



# Event Sourcing bei



**Ronny Hartenstein** @rhflow\_de

Portal-Anschubser und Bootstrapper

## Softwareentwicklung & Projektbegleitung

Rating, Scoring, Prozessdesigner,  
exklusive Schnittstellen, Zinsrechner,  
Bausparpläne in Java / PHP

## Rechenzentrum, Ops, DevOps

ISO/IEC 27001 und  
Reliable Data Center

**100 Leute, gemischte Rollen**

**Obst, Frühstück, kostenloses Parken**

bei Fragen einfach





# Ziel heute



Angst nehmen vor großen Begriffen

Entscheidungshilfe fürs nächste Projekt

Gefühl wo es nicht passt vermitteln

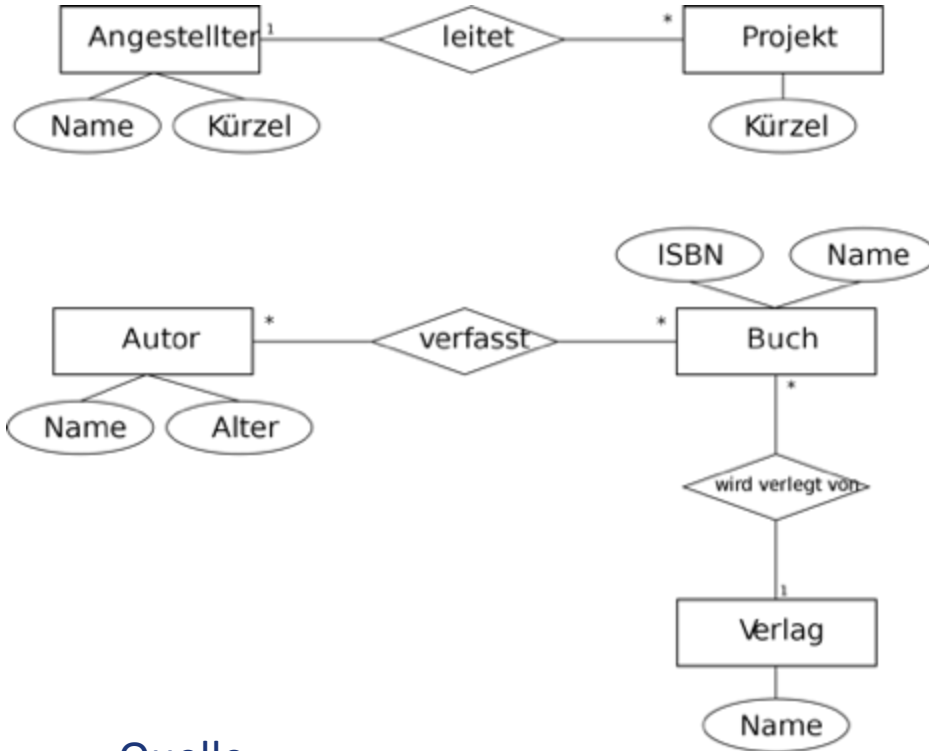
Idee geben für Komplexität zur Einführung

Konkrete Dinge auf die man aufpassen muss

# Der klassische Weg

---

# Entity-Relationship-Modell = Relationales Datenbankmodell



Entity = Tabelle  
Relation = Verknüpfung

## CRUD API:

GET /angestellte  
GET /angestellter/1  
PUT /angestellter/1  
DEL /angestellter/1  
...  
GET /angestellter/1/projekte  
...  
PUT /projekt/1

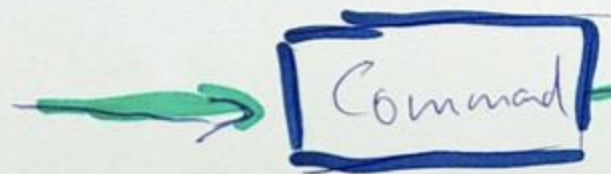
[Quelle](#)

