

Baseform

Ernährung, Training und Erholung in einer App

Eine Progressive Web App fürs iPhone, die drei Datenströme zusammenführt, die sonst in getrennten Apps liegen: was gegessen wird, wie trainiert wird und wie gut die Erholung läuft. Barcode- und QR-Scan mit Nährwerten von Open Food Facts, ein Trainingsplan mit Gewicht pro Satz und Muskel-Heatmap, Schlaf- und Aktivitätsdaten vom Fitbit über Google Health, dazu ein KI-Coach, der mit dem verbleibenden Tagesbudget rechnet. Ohne Server und ohne Konto: alle Daten bleiben auf dem Gerät.

2'275

Zeilen React und JSX

4

externe Schnittstellen, jede mit Rückfallebene

26

Übungsanimationen aus Posendaten

8

Tabellen mit Row Level Security, Schema vorbereitet

4

Abhängigkeiten zur Laufzeit

0

Server, Konten, Tracker

Gabriel Suter · Persönliches Projekt · Konzept, Design, Umsetzung und Betrieb in einer Hand

React 18 · Vite 8 · PWA · ZXing · Google Health API · Anthropic API · Supabase vorbereitet · Stand September 2026

kontakt@gabrielsuter.dev · gabrielsuter.dev · Lebensmitteldaten von Open Food Facts, ODbL

1 Kurzfassung

Baseform ist eine Progressive Web App für das eigene Training: Lebensmittel per Kamera scannen, Sätze und Gewichte im Gym erfassen, Schlaf und Erholung vom Fitbit dazuholen und aus allem zusammen einen Tagesvorschlag rechnen lassen. Sie läuft ohne Server und ohne Konto vollständig im Browser, wird über Safari auf dem Homescreen installiert und funktioniert dort auch ohne Netz. Konzept, Gestaltung, Datenmodell, Umsetzung und Betrieb liegen in einer Hand.

Der Kern in einem Satz

Alle Daten liegen im localStorage des Geräts, und jeder Schreibvorgang führt pro Tag zusammen, statt den ganzen Block zu überschreiben: nur so übersteht der Bestand zwei gleichzeitig geöffnete Fenster.

2 Ausgangslage und Ziel

Das Problem

Wer Körperrekomposition betreibt, also gleichzeitig definierter werden und Muskeln aufbauen will, braucht drei Dinge im Blick: die Kalorien- und Proteinbilanz, das Trainingsvolumen pro Muskel und die Erholung. Gängige Apps decken jeweils nur einen Teil ab und reden nicht miteinander. Dazu kommen konkrete Ärgernisse:

- **Schweizer Produkte fehlen** in vielen Nährwertdatenbanken, oder die Werte sind veraltet.
- **Konto-Zwang und Tracking** für Daten, die sehr persönlich sind: Gewicht, Schlaf, Puls.
- **Kein Netz im Gym**, aber genau dort will man Sätze und Gewichte erfassen.
- **Wiederholte Eingaben**: gleiche Gewichte, gleiche Maschinen-Einstellungen, jede Woche neu eingetippt.

Das Ziel

Eine einzige App, die auf dem iPhone-Homescreen liegt, in unter einer Sekunde startet, ohne Konto auskommt und alle drei Datenströme auf einen gemeinsamen Nenner bringt: das verbleibende Tagesbudget. Konkret sollte sie

- Nährwerte per Kamera-Scan holen, mit Rückfallebene ohne Internet,
- den Trainingsplan mit Ausführungshinweisen und Animationen abbilden und Sätze samt Gewicht protokollieren,
- Schlaf, Schritte, Ruhepuls und HRV vom Fitbit automatisch einlesen,
- und einen Coach anbieten, der aus Restbudget, Training und Erholung konkrete Vorschläge macht.

Das Ergebnis

Baseform erfüllt diese Anforderungen als Progressive Web App mit React. Die App ist für das Hosting auf Vercel eingerichtet und wird über Safari auf dem Homescreen installiert. Sie ist seit Juli 2026 im eigenen Einsatz, vor allem als Trainingsprotokoll, und wurde in mehreren Iterationen erweitert, zuletzt um die Google-Health-Anbindung und eine Persistenzschicht, die Datenverlust bei parallel geöffneten Fenstern verhindert. Als nächster Schritt ist eine Supabase-Synchronisation vorbereitet, deren Datenbankschema bereits vorliegt.

