

Wie wir das gebaut haben

Ein Blick unter die Haube – von FileMaker zum Echtzeit-Dispoplan.

Sie haben Dispoplan gerade live gesehen – jetzt die Technik dahinter.

Alexander Lemuth · ALESO SOLUTIONS

DAS GRUNDPRINZIP

FileMaker bleibt die Wahrheit

Die 30 Jahre alte FileMaker-Lösung bleibt das **führende System (System of Record)**. Alles andere ist ein schneller, jederzeit wegwerfbarer Spiegel.



FileMaker führt

Drei Datenbanken halten Studenten, Projekte und den Dispo-Plan – wie seit jeher.



Der Spiegel liefert Tempo

MongoDB & Redis spiegeln die Daten für sofortige Ladezeiten und Echtzeit-Bearbeitung.

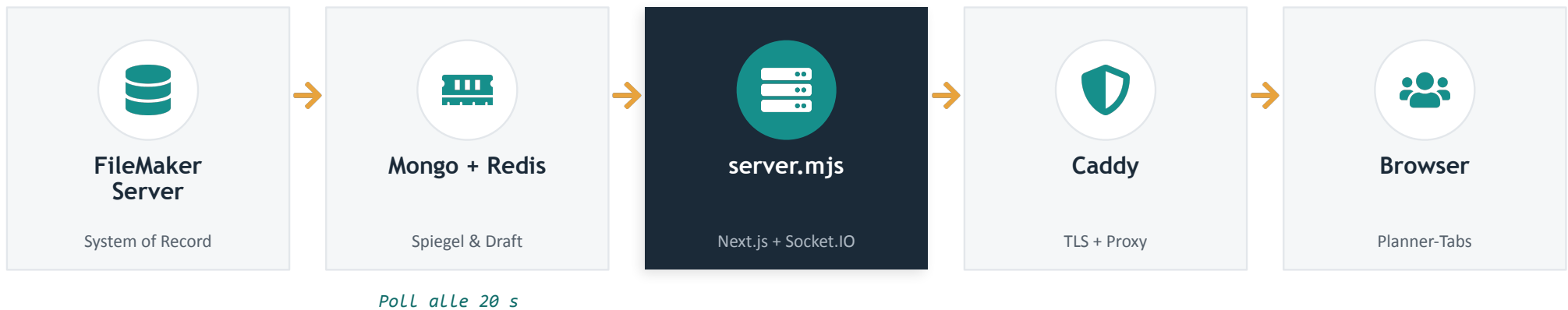


Zurück nach FileMaker

Fertige Änderungen werden gebündelt zurückgeschrieben – FileMaker bleibt aktuell.

ARCHITEKTUR

Der Datenfluss



Ein einziger Node-Prozess bedient App + Echtzeit-Schicht. FileMaker wird nur über die Data API erreicht – nie direkt aus dem UI.

TECHNOLOGIE-STACK

Was verbaut ist

Runtime & Framework

Next.js 16.3.5

React 19.2.4

TypeScript 5

Node 22

Server & Echtzeit

Socket.IO 4.8.3

ioredis 5.10.1

MongoDB 7.1.1

fm-data-api 3.0.1

Frontend

react-virtual 3.13

day-picker 10.0

lucide-react 1.7

jspdf 4.2.1

Security & Infra

jsonwebtoken 9.0.3

Docker –

Caddy latest

GitHub Actions –

DATEN-PIPELINE

Fünf Stufen - von FileMaker zurück nach FileMaker

1 Sync-Loop starten

Beim Serverstart läuft der FileMaker-Poll – auch ohne verbundenen Browser.

2 Pollen & spiegeln

Alle 20 s Geändertes holen, in MongoDB upserten, Löschungen per Diff erkennen.

3 An Browser pushen

Änderungen via Server-Sent Events an jeden offenen Tab senden.

4 Seitenaufruf liest Spiegel

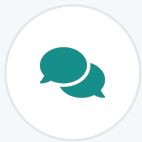
Ein /dispoplan-Load liest MongoDB – nie direkt FileMaker.

5 Bearbeiten & zurückschreiben

Edits leben in Redis; beim Speichern gebündelt (15er-Chunks) nach FileMaker.

