Microservices und Event-getriebene Architektur · WS26

Abschluss des Grundlagenteils und Kick-off des Projektteils

MEA, Donnerstag, 26.11.2026



ArchiLab

Der Tag

10:00	Statusrunde zum Walking Skeleton
10:30	Teamarbeit mit dem Ziel: Der Happy Path läuft über alle Teams
12:30	Mittagspause
13:15	Vorführung des Walking Skeletons
14:00	Reflexion des Grundlagenteils: die Retrospektiven der Teams
15:00	Die Schwerpunkte des Projektteils
15:30	Kick-off: wie das Team kitchen-storage mit dem Agenten arbeitet
15:50	*Inhaltsimpuls*: Git-basiertes Arbeiten im Team
16:30	Ausblick auf den Projektteil und Hausaufgabe

ArchiLab

10:00 bis 10:30

Statusrunde zum Walking Skeleton

Was funktioniert schon, was noch nicht? Wer braucht wen?



Überblick

Team	Kafka	Test mit Kafka	REST-Tests	Doku	ADRs	ADR 0008	Tests in Pipeline	Retro
kitchen-storage	erledigt	weitgehend BoxEmptied, OrderFulfilled fehlen	erledigt	weitgehend Clients-Seite auf Branch	erledigt	erledigt	ja	offen leere Datei auf Branch
menu-ordering	teilweise CheckIn: Fix in !16	erledigt	erledigt	weitgehend Clients-Seite auf Branch	erledigt	erledigt	ja	erledigt
pickup	weitgehend Topic, DB seit heute auf master	erledigt	erledigt	weitgehend Katalogseite auf Branch	erledigt	erledigt	ja bei jedem Push	erledigt
reservation	weitgehend Topic, DB seit heute auf master	erledigt	weitgehend	offen	erledigt Issue offen	erledigt	teilweise nicht auf main	erledigt

> Seit dem 19.11. viel geschafft: pickup und reservation haben aufgeholt. Was die Vorführung heute aufhält, liegt fast nur zwischen den Teams

ArchiLab

Der Happy Path über alle Teams

Event	Producer schreibt auf	Consumer liest	
CheckIn reservation → menu-ordering	`bl.reservation.events.v1`, `reservation.check-in`	auf main noch `reservations`, `CheckIn`	**bricht** Fix in menu-ordering !16
MenuChanged menu-ordering → kitchen-storage	`bl.menu.events.v1`	dasselbe Topic, aber Zutaten-IDs unbekannt	**bricht** Rezept abgelehnt
OrderPlaced menu-ordering → kitchen-storage	`bl.order.events.v1`	dasselbe Topic und Typ	**hängt** ohne Rezept: UnknownDish
StationManningStatusChanged, OrderFulfilled kitchen-storage → menu-ordering	`bl.station.events.v1`, `bl.fulfillment.events.v1`	dieselben Topics und Typen	**passt** im Code
OrderFulfilled kitchen-storage → pickup	`bl.fulfillment.events.v1`	dasselbe Topic und Typ	**passt** im Code
BoxEmptied pickup → kitchen-storage	`bl.box.events.v1`	dasselbe Topic, Küche lernt daraus die Boxen	**passt** im Code
OrderPlaced menu-ordering → reservation	`bl.order.events.v1`	Listener da, Handler leer	**ohne Wirkung**

> Alle Topic-Namen passen jetzt zusammen, bis auf CheckIn auf main von menu-ordering. Gespielt hat den Happy Path noch niemand. Alle Topics gibt es auf master von local-dev-env erst seit heute 9:08

ArchiLab

local-dev-env: Topics und Datenbanken

Team	Topics in `config/kafka/init.sh`	Datenbank in `config/postgres/init.sql`
kitchen-storage	vier `bl.*`-Topics	`kitchen_storage`
menu-ordering	`bl.menu.events.v1`, `bl.order.events.v1`	`menu_ordering`
pickup	`bl.box.events.v1`	`pickup`
reservation	`bl.reservation.events.v1`	`reservation`

> Seit heute 9:08 alles auf master: die drei Branches der Teams gemergt, reservation dazu. reservation hatte Topic und Datenbank um 7:37 selbst auf einem Branch vorbereitet. Claude Code hat den Branch beim Vorbereiten des Merge-Skripts zu spät gesehen und die Werte deshalb aus der `application-kafka.yaml` von reservation übernommen. Der Inhalt ist derselbe, deshalb ist der Branch gelöscht. Jetzt bei allen: local-dev-env aktualisieren und mit `down -v` neu starten

