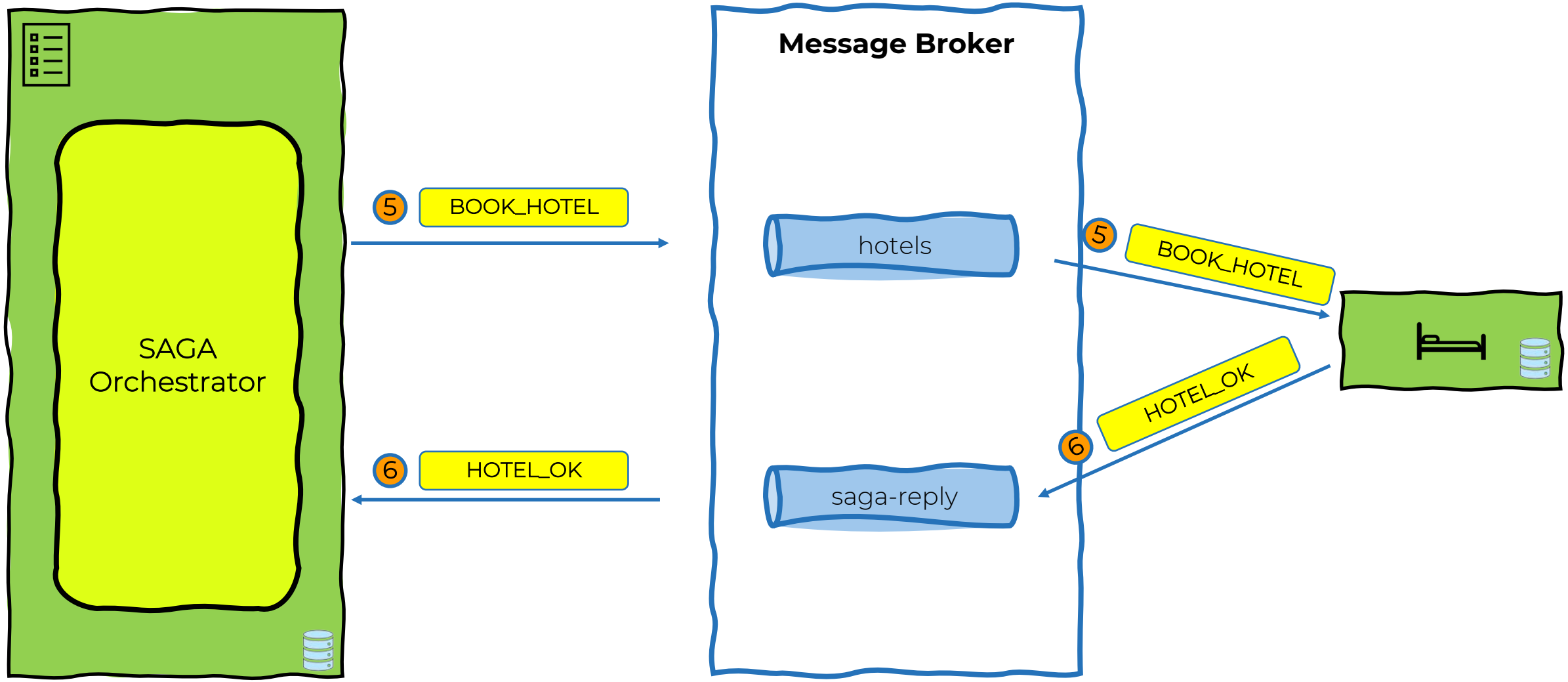
1. VERIFY_CARD



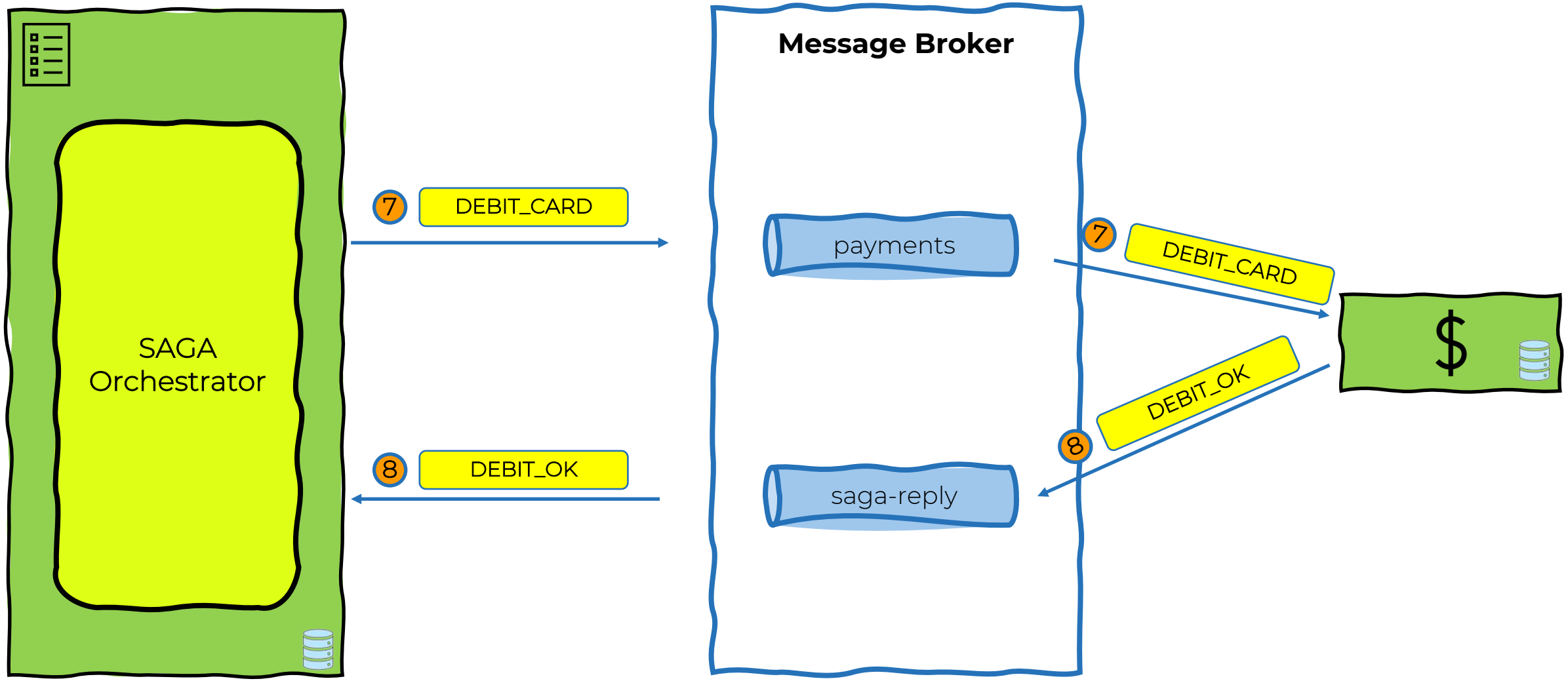
2. BOOK_FLIGHT



