

**YOUR OFFLINE-FIRST APP
IS NOT A SYNC QUEUE.**

YOUR APP TESTIFIES.
IT DOESN'T RULE.



- 1 Why a queue isn't enough
- 2 Recording reality
- 3 Witness vs Judge, in code
- 4 One module, three runtimes
- 5 Conflict as a decision
- 6 What it cost
- 7 Takeaways

SEVEN STOPS · AND ONE EVENT MAKING ALL OF THEM.

TAP · RECORD · PAINT · UPLOAD · JUDGE · RECONCILE · CONFIRMED

01 ———

WHY A QUEUE ISN'T ENOUGH

A queue is the obvious answer. Watch it lose.

TAP the station we enter

TAP · RECORD · PAINT · UPLOAD · JUDGE · RECONCILE · CONFIRMED

THE SILENT CLOBBER

BOTH OFFLINE. BOTH RIGHT.

tank 12 · 1,000 kg

PAULA docket 41 · +300 · ~~now 1,300~~

BEN docket 52 · +400 · now 1,400

tank 12 · weight_kg = 1,400

| Paula · 14:03 · docket 41 → tank 12 ← OUR EVENT

TAP · RECORD · PAINT · UPLOAD · JUDGE · RECONCILE · CONFIRMED

WITNESS · JUDGE

CRDT?

The client records what happened.

Paula · 14:03 · docket 41 → tank 12

It never rules on it.

02

RECORDING REALITY

An immutable event log, not a table of current values.

RECORD

Paula's tap becomes a row

TAP · **RECORD** · PAINT · UPLOAD · JUDGE · RECONCILE · CONFIRMED

IMMUTABLE

NO FOREIGN KEYS

CLIENT MINTED

THE EVENT

<code>operational_event</code>	
<code>id</code>	client UUID
<code>event_type</code>	OperationalEventType
<code>event_data</code>	JSON – the intent
<code>version_snapshot</code>	what the user saw
<code>device_timestamp</code>	when it happened
<code>processing_state</code>	server-managed

OUR EVENT

DOCKET_ASSIGN
41 → tank 12
14:03

TAP · **RECORD** · PAINT · UPLOAD · JUDGE · RECONCILE · CONFIRMED

IMMUTABLE

NO FOREIGN KEYS

CLIENT MINTED

IMMUTABLE

We never edit what an event says.
We never delete one.

Wrong? Append a correction. Define
reality forward.

```
docket 41 → tank 12  
docket 41 → renamed  
docket 41 → correction
```

OUR EVENT

—

DOCKET_ASSIGN

41 → tank 12

—

14:03

—

TAP · **RECORD** · PAINT · UPLOAD · JUDGE · RECONCILE · CONFIRMED

IMMUTABLE

NO FOREIGN KEYS

CLIENT MINTED

NO FOREIGN KEYS

event_data · docket 41 →

~~FOREIGN KEY~~

The intent must be recordable before
the result exists.

OUR EVENT

—

DOCKET_ASSIGN

41 → tank 12

—

14:03

—

↑ event_data ·
docket 41

TAP · **RECORD** · PAINT · UPLOAD · JUDGE · RECONCILE · CONFIRMED

IMMUTABLE

NO FOREIGN KEYS

CLIENT MINTED

CLIENT-MINTED IDENTITY

```
DocketAssignEvent(  
  docketId = docketId,  
  vesselId = vesselId,  
  wineId    = Uuid.random().toString(),  
) : WineryEvent
```

```
data class OperationalEvent(  
  val id: String,  
  val eventType: OperationalEventType,  
  val eventData: WineryEvent,  
)
```

OUR EVENT

39b1...

DOCKET_ASSIGN

41 → tank 12 + W

—

14:03

—

↑ id · wine W

TAP · **RECORD** · PAINT · UPLOAD · JUDGE · RECONCILE · CONFIRMED

03 ———

WITNESS VS JUDGE, IN CODE

Two engines, two writes: one derives, one records.

PAINT the screen updates

UPLOAD the event leaves

TAP · RECORD · **PAINT** · UPLOAD · JUDGE · RECONCILE · CONFIRMED

TWO WRITES FOR EVERY TAP

Recording engine → the event

ALWAYS SUCCEEDS · THE WITNESS ·
UPLOADED

Processor engine → the state

DERIVES ROWS · THE LOCAL JUDGE ·
NEVER LEAVES

OUR EVENT

39b1...