ArchiLab

Worauf es beim Start ankommt

1. local-dev-env frisch: `down -v`, `up`, `kafka-init` endet mit Exited (0)
2. **jeder Service mit Kafka und eigener Datenbank:**
   kitchen-storage: Launch-Profil `http-postgres-kafka`
   menu-ordering, reservation: Spring-Profil `kafka`
   pickup: immer mit Kafka
3. pickup früh starten: meldet seine Boxen einmal als BoxEmptied, nur mit leerer Datenbank
4. Gericht in menu-ordering anlegen, bevor jemand bestellt
5. Gast checkt ein, dann bestellt er

> Ohne Profil laufen kitchen-storage, menu-ordering und reservation mit Stubs im Speicher. Dann ist alles grün, aber kein Event geht raus. Dazu passt der Post im Discord-Forum zu `@Profile("kafka")`: In den Lern-Modulen 3 und 4 startet der Service ohne das Profil gar nicht, unser Issue verlangte das Gegenteil. Mehr dazu bei den Schwerpunkten (Testdesign)

ArchiLab

Zutaten-IDs: kitchen-storage kennt die Rezepte nicht

Was wir sehen

- menu-ordering leitet die Zutaten-ID aus dem Namen ab (`Ingredient.idFor`)
- kitchen-storage startet mit eigenen, festen IDs (`InitialIngredients`)
- kitchen-storage lehnt Rezepte mit unbekannten Zutaten ab
- ADR 0008 von menu-ordering (19.11.) beschreibt genau das, umgesetzt ist es noch nicht

Warum das ein Problem ist

- MenuChanged wird verworfen, die Küche kennt kein Gericht
- jedes OrderPlaced endet in `UnknownDish`
- es wird nie gekocht, also auch kein OrderFulfilled, keine Box

Wie man es fixt

- für heute: kitchen-storage übernimmt in `appsettings.Development.json` die IDs, die menu-ordering aus dem Namen ableitet
- Gerichte in menu-ordering dann nur mit genau diesen Zutatennamen anlegen
- danach wie im ADR: menu-ordering liest `IngredientListChanged`, die abgeleiteten IDs fallen weg

ArchiLab

Wer braucht wen bis 12:30?

Wer	mit wem	was
Stefan	alle	local-dev-env auf master: erledigt. Jetzt überall pullen und `down -v`
menu-ordering	reservation	!16 mergen, einen CheckIn durchschicken
menu-ordering	kitchen-storage	Zutaten-IDs angleichen, ein Gericht anlegen, MenuChanged kommt an
kitchen-storage	pickup	Boxen lernen, OrderFulfilled, Box leeren, BoxEmptied
reservation		Start im Docker-Container: dev nach main mergen (Profil `kafka` im Container, JTE-Fix). Sonst heute aus der IDE mit Profil `kafka` starten
alle		einmal der ganze Happy Path an einem Rechner, vor 12:30

> 14:00 Retrospektiven: kitchen-storage bitte bis dahin fertig

ArchiLab

10:30 bis 12:30

Teamarbeit

Der Happy Path läuft über alle Teams



ArchiLab

Teamarbeit bis 12:30

- Ziel: Der Happy Path läuft über alle Teams, vom QR-Code am Sitzplatz bis zur geöffneten Abholbox
- alle Teams im selben Raum
- Fragen zwischen den Teams direkt klären, nicht über Discord
- was auf main ist, zählt: Merge Requests heute noch mergen
- Zwischenstand um 11:30, kurz im Plenum

ArchiLab

13:15 bis 14:00

Vorführung des Walking Skeletons

Vom QR-Code am Sitzplatz bis zur geöffneten Abholbox



ArchiLab

Vorführung des Walking Skeletons

reservation	Gast scannt den QR-Code am Sitzplatz und checkt ein: CheckIn
menu-ordering	Gast bestellt ein Gericht: OrderPlaced
kitchen-storage	Küche kocht, Schritt für Schritt, und stellt das Essen in eine Box: OrderFulfilled
pickup	Box wird gefüllt, Gast öffnet und leert sie: BoxEmptied
kitchen-storage	Box ist wieder frei

> Jedes Team zeigt seinen Schritt am eigenen Client. Was nicht läuft, sagen wir offen, und es geht in die Retrospektive