**Und mit Event Sourcing?**

**Was ist ES?**

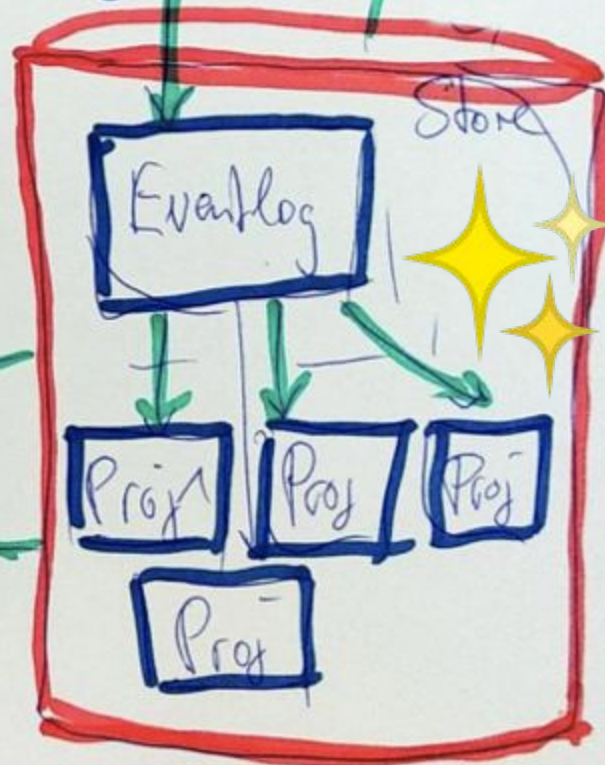
**Was beinhaltet ES?**

---



Schreiben

Event Sourcing



lese





# Artefakte



**Command**                      was soll passieren (Zukunft)

Handlungsaufforderung eines Akteurs mit dessen Intention,  
optional Seiteneffekte, publizieren Events

**Event**                              was ist passiert (Vergangenheit)

objektives Geschehen in der Vergangenheit

**Store (Eventlog)**              wo es gespeichert ist

**Projection**                      wie es dargestellt werden soll

**Aggregate[Root]**              wie ist der aktuelle Zustand einer Entität



## **CQRS, einfach so..**

geht gar nicht ohne,

weil Ad-hoc-Query auf Eventlog langsam ist

Lösung: Projektionen, später..

# Naming is hard



Domain Driven Development hilft

**Ubiquitous Language** im Bounded Context

Bounded Context → gebundener (geschlossener) Kontext

**Event Storming**

# Ubiquitous Language



gemeinsame allgegenwärtige rigorose Sprache

Genau ein Begriff für ein Artefakt - Glossar

Verwendung in Kommunikation, Challenges, Story, Entwicklung

Hilfsmittel: Event Storming

.. spätestens bevor die 1. Zeile Code entsteht

(ein Beispiel)