DOCKET_ASSIGN

41 → tank 12 + W

—

14:03

PENDING

↑ processing_state

TAP · RECORD · **PAINT** · UPLOAD · JUDGE · RECONCILE · CONFIRMED

ONLY THE EVENT GOES UP

```
tap "Assign"  
  └─ record() → event → server  
  └─ process() → rows x never uploaded
```

```
phone ← rows ← server
```

OUR EVENT

39b1...

DOCKET_ASSIGN

41 → tank 12 + W

—

14:03

PENDING

TAP · RECORD · PAINT · **UPLOAD** · JUDGE · RECONCILE · CONFIRMED

WHO CLEANS UP? NOBODY.

0 lines of cleanup code

The sync layer re-derives every local row from the server's.

OUR EVENT

39b1...

DOCKET_ASSIGN

41 → tank 12 + W

—

14:03

PENDING

TAP · RECORD · PAINT · UPLOAD · JUDGE · **RECONCILE** · CONFIRMED

ONE EVENT, END TO END

- 1 tap → both ids minted, event written, tank painted
- 2 event uploads — only the event
- 3 server runs the **same** processor → authoritative W
- 4 authoritative W syncs down, overwrites the guess

TAP · RECORD · PAINT · **UPLOAD** · JUDGE · RECONCILE · CONFIRMED

ONE MODULE, THREE RUNTIMES

Two of them you already guessed. The third is the point.

JUDGE our event reaches the server

TAP · RECORD · PAINT · UPLOAD · **JUDGE** · RECONCILE · CONFIRMED

THE JUDGE AND THE WITNESS ARE THE SAME CODE.

OUR EVENT

39b1...

DOCKET_ASSIGN

41 → tank 12 + W

—

14:03

PENDING

Mobile ← Kotlin → JS → npm → private registry → Node

TAP · RECORD · PAINT · UPLOAD · **JUDGE** · RECONCILE · CONFIRMED

ONE MODULE, THREE TARGETS

```
plugins {  
    alias(libs.plugins.kmp.library)  
}  
kotlin.js(IR) {  
    nodejs()  
    binaries.library()  
    generateTypeScriptDefinitions()  
}
```

← you already
have this

← everything
we added

64 event types
0 reimplemented

@strvcom/winery-judge@1.1.0-35507a2f

TAP · RECORD · PAINT · UPLOAD · **JUDGE** · RECONCILE · CONFIRMED

TWO IMPLEMENTATIONS ALWAYS DRIFT

Every mobile+server team writes
each rule twice.

Two copies drift. Not **if** — **when**.

one rule ➔ client copy ➤ server copy

OUR EVENT

39b1...

DOCKET_ASSIGN

41 → tank 12 + W

—

14:03

PENDING

@strvcom/winery-judge@1.1.0-35507a2f

TAP · RECORD · PAINT · UPLOAD · **JUDGE** · RECONCILE · CONFIRMED

THE SERVER PIPELINE

```
const judge = new WineryJudgeJs(db)

judge.validateEvent(type, ver, data)
await judge.validateVersionSnapshot(
  snapshot
)
await judge.processEvent(id, ws, ...)
```

OUR EVENT

39b1...