ArchiLab

14:00 bis 15:00

Reflexion des Grundlagenteils

Die Retrospektiven der Teams



Wie wir die Retrospektiven nutzen

- jedes Team stellt seine Retrospektive vor, etwa 15 Minuten
- **die anderen Teams schreiben währenddessen Karten:**
  Was ist mir unklar?
  Was möchte ich vertiefen?
- ein Gedanke pro Karte, groß geschrieben
- die Karten kommen danach an die Tafel

ArchiLab

15:00 bis 15:30

Die Schwerpunkte des Projektteils

Karten an die Tafel, Dot Voting



ArchiLab

Die Schwerpunkte des Projektteils

- 15:00 bis 15:30
- die Karten der Teams an die Tafel, ähnliche fassen wir zusammen
- dazu unsere eigenen Karten: Themen, die uns in Ihrem Code aufgefallen sind
- jede:r verteilt drei Punkte (Dot Voting)
- die meisten Punkte setzen die Schwerpunkte ab dem 10.12.

Unsere Karten

Events, die voneinander abhängen
MenuChanged vor OrderPlaced: was, wenn die Reihenfolge nicht stimmt?

Der Stand der anderen
Wie kommt ein Service an Daten, die ein anderer besitzt?

Zustellgarantien End-to-End, Teil 1
Duplikate: Outbox und idempotente Consumer

Zustellgarantien End-to-End, Teil 2
Eine Nachricht, die nie verarbeitet werden kann

Testdesign
Was ist der Normalfall, was das Double?

Pipeline-Design
Was läuft wann, wo, gegen welchen Broker?

Wem gehört ein Topic?
zentral in local-dev-env oder beim Producer?

KI: erst entscheiden, dann schreiben
Wer trifft die Entscheidung, wer schreibt den ADR?

KI: Definition of Done
Wann ist etwas fertig, für Menschen und für den Agenten?

ArchiLab

Events, die voneinander abhängen

Was wir gesehen haben

- MenuChanged auf `bl.menu.events.v1`, OrderPlaced auf `bl.order.events.v1`
- kitchen-storage liest jedes Topic mit eigenem Listener, also parallel
- Kafka garantiert die Reihenfolge nur innerhalb einer Partition
- OrderPlaced vor dem Rezept: `UnknownDish`, Record übersprungen, Bestellung verloren

Wann das passiert

- im laufenden Betrieb selten: Gerichte gibt es lange vor der ersten Bestellung
- sicher nach einer Pause: beide Listener arbeiten den Rückstand ab, Reihenfolge ist Zufall
- wenn das Rezept abgelehnt wurde (heute: Zutaten-IDs)
- bei einer neuen Instanz mit leerer Datenbank

> "Ein Topic pro Aggregate" und "ein Listener pro Topic" enthalten eine Entscheidung über Reihenfolge, die niemand bewusst getroffen hat

ArchiLab

Vier Antworten der Architektur

Antwort	wie	dafür	dagegen
1. Consumer wartet	Bestellung als "wartet auf Rezept" parken, verarbeiten, sobald das Rezept kommt	fachlich sauber, nichts geht verloren	Consumer wird komplexer, braucht Zeitlimit
2. Event trägt alles	OrderPlaced enthält das Rezept oder einen Verweis mit Version	Abhängigkeit verschwindet	fetteres Event, Kopplung an das Datenmodell des Producers
3. erst einlesen, dann verarbeiten	Rezepte als State Transfer auf einem compacted Topic, beim Start vollständig lesen, dann erst Bestellungen	lokale Kopie, keine Laufzeitkopplung	Start dauert, löst nur den Start, nicht den laufenden Betrieb
4. synchron nachfragen	bei unbekanntem Gericht das Rezept per REST bei menu-ordering holen	einfach, immer aktuell	Laufzeitkopplung: menu-ordering muss erreichbar sein

> Keine Antwort ist umsonst. Welche passt, hängt davon ab, wie oft der Fall eintritt und was ein Fehler fachlich kostet

ArchiLab

Der Stand der anderen

- menu-ordering braucht die Zutaten von kitchen-storage, kitchen-storage die Boxen von pickup
- **heute zwei Lösungen im Code nebeneinander:**
  kitchen-storage kündigt bei jedem Start alle Zutaten neu an
  pickup meldet seine Boxen genau einmal, beim ersten Start
- Kafka löscht Events nach der Retention, Standard 7 Tage
- ein Service, der später dazukommt, findet den Anfang nicht mehr
- ADR 0008: State Transfer (`db.*.ecst.v1`), bisher von keinem Team genutzt