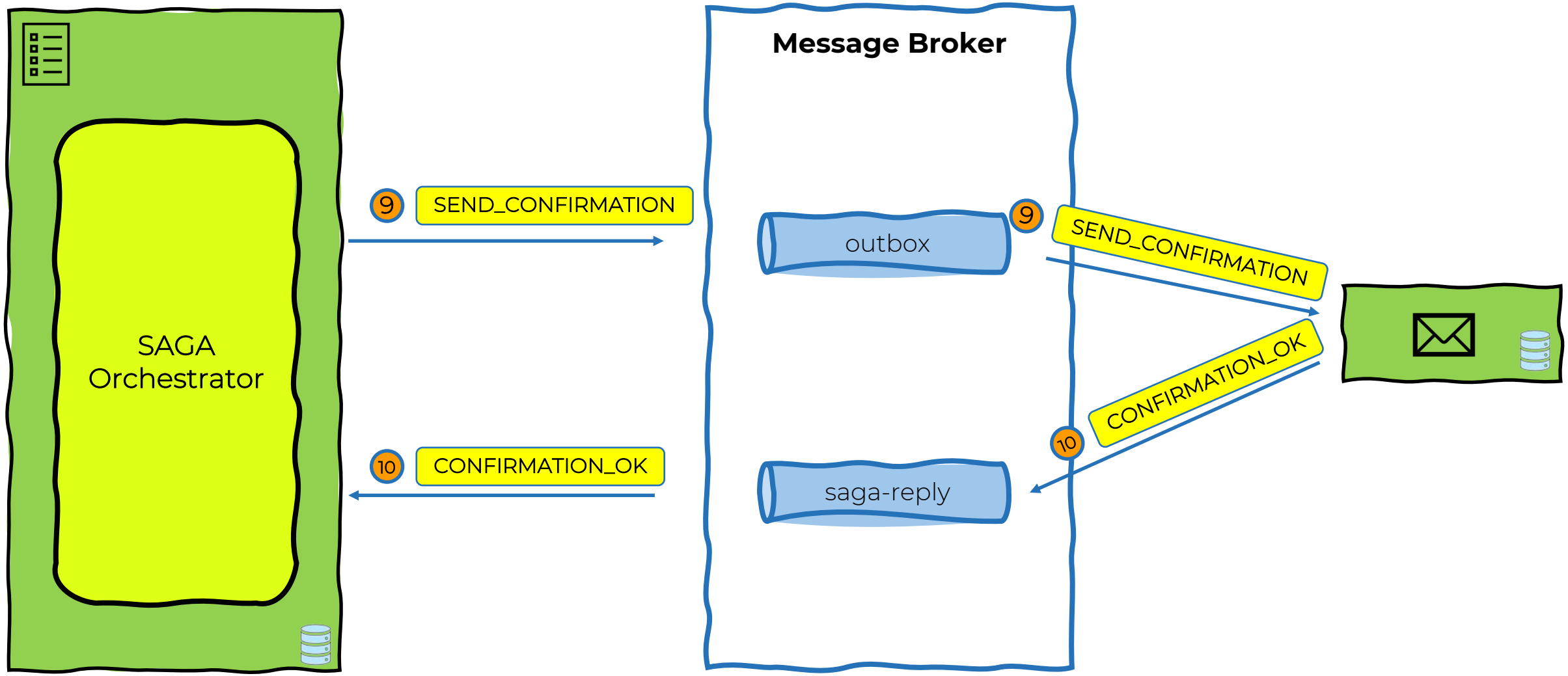
3. BOOK_HOTEL


SAGA
Orchestrator
5
BOOK_HOTEL
6
HOTEL_OK
Message Broker
hotels
saga-reply
5
BOOK_HOTEL
HOTEL_OK
6