DOCKET_ASSIGN

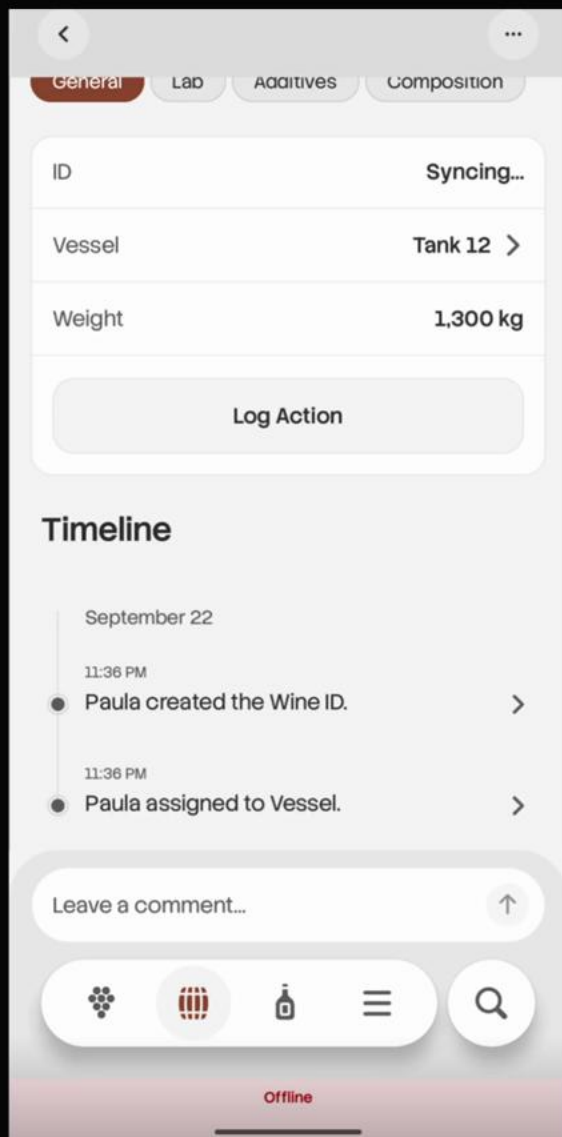
41 → tank 12 + W

—

14:03

PENDING

TAP · RECORD · PAINT · UPLOAD · **JUDGE** · RECONCILE · CONFIRMED



OUR EVENT

39b1... HARVEST DOCKET ASSIGN
docket D-0041 → Tank 12 · 300 kg
snapshot docket v2 · vessel v6

- 1 validate no errors
- 2 check snapshots no conflicts
- 3 process PROCESSED

the judge asked the database for
operational_event IN_PROCESSING
wine 1,300 kg
wine_lineage 1,000 kg
wine 0 kg v+1
vessel → wine W-D540 v+1
operational_event CELLAR_WINE_CREATED
docket 1,900 kg v+1
docket_wine 300 kg
operational_event PROCESSED

same package the phone runs · 9 writes

\W > █

TAP · RECORD · PAINT · UPLOAD · **JUDGE** · RECONCILE · CONFIRMED

05

CONFLICT AS A DECISION

Most conflicts aren't. The real ones get an author.

RECONCILE the rail forks · Ben gets a card

TAP · RECORD · PAINT · UPLOAD · **JUDGE · RECONCILE** · CONFIRMED

BACK TO PAULA AND BEN

```
versionSnapshot = listOf(  
    fetchVersion("docket", docketId),  
    fetchVersion("vessel", vesselId),  
)
```

A record of the reality the user
saw.

OUR EVENT

39b1...

DOCKET_ASSIGN

41 → tank 12 + W

docket v2

vessel v6

14:03

PENDING

↑ processing_state

TAP · RECORD · PAINT · UPLOAD · **JUDGE** · RECONCILE · CONFIRMED

THE WORLD CHANGED UNDERNEATH

Paula · vessel v6 · server v6 → **apply**

Ben · vessel v6 · server v7 → **mismatch**

OUR EVENT

39b1...

DOCKET_ASSIGN

41 → tank 12 + W

docket v2

vessel v6

14:03

PROCESSED

BEN'S EVENT

b19c...

DOCKET_ASSIGN

52 → tank 12

docket v1

vessel v6

14:05

CONFLICT

TAP · RECORD · PAINT · UPLOAD · **JUDGE** · RECONCILE · CONFIRMED

A MISMATCH ISN'T A CONFLICT

```
interface WineryEvent {  
    fun modifiedAttributes(): Map<..>  
    fun touchedEntities(): Map<..>  
}  
  
rename → vessel { name }  
assign → vessel { occupant_wine_id }
```

↑ auto-resolve

BEN'S EVENT

b19c...

DOCKET_ASSIGN

52 → tank 12

docket v1

vessel v6

14:05

CONFLICT

TAP · RECORD · PAINT · UPLOAD · **JUDGE** · RECONCILE · CONFIRMED

THE BRANCH

version mismatch

- |— disjoint columns → apply
- | → CONFLICT_RESOLVED_AUTOMATICALLY
- └ real overlap → human
- CONFLICT_RESOLUTION_REQUESTED

BEN'S EVENT

b19c...