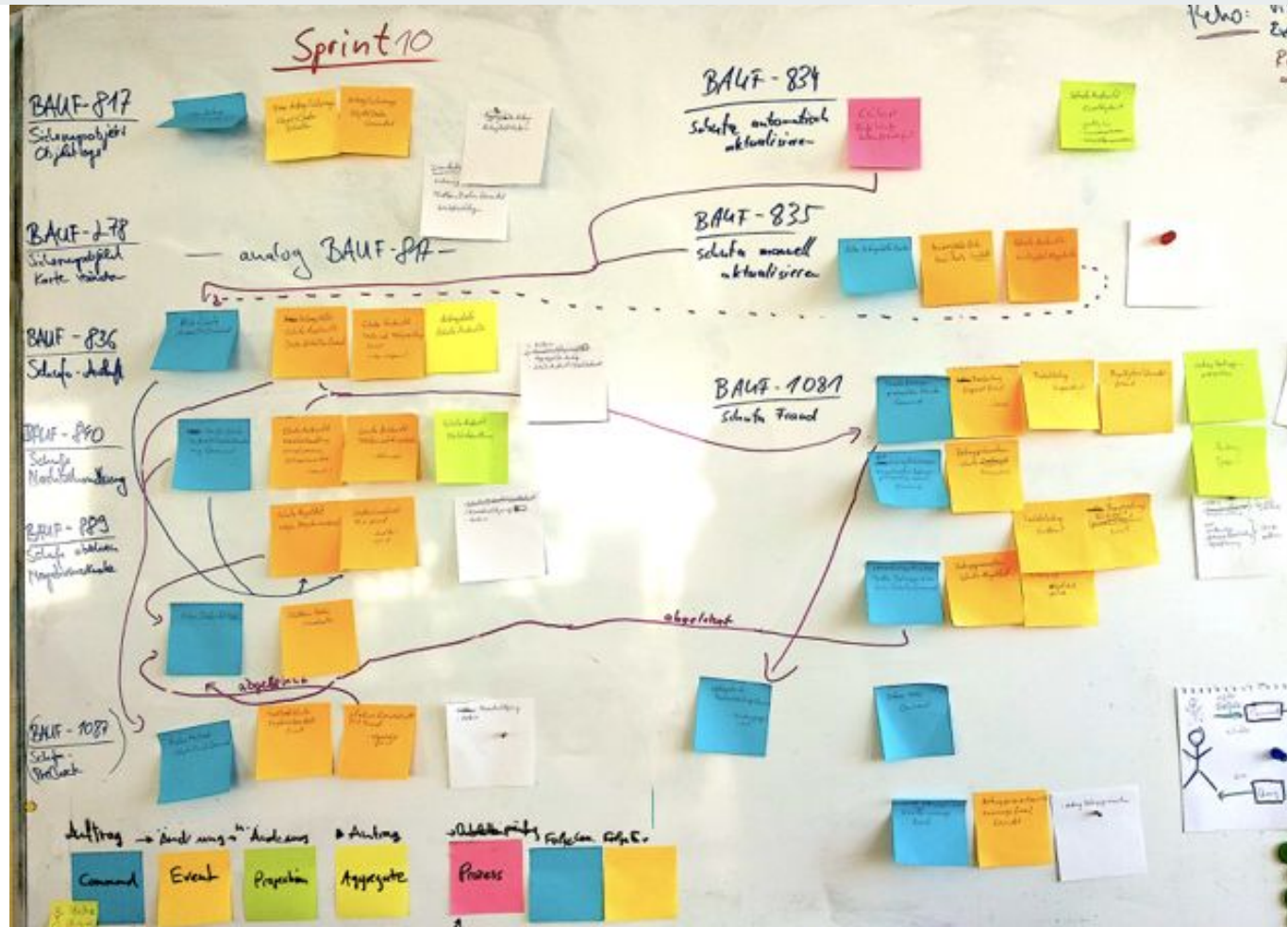
## Command

## Event

## Projection

## Aggregate

## Prozess





Neuer Nutzer  
Registriert

Registrierungs  
Bestaetigungs  
Mail Gesendet

Persönliche  
Daten Geaendert

Registriere  
Nutzer

Neuer Nutzer  
Registriert

Sende  
Registrierungs  
Bestaetigungs  
Mail

Registrierungs  
Bestaetigungs  
Mail Gesendet

Ändere  
Persönliche  
Daten

Persönliche  
Daten Geändert

# Query über Projektionen

Replay der Events, nicht der Commands!  
(keine doppelten Seiteneffekte)

“Save everything, Replay what you need”

haben keine Relationen, keine Normalisierung

Je Feature eine Projektion

einfach zu erstellen,  
einfach zu löschen,  
einfach zu ändern

Rebuild ist kein großes Problem bei  
Datenmengen < Google

projection_aggregate_zu_antrag
projection_antraege_in_bearbeitung
projection_antragsreporting_export
projection_antrags_import
projection_antrags_uuid_nummern
projection_antrag_betrugspraevention
projection_antrag_dubletten
projection_antrag_dubletten_ablehnung
projection_antrag_dubletten_pruefung
projection_antrag_durchlaufzeit
projection_antrag_feld_historie
projection_antrag_historie
projection_antrag_notizen
projection_antrag_sayt
projection_antrag_sperre
projection_antrag_status
projection_antrag_uebergabe
projection_antrag_unterlagen_pruefung
projection_benachrichtigung
projection_dokument_uebertragung_plattform
projection_erinnerungs_email_kein_unterlageneingang
projection_reporting_antrag_status
projection_reporting_in_bearbeitung_genommen
projection_reporting_stand_der_bearbeitung
projection_schufa_auskunft_gueltigkeit
projection_schufa_auskunft_nachbehandlung
projection_scoring_note_neukunde_abfrage
projection_statusreporting_fachabteilung
projection_statusreporting_kdm
projection_tagesplanung
projection_tagesreporting
projection_wiedervorlage