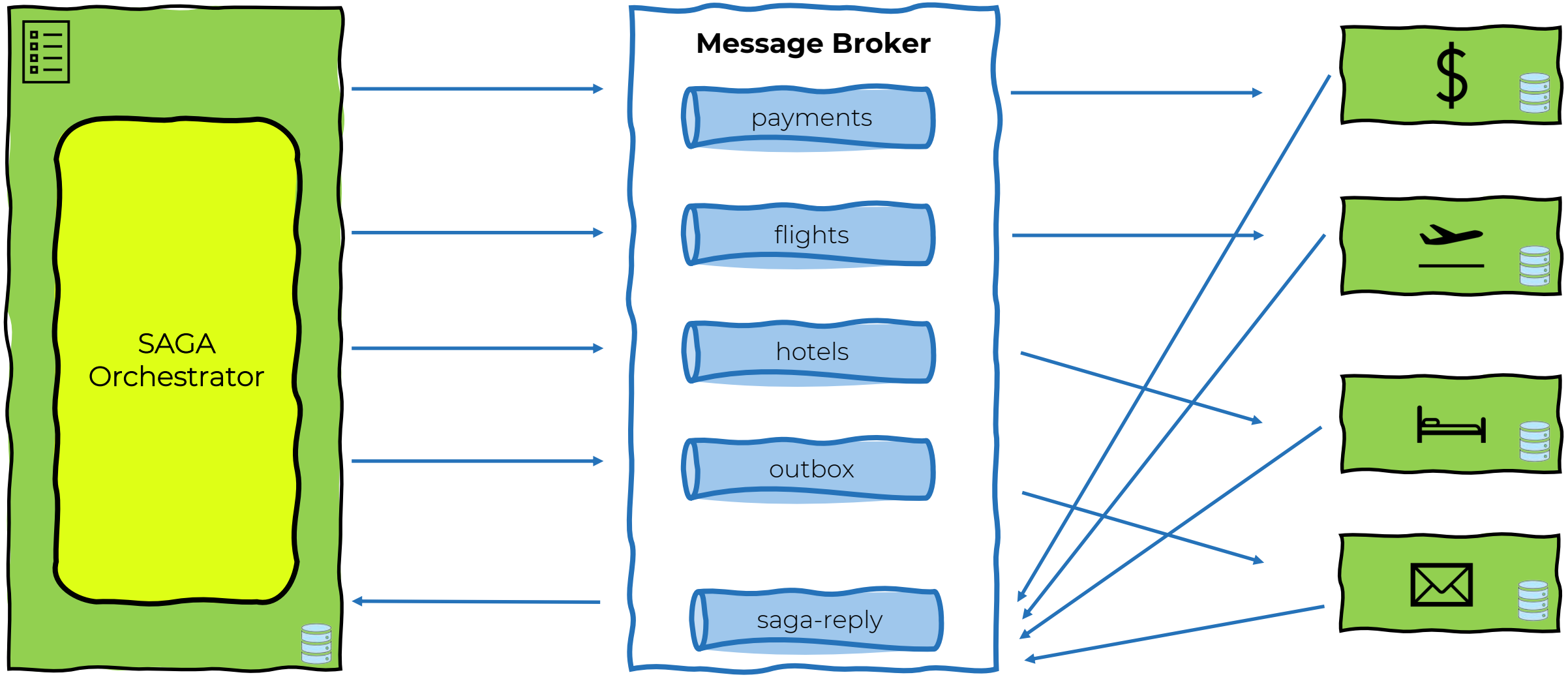
4. DEBIT_CARD


SAGA
Orchestrator
7
DEBIT_CARD
8
DEBIT_OK
Message Broker
payments
saga-reply
7
DEBIT_CARD
DEBIT_OK
8
$