> Kafka hat dafür eine eigene Antwort: Log Compaction. Die Varianten auf der nächsten Folie

ArchiLab

Der Stand der anderen: vier Varianten

Variante	wie	dafür	dagegen
State Transfer	compacted Topic, Consumer hält lokale Kopie, ein neuer liest einmal ab Offset 0	keine Laufzeitkopplung, neuer Consumer findet den Stand	Kopie hinkt hinterher, bis die Nachricht verarbeitet ist. Consumer bauen auf der Struktur der Entity auf: der Producer kann sein Modell nicht mehr frei ändern
Snapshot plus Events	einmal per API den Stand holen, dann den Events folgen	kein compacted Topic nötig	Snapshot muss sagen, ab welcher Event-Position er gilt. Sonst fehlen Änderungen dazwischen, oder alte Events überschreiben neue Daten. Beim Start muss der Besitzer erreichbar sein
synchron abfragen	bei Bedarf beim Besitzer nachfragen, oft mit Cache	einfach, aktuell	Laufzeitkopplung: ist der Besitzer nicht erreichbar, kann der Consumer nicht arbeiten. Ein Cache mildert das, kann aber veralten
neu ankündigen	Producer sendet seinen Stand bei jedem Start oder auf Anfrage	schnell gebaut	Workaround: Last bei jedem Start, Consumer bekommt alles noch einmal. Ein Consumer, der später dazukommt, wartet auf den nächsten Start

> Log Compaction: Kafka behält pro Key mindestens die letzte Nachricht, unbegrenzt. Kein "alle Events für immer", sondern der letzte Stand pro Entity

ArchiLab

Zustellgarantien End-to-End

Teil 1: Duplikate

- Outbox-Relay sendet, stürzt ab, bevor es "gesendet" vermerkt: sendet erneut
- Producer bekommt keine Bestätigung, sendet erneut
- Consumer verarbeitet, stürzt vor dem Commit des Offsets ab: bekommt den Record noch einmal
- also: at-least-once, der Consumer muss idempotent sein

Teil 2: Nachrichten, die nie gehen

- Poison Message: unlesbar, oder fachlich nicht verarbeitbar
- heute bei kitchen-storage: loggen, überspringen, committen
- Alternativen: Blockieren, Retry mit Backoff, Dead Letter Topic
- dafür erst unterscheiden: transienter oder permanenter Fehler?

Die Outbox sorgt dafür, dass nichts verloren geht. Dafür kommen Duplikate. Teil 1 und Teil 2 sind zwei Seiten derselben Garantie

ArchiLab

Testdesign und Pipeline-Design

Testdesign: was auf welcher Ebene?

- Unit-Tests mit Test-Doubles
- Komponententest mit echtem Broker: Testcontainers oder local-dev-env?
- Contract-Tests zwischen Producer und Consumer
- End-to-End über den Happy Path
- Was ist der Normalfall, was das Double? `@Profile("!kafka")` startet ohne Broker, sendet aber nichts

Pipeline-Design: wann und wo?

- Merge Request oder main, was läuft wo?
- Stages: Build, Tests, Contract, Image, Deployment
- Tests mit echtem Broker: Runner mit Docker?
- Ist die AsyncAPI-Validierung schon ein Contract-Test?
- Event-Katalog aktualisieren

> Die Tests mit echtem Broker liegen auf der Grenze: entscheiden im Testdesign, umsetzen in der Pipeline

ArchiLab

Wem gehört ein Topic?

Zentral, wie heute

- alle Topics in `init.sh` von local-dev-env
- ein Blick auf alles, eine Stelle für Namen, Partitionen, Compaction
- aber: jedes Team ändert dieselbe Datei, heute drei Branches mit Konflikten

Beim Producer

- Service legt seine Topics beim Start selbst an
- oder: Deklaration im Service-Repo, local-dev-env sammelt ein
- oder: Topics aus den AsyncAPI-Contracts erzeugen
- klare Ownership, aber Governance wird schwerer

> Lose Kopplung spricht für den Producer. Einfacher wird local-dev-env dadurch nicht, die Komplexität verschiebt sich in das Einsammeln