Meine Rolle

- **Konzept:** Funktionsumfang aus dem eigenen Bedarf abgeleitet, priorisiert nach dem, was im Gym und in der Küche tatsächlich anfällt.
- **Design:** eigenes Farbsystem, Kalorienring, Gewichtskurve, Körpergrafiken und 26 Übungsanimationen als handgezeichnetes SVG, ohne CSS-Framework.
- **Frontend:** React 18 mit Hooks, fünf Screens mit gemeinsamem Zustand, kontrollierte Formulare mit Validierung.
- **Daten und Persistenz:** Speicherformat, Merge-Speicherung pro Tag, Migration alter Sicherungen, Backup als JSON, Postgres-Schema mit Row Level Security.
- **Integrationen:** Kamera-Scan mit ZXing, Open Food Facts, OAuth für Google Health, Anthropic-API für den Coach.
- **Betrieb und Dokumentation:** PWA-Installation, Deployment über GitHub und Vercel, README und Deployment-Anleitung für Nicht-Entwickler.

3 Funktionsumfang

Fünf Bereiche, erreichbar über die untere Navigation.

Heute: das Tagesbudget

- Ring mit den gegessenen Kalorien gegen individuell konfigurierte Zielwerte, darunter Balken für Protein, Fett und Salz.
- Mahlzeiten gruppiert nach Frühstück, Mittag, Abend und Snacks, je mit Zwischensumme.
- Jeder Eintrag zeigt Menge in Gramm und Kalorien und lässt sich mit einem Tipp entfernen.
- Überschreitungen werden farblich markiert, der Ring wechselt die Farbe.
- Alle Zahlen in Tabellenziffern und im Schweizer Zahlenformat.

Scannen: Lebensmittel erfassen

- **Kamera-Scan** liest EAN-13, EAN-8, UPC-A und UPC-E. QR-Codes werden mitgelesen: steckt ein GS1 Digital Link oder eine Produkt-URL dahinter, wird die Artikelnummer daraus gezogen.
- **Open Food Facts** liefert Name, bevorzugt deutsch, Marke, Nährwerte pro 100 g und die Portionsgrösse. Die Abfrage bricht nach fünf Sekunden ab, dann greift eine kleine Demo-Datenbank mit Schweizer Produkten.
- **Frischprodukte:** zwölf Grundnahrungsmittel ohne Barcode, von Banane bis Pouletbrust, mit Suchfeld.
- **Eigenes Produkt:** Werte pro 100 g für Restaurantessen oder Selbstgekochtes.
- Danach Menge per Schieberegler von 10 bis 500 g, Mahlzeit wählen, hinzufügen. Die Herkunft der Werte wird als Etikett angezeigt.

Training: Plan, Protokoll und Wochenauswertung

- Vier Trainingstage, Legs, Pull, Push und Abs, mit 25 Übungen; dahinter liegen 26 Animationen, weil sich zwei Übungen eine teilen und zwei Bewegungen für Übungen ausserhalb des Plans bereitstehen. Sieben Muskeln sind als FOKUS markiert und bekommen in der Auswertung Vorrang.
- **Pro Übung:** Zielmuskel als Körpergrafik, Sätze zum Abhaken, Wiederholungsbereich, Gewicht pro Satz, Maschinen-Einstellung, animierte Ausführung mit Erklärtext.
- **Gewichte wandern mit:** beim ersten Abhaken übernimmt die App den Stand vom letzten Training als Vorschlag, mit Datumshinweis. Ein Knopf überträgt ein Gewicht auf alle Sätze.
- **Eigene Übungen** lassen sich pro Tag anlegen und wieder entfernen, erfasste Sätze bleiben im Protokoll erhalten.