DOCKET_ASSIGN

52 → tank 12

docket v1

vessel v6

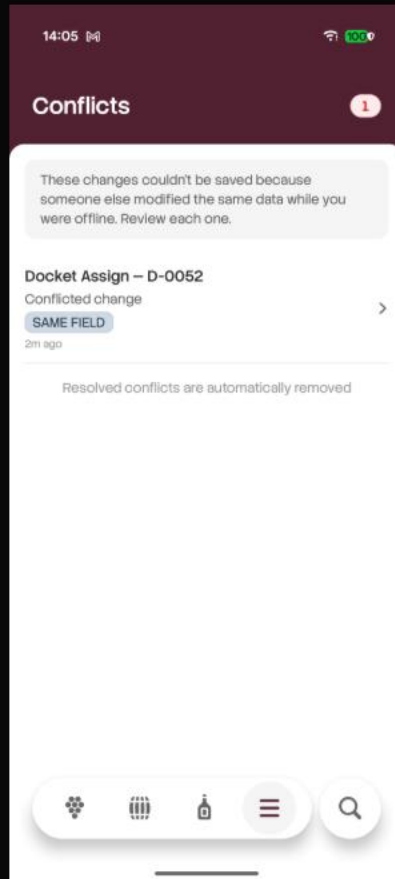
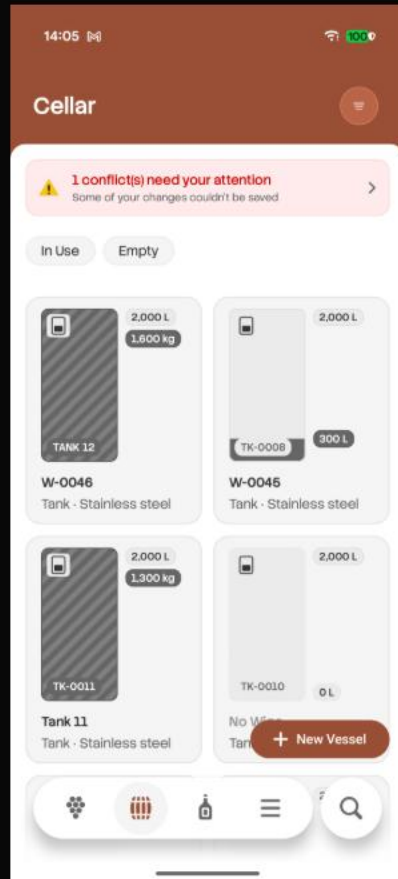
14:05

CONFLICT

↑ the escalation

TAP · RECORD · PAINT · UPLOAD · JUDGE · **RECONCILE** · CONFIRMED

THE CONFLICT, ON SCREEN



SAME FIELD / VOLUME CONFLICT

DUPLICATE

ENTITY CHANGED

three kinds, three ways out

TAP · RECORD · PAINT · UPLOAD · JUDGE · **RECONCILE** · CONFIRMED

SAME FIELD

ENTITY CHANGED

DUPLICATE

DEFINE REALITY FORWARD



Conflict: Tank 12

UPDATE & RETRY

DISCARD

Both original events stay in the log. Forever.

TAP · RECORD · PAINT · UPLOAD · JUDGE · RECONCILE · **CONFIRMED**

SAME FIELD

ENTITY CHANGED

DUPLICATE

DEFINE REALITY FORWARD

BEN'S EVENT

b19c...

DOCKET_ASSIGN

52 → tank 12

docket v1

vessel v6

14:05

~~CONFLICT~~

PROCESSED

A conflict becomes a decision,
with an author and a timestamp.

↑ resolves b19c...

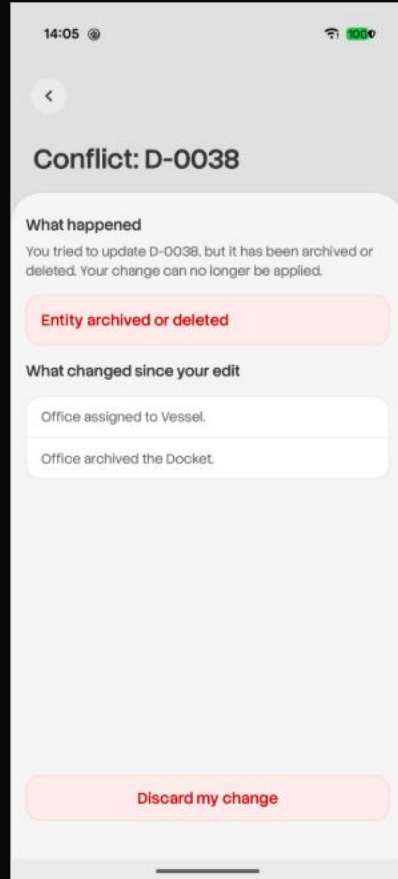
TAP · RECORD · PAINT · UPLOAD · JUDGE · RECONCILE · **CONFIRMED**

SAME FIELD

ENTITY CHANGED

DUPLICATE

WHEN THE THING YOU TOUCHED IS GONE



ENTITY CHANGED · DOCKET 38

The work is valid. Its subject isn't.

DISCARD

NO UPDATE & RETRY

There's nothing to retry against.