Ereignisse									
id	aggregate_uid	aggregate_name	event_uid	event_name	version	erfolgt_am	daten	meta	
erfolge	UID des Aggregats	Name des Aggregats	UID des Ereignisses	Name des Ereignisses	Version	Zeitpunkt des Ereignisses	Daten des Ereignisses als JSON	Meta-Daten des Ereignisses als JSON	
1	e3890734-17a3-4914-947e-c290f33a03fb	Nutzer	b425f021-5ac1-416d-8b84-74c060fa4526	NeuerNutzerRegistriert	1	2019-07-16 17:06:34.808101	{ "vorname": "Annerose", "nachname": "Sch\r\u00f6der", ...	{ "erfolgt_durch": "", "ausloesender_command": "Regist...	
2	e3890734-17a3-4914-947e-c290f33a03fb	Nutzer	ce201b20-eee6-4944-9c8b-8c246f192d05	RegistrierungsBestaetigungsMailGesendet	2	2019-07-16 17:06:34.885097	{ "link": "https://vimmportal.local/registrierung..."	{ "erfolgt_durch": "", "ausloesender_command": "SendeR...	
3	e3890734-17a3-4914-947e-c290f33a03fb	Nutzer	9467e366-82f4-4028-a050-fbc5d67d1b71	NutzerFreigeschalten	3	2019-07-16 17:06:34.900051	[]	{ "erfolgt_durch": "", "ausloesender_command": "Schalt...	
4	226dc8ed-09c5-546d-a2cd-1337549e5706	ImmoFio	711ac966-88c6-4cb3-b180-c47bdd3e4039	FiolmmobilieZuMerkzettelHinzuefuegt	1	2019-07-16 17:06:34.914210	{ "nutzer_uid": "e3890734-17a3-4914-947e-c290f33a03..."	{ "erfolgt_durch": "", "ausloesender_command": "FuegeF...	
5	226dc8ed-09c5-546d-a2cd-1337549e5706	ImmoFio	bf07056f-41ed-4cbd-8188-8bfd44b74ea3	FiolmmobilieVonMerkzettelEntfernt	2	2019-07-16 17:06:34.929864	[]	{ "erfolgt_durch": "", "ausloesender_command": "Entfer...	
6	94466247-6158-4092-9934-c646fda24ca5	Nutzer	a1969b70-4b26-4573-8ba5-b4971951f89a	NeuerNutzerRegistriert	1	2019-07-16 17:06:35.034354	{ "vorname": "Ingeburg", "nachname": "Lauer", "email": "...	{ "erfolgt_durch": "", "ausloesender_command": "Regist...	
7	94466247-6158-4092-9934-c646fda24ca5	Nutzer	1d57f7f7-2b52-4cb6-8d3e-009a90bb6526	RegistrierungsBestaetigungsMailGesendet	2	2019-07-16 17:06:35.080262	{ "link": "https://vimmportal.local/registrierung..."	{ "erfolgt_durch": "", "ausloesender_command": "SendeR...	
8	94466247-6158-4092-9934-c646fda24ca5	Nutzer	1705e28d-d12b-49e2-accf-5df29876fbc4	NutzerFreigeschalten	3	2019-07-16 17:06:35.092484	[]	{ "erfolgt_durch": "", "ausloesender_command": "Schalt...	
9	00867b4e-924a-42dd-85dd-d6a1d7c7d723	Nutzer	45aea971-6852-4112-8450-4baf05a2307c	NeuerNutzerRegistriert	1	2019-07-16 17:06:35.202884	{ "vorname": "Kirstin", "nachname": "Vo\u00df", "email": "...	{ "erfolgt_durch": "", "ausloesender_command": "Regist...	
10	00867b4e-924a-42dd-85dd-d6a1d7c7d723	Nutzer	e3130995-3487-495e-ab59-c521b449f86f	RegistrierungsBestaetigungsMailGesendet	2	2019-07-16 17:06:35.244811	{ "link": "https://vimmportal.local/registrierung..."	{ "erfolgt_durch": "", "ausloesender_command": "SendeR...	
11 onsle	00867b4e-924a-42dd-85dd-d6a1d7c7d723	Nutzer	75bd27d8-e97a-40c6-aa92-	NutzerFreigeschalten	3	2019-07-16 17:06:35.260239	[]	{ "erfolgt_durch": "", "ausloesender_command": "Schalt...	

**Wann passt ES? Wann nicht?**

—



**Super bei..** Anforderungen mit zeitlichen Relationen -> Prozesse

**Schlecht bei..** reinen CRUD Anwendungen


**Don't do it..** Subdomains wie Auth und Nutzermanagement -  
(ist das falsche Beispiel zum lernen - haben wir mittlerweile aber auch gemacht :)

Beispiel:

- Kreditantrag kommt von Plattform rein
- wird mit zig Daten von APIs befüllt (Schufa, Rating, Stammdaten)
- Bearbeiter übernimmt Antrag, fordert Unterlagen nach, automatische Erinnerung nach x Tagen
- übergibt an andere Fachabteilung, die hat eine gewisse Zeit zum antworten
- automatische Statuswechsel nach Ablauf von Servicelevels
- u.a. Schufa-Auskünfte haben nur begrenzte Gültigkeit, müssen neu gezogen werden