- Ein Suchfeld filtert die Übungen des Tages.
- **Wochenauswertung:** erledigte Sätze von Montag bis Sonntag gegen den Plan, pro Muskel als Balken und als Heatmap vorne und hinten. FOKUS-Muskeln unter Plan werden hervorgehoben.

Coach: Vorschläge aus dem Restbudget

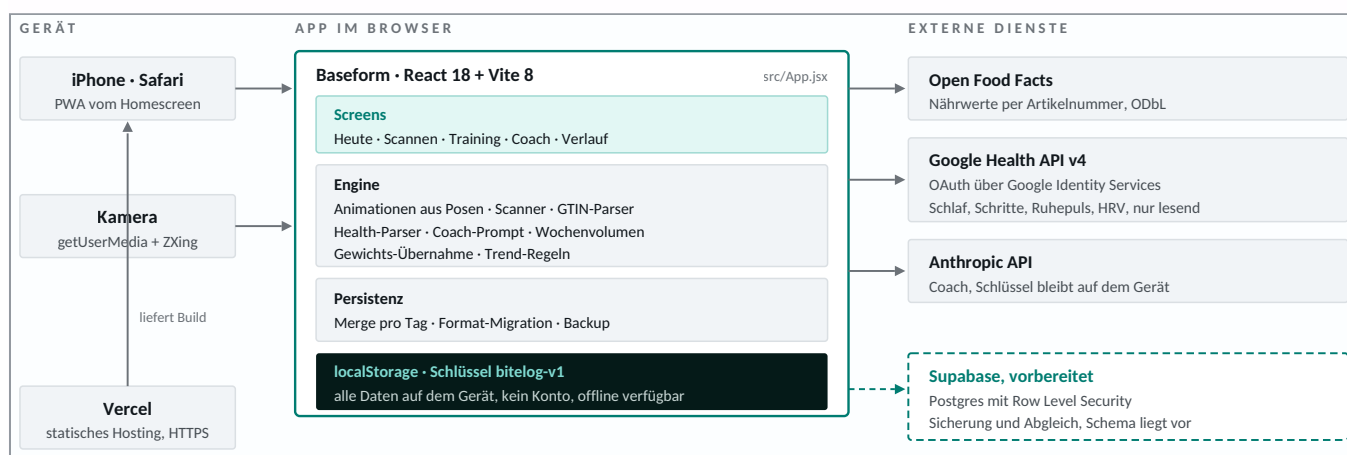
- Drei Modi: Mahlzeiten nur aus den Zutaten, die zuhause sind; Einkaufsvorschläge mit Liste und grober Kostenschätzung in CHF; oder eine Analyse des ganzen Tages.
- Der Prompt enthält Profil und Ziele, jede gegessene Position mit Nährwerten, das Restbudget, die heute erledigten Übungen samt Gewicht pro Satz und Vorwert vom letzten Mal, dazu Schlaf, HRV und Ruhepuls.
- Der API-Schlüssel wird nur lokal gespeichert und direkt vom Gerät an die Schnittstelle gesendet. Ohne Schlüssel erklärt die App, wie man einen bekommt.

Verlauf: Gewicht, Erholung, Kalender, Backup

- **Morgengewicht** mit Plausibilitätsprüfung, Kurve der letzten 30 Messungen, 7-Tage-Schnitt und Vergleich zur Vorwoche. Ein Regelwerk übersetzt die Differenz in eine Empfehlung: fällt der Schnitt schneller als 0.4 kg pro Woche, mehr essen; steigt er über 0.25 kg, weniger; dazwischen nichts ändern.
- **Fitbit über Google Health:** Schlafdauer mit Phasen, Schritte, Verbrauch, Ruhepuls und HRV. Die Werte bleiben nach dem Abruf offline sichtbar.
- **Kalender:** Monatsansicht mit drei Punkten pro Tag für Mahlzeiten, Training und Schlaf. Ein Tag zeigt alle Einträge, Sätze mit Gewicht und die Schlafphasen.
- **Backup:** Export als JSON-Datei, Import auf einem neuen Gerät.

4 Architektur

Eine Progressive Web App ohne eigenen Server. Der Browser ist die Laufzeit, das Gerät der Speicher. Vier externe Dienste kommen dazu, jeder mit definiertem Fehlerfall und einer Rückfallebene.