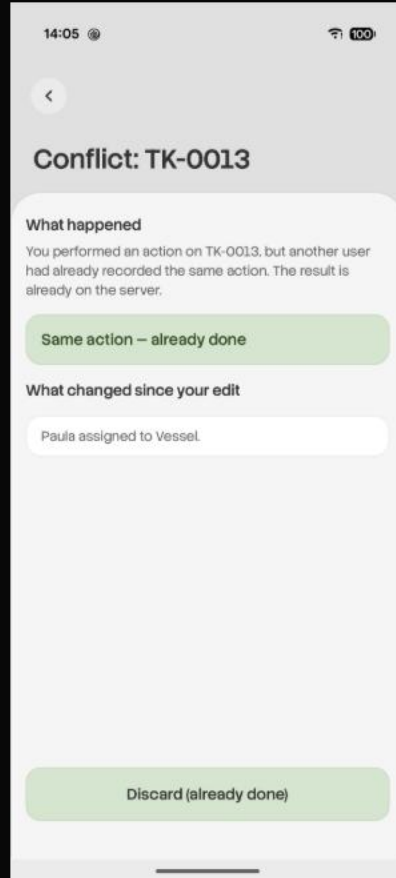
TAP · RECORD · PAINT · UPLOAD · JUDGE · RECONCILE · CONFIRMED

SAME FIELD

ENTITY CHANGED

DUPLICATE

WHEN THE WORK IS ALREADY DONE



DUPLICATE · DOCKET 45

The work is valid. It already happened.

DISCARD

NO UPDATE & RETRY

Somebody else already did it.

TAP · RECORD · PAINT · UPLOAD · JUDGE · RECONCILE · CONFIRMED

CRDT?

Our event log already is a CRDT.

APPEND-ONLY · IMMUTABLE · MERGES BY UNION

TAP · RECORD · PAINT · UPLOAD · JUDGE · RECONCILE · **CONFIRMED**

CRDT?

What we refused to make convergent: **the projection.**

TANK 12 → ONE WINE

PICK ONE	the other's grapes vanish
ADD THE KILOS	400 kg under the wrong docket
KEEP BOTH	the next sample can't say which
NONE OF THOSE	a new wine. No merge can vouch for it.

TAP · RECORD · PAINT · UPLOAD · JUDGE · RECONCILE · **CONFIRMED**

CRDT?

When all you need is the result

TEXT · WHITEBOARD · SHOPPING LIST

MERGE IT

When the testimony is the product

WINE · STOCK · ACCOUNT

RECORD IT – WITH SOMEONE'S NAME ON IT

TAP · RECORD · PAINT · UPLOAD · JUDGE · RECONCILE · **CONFIRMED**

06

WHAT IT COST

One limit we chose. One toll we pay.

THE RAIL where it broke

TAP · RECORD · PAINT · UPLOAD · ~~JUDGE~~ · RECONCILE · CONFIRMED

WE DIDN'T EVENT-SOURCE EVERYTHING

- the ledger → entities whose history is the product
- plain CRUD → creating and deleting catalog rows

Traceability has to earn its cost.

TAP · RECORD · PAINT · UPLOAD · ~~JUDGE~~ · RECONCILE · CONFIRMED

THE TAX NOBODY BLOGS ABOUT

“Zero duplicated logic” — true for the rules. Not free at the type boundary.

- **suspend** → Promise gymnastics

```
try { process(event) }  
catch (e: Throwable) {  
    withContext(NonCancellable) {  
        nestJs.rollback().await()  
    }  
}
```

TAP · RECORD · PAINT · UPLOAD · **JUDGE** · RECONCILE · CONFIRMED

THE TAX NOBODY BLOGS ABOUT

“Zero duplicated logic” — true for the rules. Not free at the type boundary.

- Long → Double
- suspend → Promise gymnastics



TAP · RECORD · PAINT · UPLOAD · ~~JUDGE~~ · RECONCILE · CONFIRMED

07 ———

TAKEAWAYS

Five things to carry home.

CONFIRMED cracks repaired · both events landed

TAP · RECORD · PAINT · UPLOAD · JUDGE · RECONCILE · **CONFIRMED**

- 1 Store events, not mutations.
 - 2 Let the client witness. Judge where you see all.
 - 3 Mint the id — and make it travel in the payload.
 - 4 Record what the user saw. Let disjoint edits settle.
 - 5 Ship each rule once. Compile it to your server.
-

Apps that behave the same on a plane as at a desk.

no lost writes · no silent divergence · conflicts decided

TAP · RECORD · PAINT · UPLOAD · JUDGE · RECONCILE · CONFIRMED

YOUR APP TESTIFIES. IT DOESN'T RULE

BLOG



THANK YOU!

BLOG