Mehrere Disponenten, derselbe Tag

Jeder Tab tritt einem **Socket.IO-Room pro Projekt** bei. Edits gehen sofort an alle im Room und landen in Redis – dauerhaft nach MongoDB erst beim expliziten Save.



plannerUpdate

Live-Delta an den Room – sofort bei allen sichtbar.



Redis-Draft

Nicht gespeicherte Edits liegen flüchtig in Redis.



30-Sek-Sweep

Flusht geänderte Drafts – ein Redis-Neustart verliert nichts.

Konflikte lösen - ohne zu sperren

Statt ein ganzes Projekt zu sperren, führt der Server eine **Besitzer-Karte pro Slot**.



So funktioniert es

- Wer eine Zelle zuletzt bearbeitet, besitzt sie
- Ein Slot wird frei, sobald der Edit auf den FileMaker-Stand zurückgeht
- Besitz wandert zum nächsten Editor – auch nach einem Verbindungsabbruch



Warum das elegant ist

- Kein Sperr-Ballast, der ein ganzes Projekt blockiert
- Keine verwaisten Locks nach einem Verbindungsabbruch
- Feinste Granularität: Konflikt nur auf Zellebene, nicht projektweit

planner:slotowners: – der neueste Editor gewinnt die Zelle.*

EINE BEWUSSTE ENTSCHEIDUNG

Warum ein eigener Server?

Socket.IO muss sich an einen **rohen http.Server** hängen, um den WebSocket-Upgrade zu behandeln – Next.js' eingebauter Handler gibt keinen her.

next start

Der Standard-Handler – bequem, aber kein Zugriff auf den rohen Socket. Kein WebSocket-Upgrade möglich.



server.mjs

Umschließt Nexts Handler in einem eigenen http-Server – Socket.IO hängt sich an, plus zwei interne HTTP-Hooks für den Sync-Loop.

Ein Prozess, zwei Welten: Next.js-Rendering und Echtzeit-Sockets teilen sich denselben Server.

Kein Login - zwei clevere Wege



Disponenten

Einmal-Token, dann Session

- FileMaker gibt beim Öffnen des Links eine einmalige UUID aus
- proxy.ts verbraucht sie (60 s gültig, dann gelöscht)
- Danach ein 12-Stunden-JWT-Cookie – Refresh braucht kein zweites Token



Studenten

Magic-Links, keine Session

- Ein Link per WhatsApp führt direkt zu genau einem Studenten + Schicht
- Schicht bestätigen, Warteliste, Einsatzinfos, abmelden
- Adressiert über opaken Code oder MD5-Hash – kein Login-Schritt

Zwei Zielgruppen, zwei passgenaue Zugänge – beide ohne klassisches Passwort.

Vier Container, ein Push



app

Multi-Stage-Build → schlankes Standalone-Image



redis

Live-Draft-Cache



mongo

Durabler Spiegel



caddy

TLS + Reverse Proxy inkl. WebSocket



Docker Compose orchestriert alle vier · Caddy terminiert TLS für einsatzplanung.nextime.de

CI/CD über GitHub Actions: Build-Check → SSH auf den Server → git pull → docker compose up --build. Ein Push auf main-stable deployt in place.

WAS DAS PROJEKT BRINGT

Aus 30 Jahren FileMaker wird Echtzeit

- ✓ **Sofortige Ladezeiten** — Der Seitenaufruf liest den Spiegel, nie FileMaker direkt – die Oberfläche fühlt sich sofort an.
- ✓ **Mehrere Disponenten parallel** — Gleichzeitiges Bearbeiten desselben Tages, ohne sich zu blockieren.
- ✓ **Studenten mit einem Tap** — Schicht per WhatsApp-Link bestätigen – ohne App, ohne Login.
- ✓ **FileMaker bleibt das Herz** — Kein Systemwechsel, kein Risiko – die bewährte Lösung bleibt führend.

Moderne Schicht, bewährtes Fundament - das Beste aus beiden Welten.



Fragen & Austausch

Gern zeige ich einzelne Teile noch einmal live im Code.

Alexander Lemuth · ALESO SOLUTIONS · www.aleso-solutions.com

Dispoplan für Nextime · einsatzplanung.nextime.de