Alles läuft im Browser, die Daten bleiben auf dem Gerät. Externe Dienste werden nur angesprochen, wenn der Nutzer es auslöst; der gestrichelte Teil ist vorbereitet, aber noch nicht angebunden.

Technologie-Stack

Bereich	Technologie	Warum
UI	React 18.3, Hooks, JSX	Komponentenmodell mit klarer Zustandsführung; useMemo für Wochen- und Kalenderberechnungen
Build	Vite 8.1, @vitejs/plugin-react	Schneller Dev-Server, dynamischer Import für die Scanner-Bibliothek, kleiner Produktions-Build
Scan	@zxing/browser 0.2, @zxing/library 0.23	Bewährter Decoder für EAN, UPC und QR direkt im Browser, ohne Server-Roundtrip
Styling	Inline-Styles, zentrales Farbsystem, SVG	Keine CSS-Framework-Abhängigkeit; Ring, Kurven, Körpergrafiken und Animationen sind handgezeichnetes SVG
PWA	Web-App-Manifest, Apple-Meta-Tags, Icons	Installation auf dem Homescreen, Vollbild, eigene Statusleiste, Speicher vor Safaris Sieben-Tage-Löschung geschützt
Daten	localStorage, JSON-Backup	Sofort verfügbar, offline-tauglich, keine Konten; vorbereitet ist Supabase Postgres mit Row Level Security
KI	Anthropic Messages API, Claude Sonnet 4.6	Strukturierter Prompt mit Tageskontext, Antwortlänge und Format vorgegeben
Auth	Google Identity Services, OAuth 2.0 Token Client	Zugriff auf Google Health mit drei Nur-Lese-Scopes; Token nur im Arbeitsspeicher
Hosting	Vercel, GitHub	Automatisches Deployment bei jedem Push; HTTPS ist Voraussetzung für den Kamerazugriff
Laufzeit	Node.js 22.12 oder neuer, nur Entwicklung	Von Vite 8 vorausgesetzt; zur Laufzeit braucht die App keinen Node-Prozess

Aufbau des Codes

Der Client umfasst 2'275 Zeilen React und JSX, davon 2'266 in `src/App.jsx`. Diese eine Datei ist in klar abgegrenzte Blöcke gegliedert. Die Struktur hielt die Iterationen schnell; die Aufteilung in Module ist als Teil der Supabase-Anbindung eingeplant.

```

bitelog/
├─ index.html           PWA-Meta, Viewport, Theme
├─ src/
│  ├─ main.jsx         React-Root, StrictMode
│  └─ App.jsx          gesamte App
├─ public/
│  ├─ manifest.webmanifest  Name, Icons, standalone
│  └─ icon-{180,192,512}.png
├─ supabase/
│  └─ schema.sql       8 Tabellen, RLS-Policies
├─ vite.config.js
├─ package.json        4 Abhängigkeiten zur Laufzeit
├─ README.md           Setup, Funktionen, Daten, Lizenz
└─ DEPLOY.md           GitHub, Vercel, iPhone

```

1. **Daten:** Ziele, Farbsystem, Demo-Datenbank, Frischprodukte, Trainingsplan, Muskelregionen, 26 Animationsdefinitionen.

2. **Engine:** Animations-Hook, Kamera-Scanner, GTIN-Parser, Wochen- und Gewichtslogik, Google-Health-Client mit Parser, Coach-Prompts.
3. **UI-Bausteine:** Card, Bar, Ring, PillTabs, Icons, MuscleBadge, BodyHeatmap, ExerciseAnim.
4. **Screens:** Heute, Scannen, Training, Coach und Verlauf mit ihren Formularen.
5. **App-Shell:** Zustand, Persistenz mit Merge, Navigation, Toast.