**Wie haben wir ES  
implementiert?**

—



## Was sind die Probleme wenn Framework XYZ die Lösung ist?

Wir nutzen Prooph nicht,  
haben alles selbst gebaut ..  
.. und die Schmerzen durchlebt

Im Code liegt die Wahrheit! 🙏

Command

Event

Aggregat

Eventlog

Commandbus

Projektor, Projektion

# abweichend von der reinen DDD Lehre...



eigentlich steckt die BI + Validierung im Aggregat

bei uns aber in den Command-Handlern

Aggregat ist bei uns nur ein Value-Objekt

eigentlich: Aggregat speichert Ereignisse, Repository persistiert diese im Eventlog

bei uns: CommandBus bekommt Events vom Command und persistiert und publiziert diese



# Prozesse

derzeit: ausprogrammierte Speziallogik

künftig: **Prozessmanager**

Event → löst Command aus → emittiert Events → können wieder Commands auslösen

DelayedCommands für spezielle Timings / API-Wiederaufrufe nach Fehlern

- API nach Timeout nochmal in x Min anfragen
- Email Token nach x Stunden ungültig / erneut senden

# Best Practise



Kudos an Erik Braun für  
seine Beratung!

- MariaDB/MySQL/Postgres + JSON
- standalone: agnostisch zu DB Impl. Details - Teil-Dump einfach auf anderes System spielen, Proj bauen, fertig
- Command ist nur Nachricht, getrennt von CommandHandler Logik
- Cmd -> CmdBus -> CmdHandler -> Events
- eigentlich: Events werden nicht zurückgemeldet, ist aber super fürs Testing
- erwartbare Probleme → separate Events
- Exceptions sind die absolute Ausnahme (= Runtime Exceptions)



# Best Practise



Synchron bleiben für reduzierte Komplexität

- kein asynchroner Event-Queue oder -Bus
- keine “eventually consistence“ und keine Async-Probleme
- aber auch weniger Flexibilität

möglichst direkte Migration per DB-Updates bei Event Änderungen

- keine Upcaster nötig

# Dark Side of ES

 Datenbank-Schema ist implizit und versteckt

→ Relation von Aggregat zu Aggregat in Ereignis als Data übergeben

Änderungen in Eventdaten → ⚡ Split/Join von Events

Upcasting? oder einfach ändern? + Projektion Rebuild

Information Immutability ≠ Data Immutability

Zero-Downtime-Migrations als Ziel

**Wie steht es um die  
Testfähigkeit?**

---



Commands vollständig testen

dann ist die BI komplett getestet und stabil für Änderung

Commands sind agnostisch gegenüber CLI-Script, REST-API, Controller

Pro-Tipp: Projektionen testen

# Lohnt der Aufwand?

---



Gibt es zeitlich gesteuerte Prozesse?

Ist es ein langfristiges Projekt?

Sind historisierte Daten spannend?

Ist Analyse des Kundenverhaltens wichtig?

Gibt es genügend Budget?

# Was nun mitnehmen?



Angst genommen vor den Begriffen und Akronymen

Entscheidungshilfe fürs nächste Projekt gegeben

Gefühl bekommen, wo es nicht passt

Idee dafür haben, was es ist  
und was es für Komplexität bei Einführung bedeutet

Konkrete Dinge, auf die man aufpassen muss,  
bzw. was einen erwartet



# Mindestens fürs nächste Projekt

Ubiquitous Language + Event Storming

Commands als BI Klammer





# Tooling

**Prooph:** quasi die Referenzimplementierung im PHP-Umfeld

<http://getprooph.org/>

**Eventstore:** eine DB speziell für Event Sourcing

<https://eventstore.org/>



# Quellen / Lese-Empfehlungen

DDD, Hexagonal, Onion, Clean, CQRS, ... How I put it all together

<https://herbertograca.com/2017/11/16/explicit-architecture-01-ddd-hexagonal-onion-clean-cqrs-how-i-put-it-all-together/>

The Missing Post About Event Sourcing

<https://medium.com/@nicolopigna/the-missing-post-about-event-sourcing-eac91f6e3784>

Dark Side of Event Sourcing

[https://www.researchgate.net/publication/315637858\\_The\\_dark\\_side\\_of\\_event\\_sourcing\\_Managing\\_data\\_conversion](https://www.researchgate.net/publication/315637858_The_dark_side_of_event_sourcing_Managing_data_conversion)

<https://speakerdeck.com/overeemm/kandddinsky-2017-the-dark-side-of-event-sourcing-managing-data-conversion?slide=96>



**Danke für die Aufmerksamkeit!**

Gibt es Fragen?

Lust bei uns mitzumachen?