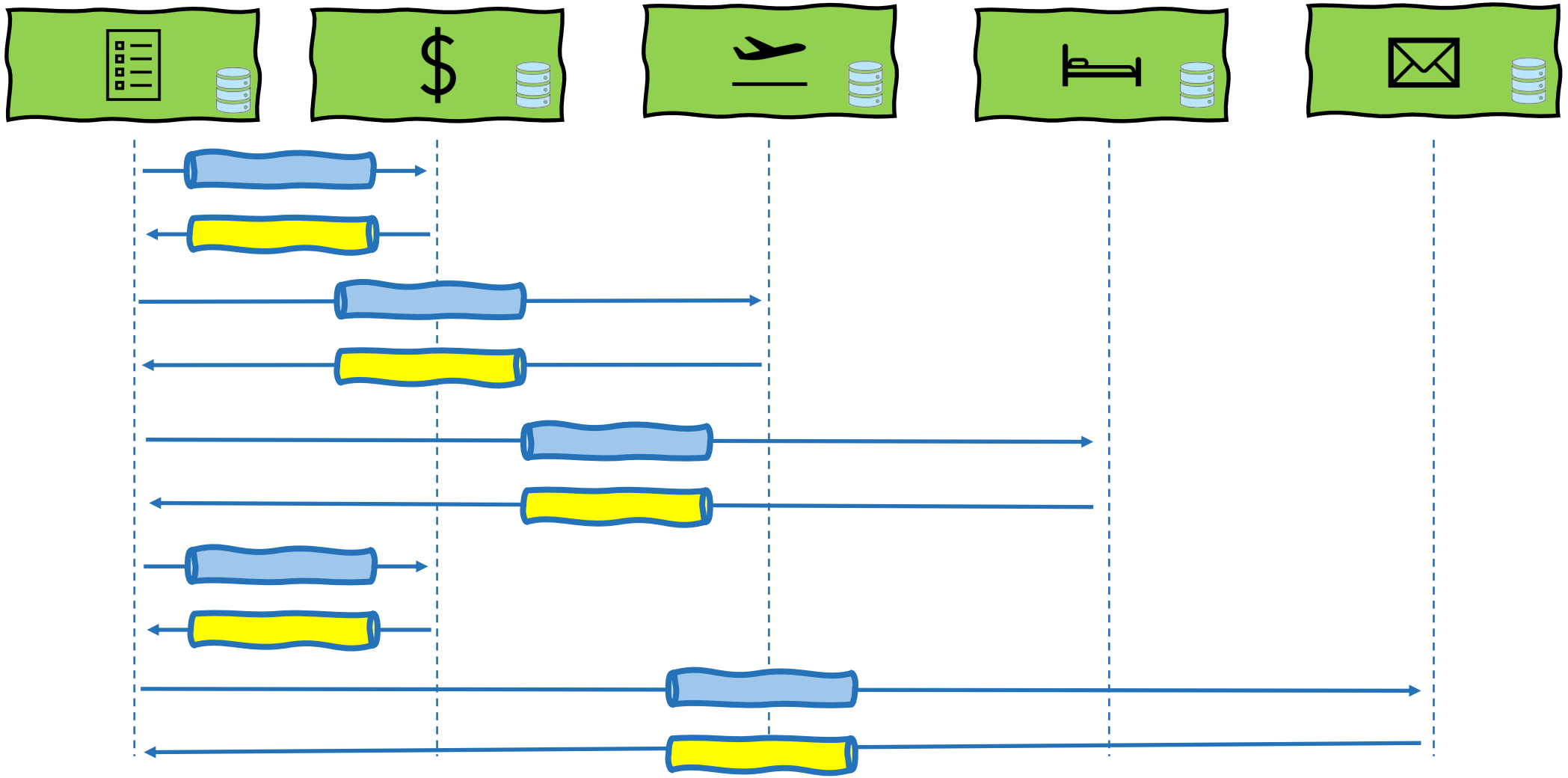
5. SEND_CONFIRMATION



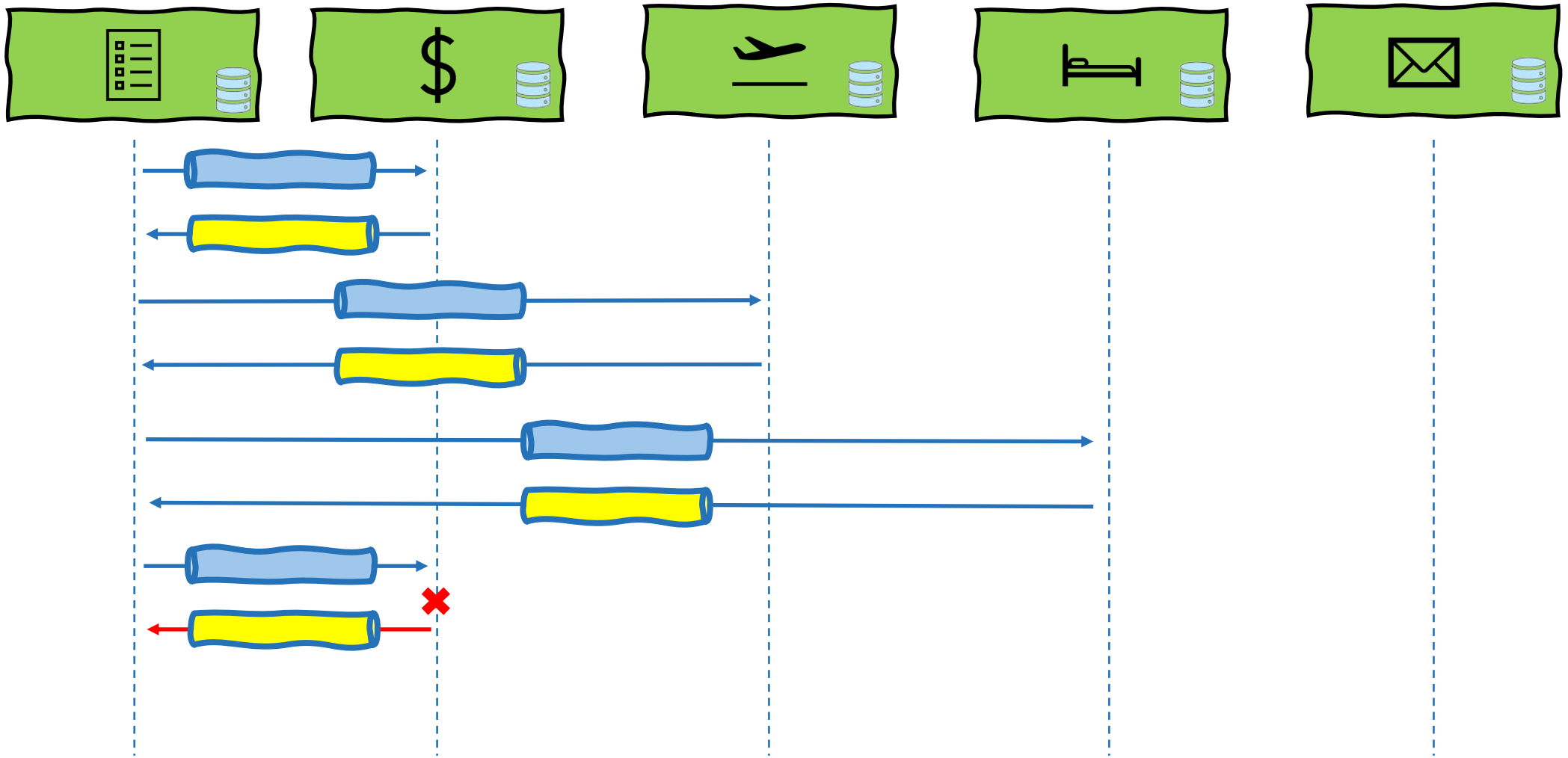
ORCHESTRATION-BASED SAGA



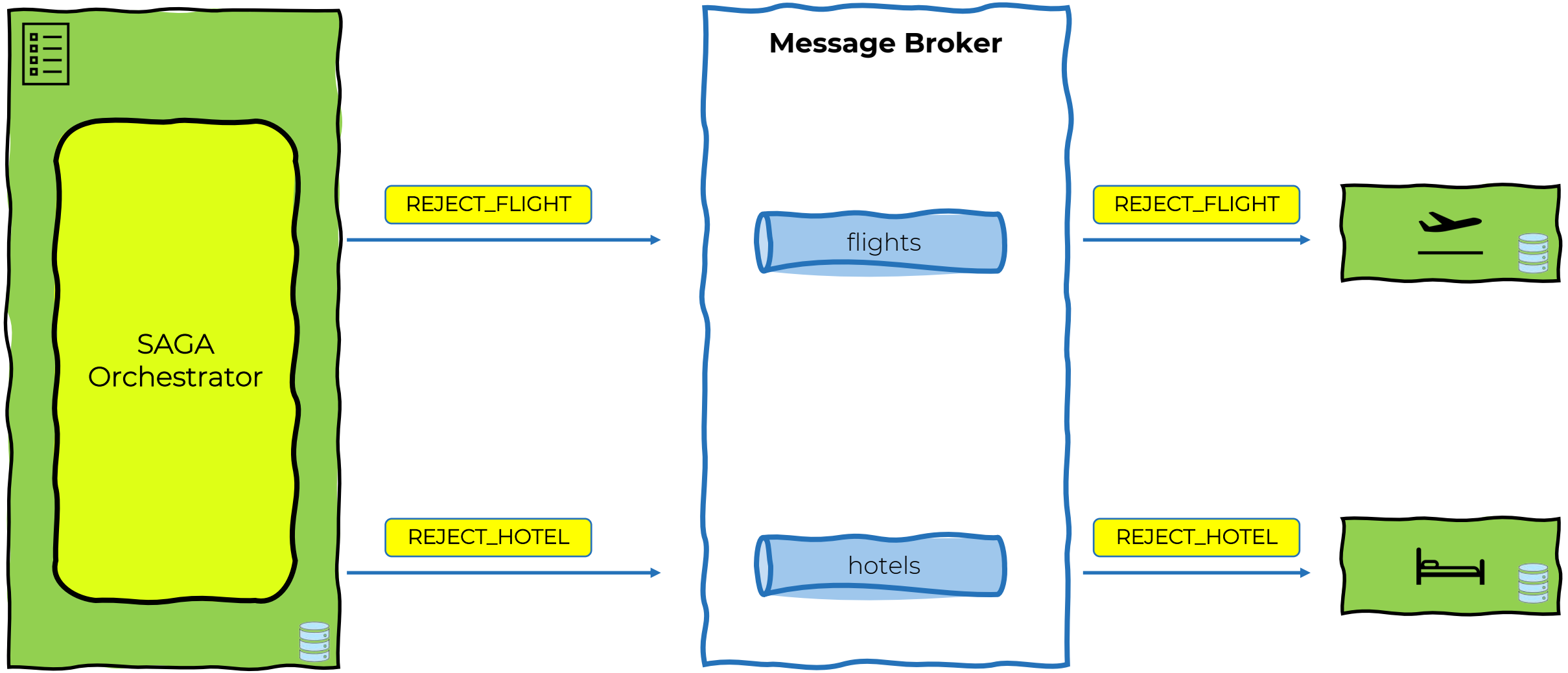
ORCHESTRATION-BASED SAGA



ORCHESTRATION-BASED SAGA



REJECT SERVICES



VORTEILE / NACHTEILE

- **Einfache Abhängigkeiten**
- **Loose Koppelung**
- **Trennung der Verantwortlichkeit**

- **Risiko der Zentralisierung**

SAGA-STRUKTUR

	Step	Service	Transaction	Compensation Transaction
A	1	PaymentService	verifyCard()	---
A	2	FlightBookingService	bookFlight()	rejectFlight()
A	3	HotelBookingService	bookHotel()	rejectHotel()
B	4	PaymentService	debitCard()	---
C	5	ConfirmationService	sendConfirmation()	---

A Compensatable transactions

B Pivot transactions

C Retriable transactions

EDGE CASES

ACID IN SAGA

- ✓ **A** tomicity
- ✓ **C** onsistency
- ✗ **I** solation ➔ Lack Of Isolation
- ✓ **D** urability