KI im Projektteil

Erst entscheiden, dann schreiben

Das Team entscheidet, der ADR hält es fest und ist dann Input für den Agenten. Ein Entwurf vom Agenten bleibt "proposed", bis das Team ihn durchgesehen hat

Definition of Done

Fertig ist, was über einen Merge Request auf main ist, mit grüner Pipeline, Contract und Katalog aktuell, Issue geschlossen. Lesbar für Menschen und für den Agenten

Kontext für den Agenten

Was steht zentral im Workspace, was im Service-Repo? CLAUDE.md, Konventionen, Verweise auf ADRs

Leitplanken

Was darf der Agent nicht? Hooks, geschützte Branches, Freigaben

Zwei davon als Karten an der Tafel. Kontext und Leitplanken sind ohnehin Thema am 10.12.

ArchiLab

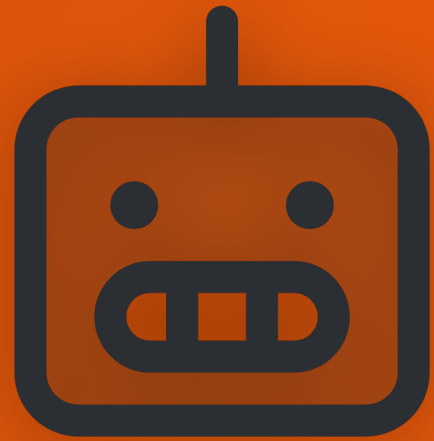
Was einfach erledigt werden muss

- Zutaten nach ADR 0008 von menu-ordering umsetzen
- CheckIn in der Vergangenheitsform, abgestimmt mit menu-ordering
- Tests auf jedem Merge Request und auf main
- erledigte Issues schließen, Doku-Branches mergen
- **ADR 0007 von menu-ordering: beides ja**
  der Gast sieht, wie weit sein Essen ist und wie lange es noch dauert
  Gerichte ohne verfügbare Zutaten kann man nicht bestellen

15:30 bis 15:50

Kick-off des Projektteils

Wie das Team kitchen-storage mit dem Agenten arbeitet

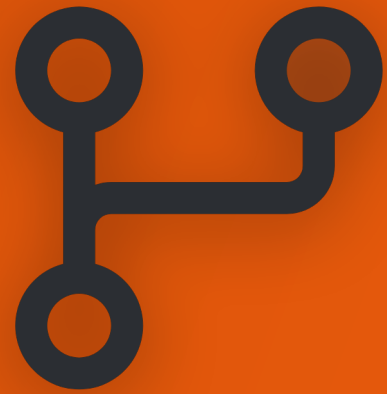


ArchiLab

15:50 bis 16:30

Inhaltsimpuls: Git-basiertes Arbeiten im Team

Wie ein kleines Team am selben Service arbeitet



ArchiLab

16:30 bis 17:00

Ausblick und Hausaufgabe

Wie es im Projektteil weitergeht



ArchiLab

Ausblick

10.12.	Eventual Consistency und Robustheit
17.12.	Integrationstag
05.01.	Abschluss der Restaurant-Fallstudie und Kick-off Microservice Dungeon
21.01.	Fortgeschrittene DevOps- und Hosting-Konzepte, Infrastructure as Code
04.02.	Monitoring mit Prometheus und Grafana
18.02.	Abschluss-Codefight und Reflexion

ArchiLab

Hausaufgabe bis 10.12.

- folgt am Ende des Tages, als Issue je Team



ArchiLab

Backup

Backup

Vertiefung auf Nachfrage: Fehlerbehandlung, Stand der anderen, Issues je Team



ArchiLab

Teil 2: Blockieren

Was passiert

- Consumer committet den Offset nicht
- versucht denselben Record immer wieder
- bis jemand den Fehler behebt: Code ändern, Daten korrigieren

Was es kostet

- die ganze Partition steht still (Head-of-Line Blocking)
- ein kaputtes MenuChanged hält alle weiteren Menüänderungen auf
- ohne Alerting merkt es niemand

Wann es passt

- Reihenfolge ist wichtiger als Durchsatz
- nichts darf verloren gehen
- Fehler sind selten und schnell behebbar

ArchiLab

Teil 2: Retry mit Backoff

Was passiert

- einige Versuche mit wachsender Wartezeit: 1 s, 2 s, 4 s, ...
- für transiente Fehler: Datenbank kurz weg, Nachbar antwortet nicht, Rezept noch nicht da
- danach: Dead Letter Topic oder überspringen