Externe Schnittstellen

Dienst	Was geholt wird	Technik	Wenn es schiefgeht
Open Food Facts	Produktname, deutsch bevorzugt, Marke, Nährwerte pro 100 g, Portions- und Packungsgrösse	REST v2, Feldauswahl per Query, AbortController mit Timeout nach fünf Sekunden	Demo-Datenbank mit Schweizer Produkten, sonst Hinweis auf die Eingabe eigener Werte
Kamera und ZXing	EAN-13, EAN-8, UPC-A, UPC-E, QR	getUserMedia mit Rückkamera in 1280 mal 720, BrowserMultiFormatReader, TRY_HARDER	Der Fehlername wird auf eine handlungsfähige Meldung abgebildet, die Tippeingabe bleibt
Google Health	Schlaf mit Phasen, Schritte, Gesamtverbrauch, Ruhepuls, HRV	Google Identity Services Token Client, Bearer-Token, fünf Endpunkte der API v4	Fehlende Metriken werden benannt, der Rest bleibt nutzbar, das Token wird verworfen
Anthropic	Coach-Antwort: Analyse, Vorrat, Einkauf	Messages API, Modell Claude Sonnet 4.6, höchstens 1'200 Tokens, direkter Browserzugriff	Klare Meldung mit Prüfhinweis zu Schlüssel und Verbindung, kein Absturz

Bewusst schlank gehalten

Keine Server, keine Konten, keine Cookies und vier Abhängigkeiten zur Laufzeit. Die App sendet Daten nur an jene Dienste, die der Nutzer aktiv auslöst: einen Barcode an Open Food Facts, den Tageskontext an Anthropic, eine Anmeldung an Google. Die Scanner-Bibliothek wird erst beim Öffnen des Scanners nachgeladen, damit der Start leicht bleibt.

Vier Umsetzungsdetails

Kamera-Scanner. Die Kamera wird vor dem Laden der Decoder-Bibliothek angefragt, damit die Erlaubnisabfrage direkt nach dem Tippen kommt. Reacts StrictMode startet Effekte doppelt, deshalb räumt der Scanner nur den eigenen Stream ab; sonst würde der erste Durchlauf die Kamera des zweiten beenden. Fehlernamen wie `NotAllowedError` werden auf konkrete Anleitungen abgebildet, inklusive Pfad in den Safari-Einstellungen. Ein Treffer löst eine kurze Vibration aus.

Feinschliff fürs iPhone. Safe-Area-Insets für Notch und Home-Indikator, `100dvh` statt `100vh`, Eingabefelder mit 16 px gegen den automatischen Zoom, `touch-action: manipulation` gegen Doppeltipp-Zoom, kein Gummiband-Scrollen, Navigation mit Backdrop-Blur. Safari löscht den Speicher einer Website nach sieben Tagen ohne Besuch; installierte Web-Apps sind davon ausgenommen, weshalb die Dokumentation erklärt, warum der Start über das Homescreen-Icon Pflicht ist.

Zugänglichkeit. Jeder Icon-Button trägt ein `aria-label`, der Scanner ist ein `role="dialog"` mit `aria-modal`, Übungszeilen sind per Tastatur auf- und zuklappbar, der Fokusring ist sichtbar, Zahlen stehen in Tabellenziffern. Fortschritt wird zusätzlich zur Farbe als Text angezeigt, etwa 9 von 13 Sätzen.

Übungsanimationen. Jede Übung ist als zwei Posen aus Punktkoordinaten beschrieben, dazu Segmente, Geräte-Linien und optionale Kabelzüge. Eine Engine interpoliert sinusförmig per `requestAnimationFrame` und hebt das arbeitende Segment

farbig hervor. So sind 26 Übungen je sechs Zeilen Daten statt eigener Komponenten. Die Systemeinstellung «Bewegung reduzieren» wird respektiert, und die 15 Muskelregionen dienen sowohl dem Badge pro Übung als auch der Heatmap.