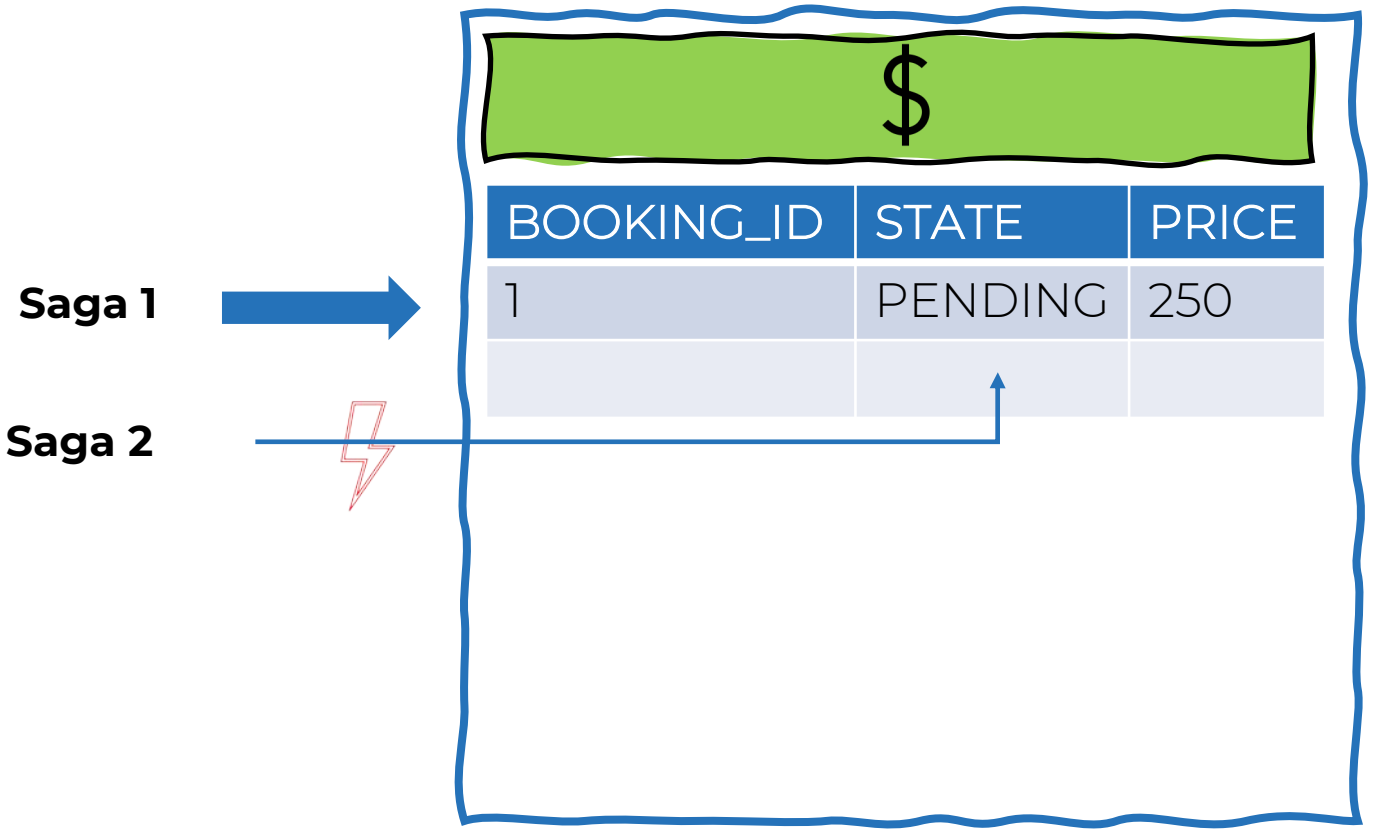
ANOMALIEN IN SAGA

- Lost Updates
- Dirty Reads
- Fuzzy / Nonerepeatable Reads

COUNTERMEASURES

- Semantic Lock
- Commutative Updates
- Reread Value
- Pessimistic View

SEMANTIC LOCK



SEMANTIC LOCK

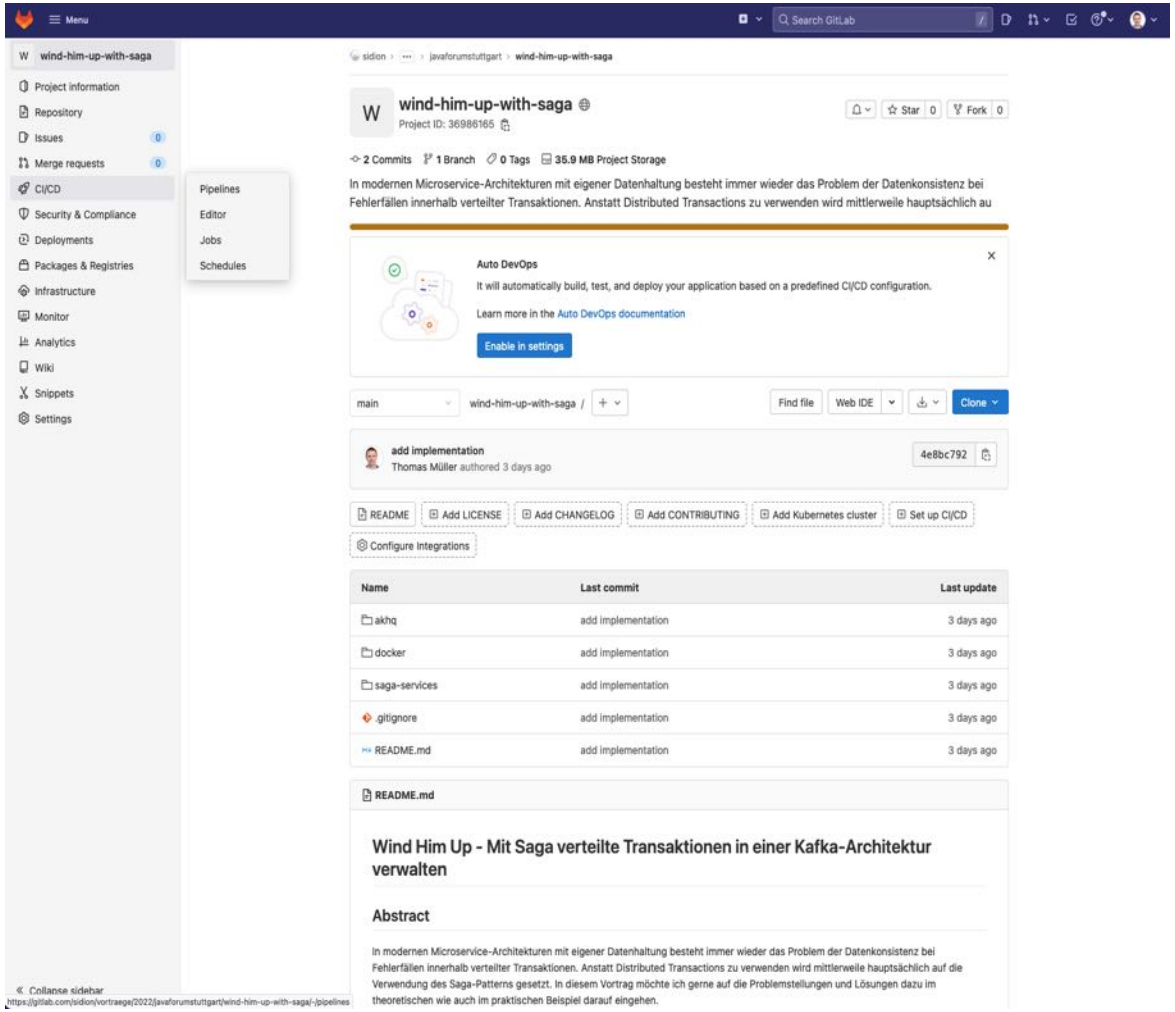
	Step	Service	SUCCES	FAIL
A	1	PaymentService	🔓	🔓
A	2	FlightBookingService	🔓	🔓
A	3	HotelBookingService	🔓	🔓
B	4	PaymentService	🔓	🔓
C	5	ConfirmationService	🔓	---