Zwei Varianten

- blockierend: die Partition wartet während des Backoffs
- nicht blockierend: Nachricht wandert in Retry-Topics (`…-retry-1`, `…-retry-2`), das Haupt-Topic läuft weiter
- nicht blockierend gibt die Reihenfolge auf

Wann es passt

- der Fehler geht wahrscheinlich von selbst weg
- in Spring Kafka: `DefaultErrorHandler` mit `BackOff` (blockierend), `@RetryableTopic` (nicht blockierend)

Teil 2: Dead Letter Topic

Was passiert

- Nachricht landet auf einem eigenen Topic, mit Fehler-Headern: Exception, Topic, Partition, Offset des Originals
- Consumer macht mit der nächsten Nachricht weiter
- jemand sieht sie sich an, behebt die Ursache, spielt sie zurück

Was es kostet

- ohne Monitoring und Weg zurück nur ein Papierkorb
- eine spätere Nachricht zur selben Entity kann die geparkte überholen

Wann es passt

- permanente Fehler, die ein Mensch klären muss
- üblich als letzte Stufe: erst Retry mit Backoff, dann Dead Letter Topic

Issues kitchen-storage

Kafka #7	**erledigt**	geschlossen. Lernt die Abholboxen jetzt aus BoxEmptied von pickup, lehnt Box-IDs ab, die keine UUID sind
Test mit Kafka #8	**weitgehend**	MenuChanged und OrderPlaced rein, CookingStepChanged raus, mit Testcontainers. BoxEmptied rein und OrderFulfilled raus fehlen
Doku #10	**weitgehend**	CLAUDE.md und README am 25.11. aktualisiert. Clients-Seite liegt noch auf Branch clients-page in documentation
Retro #13	**offen**	auf Branch reflexion nur eine leere Datei, auf main nichts

> #9, #11, #12 geschlossen. Für heute: Zutaten-IDs mit menu-ordering, Boxen mit pickup einmal durchspielen

ArchiLab

Issues menu-ordering

Kafka #10	**teilweise**	auf main liest der Service CheckIn noch von `reservations`. Happy Path nicht gespielt
CheckIn geändert #18	**weitgehend**	Fix auf Branch: Topic `bl.reservation.events.v1`, Typ `reservation.check-in`. Merge Request !16, Pipeline grün, nicht gemergt
Zutaten ADR 0008	**offen**	entschieden am 19.11.: IDs von kitchen-storage, über IngredientListChanged. Noch nicht umgesetzt
Retro #19	**erledigt**	ausführlich, auf main

> Alle Hausaufgaben-Issues geschlossen. Für heute: !16 mergen, Zutaten-IDs mit kitchen-storage

ArchiLab

Issues pickup

ADR 0008 #13	**erledigt**	Zahlen-Header jetzt binär, ein Test prüft die Bytes
Doku #11	**weitgehend**	CLAUDE.md auf main. Katalogseite und Context Map auf Branch pickup-overview-and-context-map in documentation, nicht gemergt
ADRs #12	**erledigt**	sieben ADRs, laut Commit KI-gestützt
Pipeline #15	**erledigt**	`run-tests` bei jedem Push
local-dev-env pickup_config	**erledigt**	Topic `bl.box.events.v1` und Datenbank `pickup`, seit heute 9:08 auf master
Retro #14	**erledigt**	auf main, Issue noch offen

ArchiLab

Issues reservation

Kafka #11	**weitgehend**	CheckIn nach ADR 0008 über die outbox-lib, Key Tisch-ID. Topic und Datenbank seit heute 9:08 in local-dev-env. OrderPlaced kommt an, der Handler ist leer
REST-Tests #13	**weitgehend**	MockMvc für Gast, Walk-in, Check-in
Doku #14	**offen**	kein CLAUDE.md, Katalogseite nur der Contract
ADRs #15	**erledigt**	0001 und 0002, beide accepted. Issue noch offen
Pipeline #18	**teilweise**	`mvn verify` nur bei Merge Requests mit Änderung in `src/`, nicht auf main
Datenbank #19	**weitgehend**	Profil `kafka` mit PostgreSQL, Datenbank seit heute 9:08 in local-dev-env

> Sechs Commits nur auf dev (Stand 9:33), darunter Profil `kafka` im Container und der JTE-Fix von 23:51. Für den Start im Container: dev nach main

ArchiLab