Dokumentation und Auslieferung

- **README** mit Setup, Funktionsübersicht, Aktivierung von Coach und Google Health, iPhone-Installation, Datenhaltung und Lizenzhinweis.
- **DEPLOY** beschreibt zwei Wege bis auf den Homescreen: rein im Browser über GitHub und Vercel, oder lokal mit Node und Vercel CLI.
- **Kommentare erklären das Warum**, nicht das Was: etwa weshalb der Speicherschlüssel nach der Umbenennung bleibt, warum die Kamera vor der Bibliothek angefragt wird oder wieso der Fitbit-Verbrauch nicht aufs Budget kommt.
- **Fehlermeldungen** sagen dem Nutzer, was er tun kann, statt nur, dass etwas nicht ging.
- **Git** mit sprechenden Commits, die je ein abgeschlossenes Feature-Bündel beschreiben. Ein Push auf GitHub löst das Vercel-Deployment aus; HTTPS ab Werk ist Voraussetzung für den Kamerazugriff.

5 Technische Kernprobleme

Sieben Stellen, an denen das Projekt über «Formular speichert in Browser» hinausging. Jede Karte nennt Problem, Lösung und das Prinzip dahinter.

DATENINTEGRITÄT

Speichern ohne Datenverlust

Problem

Auf dem iPhone waren die App vom Homescreen und ein Safari-Tab gleichzeitig offen. Das Fenster mit dem älteren Stand überschrieb beim Speichern den ganzen Datenblock, inklusive Tagen, die es nie gesehen hatte.

Lösung

Beim Speichern wird pro Schlüssel zusammengeführt: der Arbeitsspeicher gewinnt für die Tage, die er kennt, alle anderen Tage bleiben von der Platte erhalten. Gewichte werden nach Datum gemischt.

Warum es zählt

Der Fehler hatte reale Trainingsdaten gekostet. Die Lösung floss auch ins Supabase-Schema: eine Zeile pro Tag und Übung, damit die Kollision strukturell unmöglich ist.

ZEIT

Der richtige Tag nach Mitternacht

Problem

Der Tagesschlüssel kam aus `toISOString()`, also aus UTC. In der Schweiz landete ein Eintrag zwischen 00:00 und 02:00 auf dem Vortag, während die Kopfzeile schon den neuen Tag zeigte.

Lösung

Ein eigener Tagesschlüssel aus der lokalen Zeit, konsequent auch in der Wochenberechnung von Montag bis Sonntag.

Warum es zählt

Zeitzonenfehler sind leise und verfälschen Auswertungen, ohne je eine Fehlermeldung zu zeigen.

DATENMODELL**Stabile Schlüssel für eigene Übungen****Problem**

Eine Laufnummer als Schlüssel hätte beim Löschen einer Übung alle folgenden Schlüssel verschoben und erfasste Sätze der falschen Übung zugeordnet.

Lösung

Eigene Übungen bekommen `<tag>-c<id>` mit der ID aus der App. Beim Entfernen wird nur die Definition gelöscht, das Protokoll bleibt.

Warum es zählt

Identität ist eine Datenmodell-Entscheidung, keine Frage der Oberfläche; sie wurde vor dem Feature getroffen und ins Supabase-Schema übernommen.

MIGRATION**Alte Sicherungen bleiben lesbar****Problem**

Frühere Versionen kannten ein Gewicht pro Übung; neu gibt es Gewichte pro Satz und Trainingstag. Alte Backups hätten Werte verloren.

Lösung

Eine Normalisierung erkennt beide Altformate, hebt sie auf das neue Schema und trägt den alten Stand als heutigen Startwert ein.

Warum es zählt

Wer Daten lokal hält, trägt die Verantwortung für Formatwechsel selbst. Der Speicherschlüssel blieb bei der Umbenennung deshalb bewusst unverändert.

PARSING**QR-Codes und GS1 Digital Link****Problem**

Auf Verpackungen liegen zunehmend QR-Codes statt Strichcodes. Sie enthalten die Artikelnummer als GS1 Digital Link, also in der Form `.../01/<GTIN>/...`, oder eingebettet in eine Produkt-URL.

Lösung

Ein Parser zieht die acht- bis vierzehnstellige Nummer aus dem Text und entfernt führende Nullen, weil GS1 vierzehnstellig schreibt, Open Food Facts das Produkt aber unter der dreizehnstelligen EAN führt. QR-Codes ohne Produktnummer werden einmal gemeldet, danach läuft der Scan weiter.