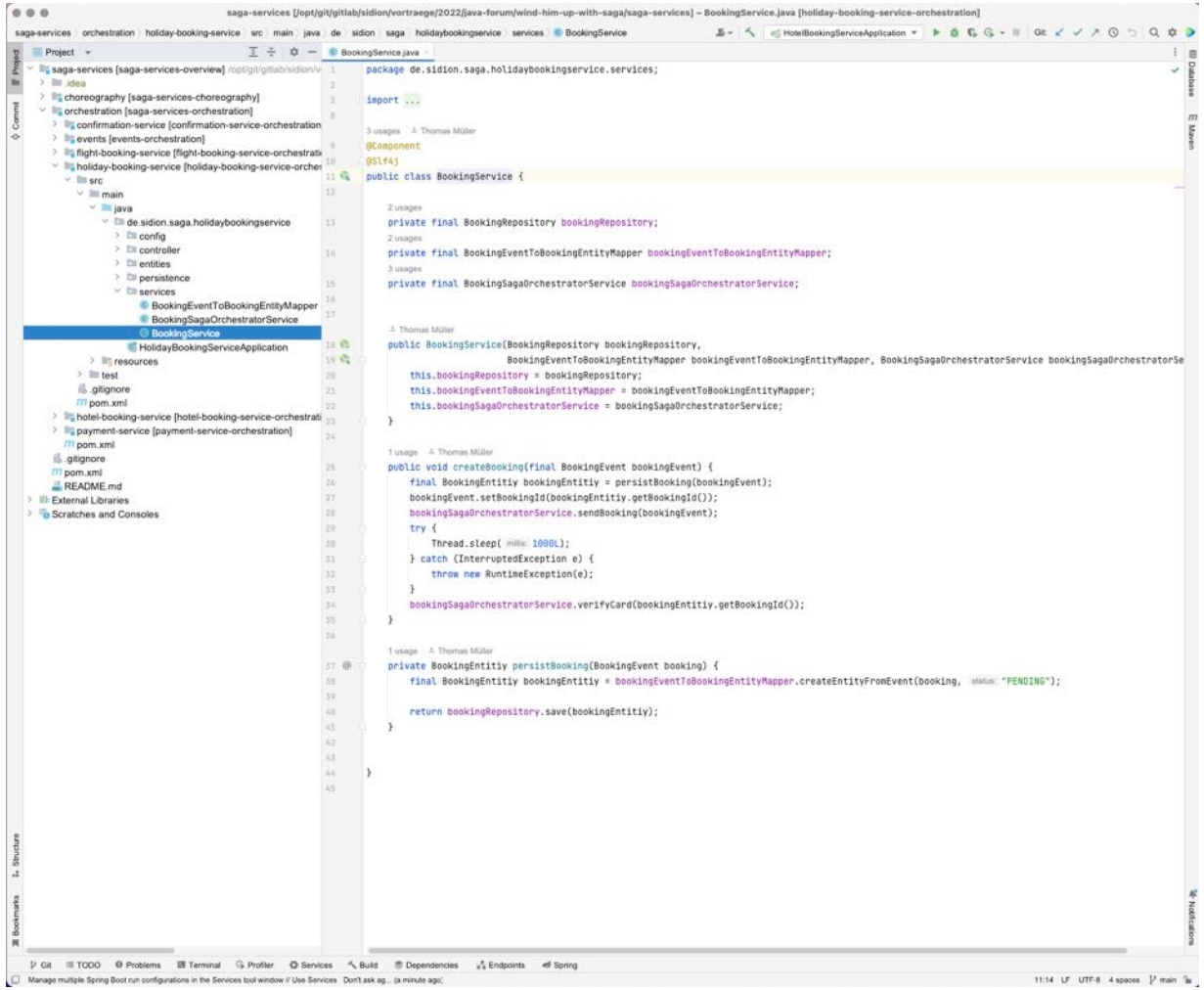
A Compensatable transactions

B Pivot transactions

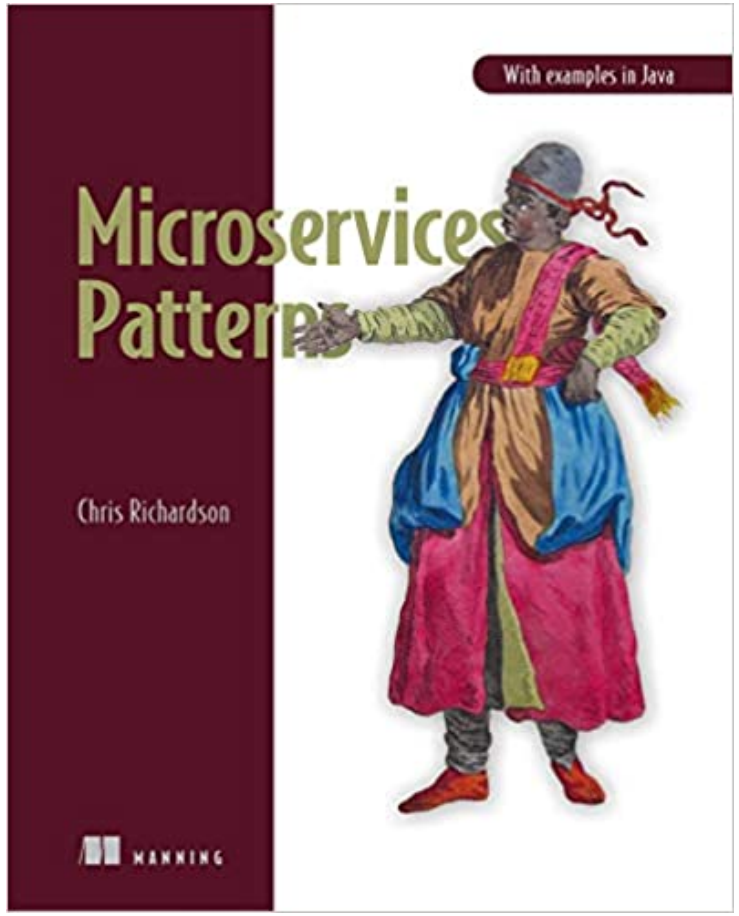
C Retriable transactions

CODING


wind-him-up-with-saga
Project information
Repository
Issues
Merge requests
CI/CD
Security & Compliance
Deployments
Packages & Registries
Infrastructure
Monitor
Analytics
Wiki
Snippets
Settings
Pipelines
Editor
Jobs
Schedules
wind-him-up-with-saga
Project ID: 36986165
2 Commits 1 Branch 0 Tags 35.9 MB Project Storage
In modernen Microservice-Architekturen mit eigener Datenhaltung besteht immer wieder das Problem der Datenkonsistenz bei Fehlerfällen innerhalb verteilter Transaktionen. Anstatt Distributed Transactions zu verwenden wird mittlerweile hauptsächlich au
Auto DevOps
It will automatically build, test, and deploy your application based on a predefined CI/CD configuration.
Learn more in the Auto DevOps documentation
Enable in settings
add implementation
Thomas Müller authored 3 days ago
4e8bc792
README
Add LICENSE
Add CHANGELOG
Add CONTRIBUTING
Add Kubernetes cluster
Set up CI/CD
Configure Integrations
Name
Last commit
Last update
akhq
docker
saga-services
.gitignore
README.md
add implementation
3 days ago
README.md
Wind Him Up - Mit Saga verteilte Transaktionen in einer Kafka-Architektur verwalten
Abstract
In modernen Microservice-Architekturen mit eigener Datenhaltung besteht immer wieder das Problem der Datenkonsistenz bei Fehlerfällen innerhalb verteilter Transaktionen. Anstatt Distributed Transactions zu verwenden wird mittlerweile hauptsächlich auf die Verwendung des Saga-Patterns gesetzt. In diesem Vortrag möchte ich gerne auf die Problemstellungen und Lösungen dazu im theoretischen wie auch im praktischen Beispiel darauf eingehen.


BookingService.java
package de.sidion.saga.holidaybookingservice.services;
@Component
@Slf4j
public class BookingService {
private final BookingRepository bookingRepository;
private final BookingEventToBookingEntityMapper bookingEventToBookingEntityMapper;
private final BookingSagaOrchestratorService bookingSagaOrchestratorService;
public BookingService(BookingRepository bookingRepository,
BookingEventToBookingEntityMapper bookingEventToBookingEntityMapper, BookingSagaOrchestratorService bookingSagaOrchestratorSe
this.bookingRepository = bookingRepository;
this.bookingEventToBookingEntityMapper = bookingEventToBookingEntityMapper;
this.bookingSagaOrchestratorService = bookingSagaOrchestratorService;
public void createBooking(final BookingEvent bookingEvent) {
final BookingEntitiy bookingEntitiy = persistBooking(bookingEvent);
bookingEvent.setBookingId(bookingEntitiy.getBookingId());
bookingSagaOrchestratorService.sendBooking(bookingEvent);
try {
Thread.sleep(millis: 1000L);
} catch (InterruptedException e) {
throw new RuntimeException(e);
bookingSagaOrchestratorService.verifyCard(bookingEntitiy.getBookingId());
private BookingEntitiy persistBooking(BookingEvent booking) {
final BookingEntitiy bookingEntitiy = bookingEventToBookingEntityMapper.createEntityFromEvent(booking, status: "PENDING");
return bookingRepository.save(bookingEntitiy);

BUCHVORSCHLAG



Microservices Patterns

von Chris Richardson

CODE

Die ausführlichen Codebeispiele findet Ihr unter:

https://gitlab.com/zaroselectro-public/saga-pattern


sidion
SIDION
Zuhören. Analysieren. Beraten.