Warum es zählt

Wer fremde Codes einliest, bekommt selten genau die Form, mit der das eigene System arbeitet. Die Vereinheitlichung gehört deshalb an die Systemgrenze, in eine Funktion, die man für sich prüfen kann; dahinter kennt der Rest der Anwendung nur noch eine einzige saubere Form. Die Alternative verteilt dieselbe Fallunterscheidung über den ganzen Code und lässt sie an genau einer Stelle veralten.

OAuth**Google Health im Browser****Problem**

Der Token-Client von Google Identity Services muss synchron aus dem Klick heraus starten, sonst blockiert Safari das Anmeldefenster als Popup. Ausser Schlaf und Schritten sind die Feldnamen der Antworten kaum dokumentiert.

Lösung

Das Token lebt nur im Arbeitsspeicher, rund eine Stunde, mit drei Nur-Lese-Scopes. Der Parser sucht erst nach bekannten Feldnamen, dann nach der ersten Zahl, bis zwei Ebenen tief. Die Nacht wird dem Aufwachttag zugeordnet, Schritte als Tagesmaximum, Puls und HRV als Tageswert aggregiert.

Warum es zählt

Die Regeln der Plattform sind Randbedingungen des Entwurfs, nicht Kleinkram für den Schluss: Wer erst nach dem Bauen nachliest, unter welchen Bedingungen ein Ablauf überhaupt starten darf, baut ihn zweimal. Dasselbe gilt für die Gegenseite. Eine Schnittstelle, deren Antwortform man nicht kennt, muss man tolerant lesen, sonst verwandelt eine Umbenennung auf fremder Seite die eigene Anzeige stillschweigend in eine leere Fläche.

KI**Ein Coach mit vollständigem Kontext****Problem**

Ein Vorschlag ist nur brauchbar, wenn er den ganzen Tag kennt. Zugleich enthält der Fitbit-Wert für den Gesamtverbrauch bereits jene Energie, die im Tagesziel schon eingerechnet ist.

Lösung

Der Prompt enthält Profil und Ziele, jede gegessene Position mit Nährwerten, das Restbudget, die heutigen Übungen mit Gewicht pro Satz und Vorwert, die Maschinen-Einstellungen sowie Schlaf, HRV und Ruhepuls. Drei Aufgabentypen mit vorgegebenem Antwortformat und Längenlimit. Die App rechnet den Verbrauch nicht dazu und sagt dem Modell ausdrücklich, dass es das ebenfalls nicht tun darf.

Warum es zählt

Ein Sprachmodell prüft die Zahlen nicht, die es bekommt, sondern setzt sie in Beziehung. Damit verschiebt sich die Verantwortung nach vorne: Wer den Kontext zusammenstellt, entscheidet über die Qualität der Antwort, und jede Kennzahl, deren Bedeutung mehrdeutig bleibt, wird irgendwann falsch verwendet. Solche Mehrdeutigkeiten gehören ausgeräumt, bevor sie im Prompt landen, denn eine flüssig formulierte falsche Empfehlung fällt weit weniger auf als ein Programmfehler.

6 Datenmodell

Alle Daten liegen unter einem Schlüssel im localStorage. Trainingsdaten sind nach Tag und Übung adressiert, damit zwei Fenster nie denselben Eintrag anfassen müssen. Übungen bekommen einen stabilen Schlüssel: `d1-0` für die erste Planübung an Tag 1, `d1-c17565000000000` für eine eigene Übung mit ihrer ID.

```
{
  entries:      [{ id, date, mealId, name, grams, kcal, p, c, f, salt }], // Mahlzeiten
  setsLog:      { "2026-09-05": { "d1-0": 4 } }, // Sätze je Tag und Übung
  exWeights:    { "2026-09-05": { "d1-0": ["85", "85"] } }, // Gewicht je Satz
  exSettings:   { "d1-0": "Plattform tief, Sitz 3" }, // Einstellung je Übung
  customExercises: { d1: [{ id, name, sets, reps, muscle, desc } ] }, // eigene Übungen
  weights:      [{ date, kg }], // Morgengewicht
  healthData:   { "2026-09-05": { asleep, stages, steps, rhr, hrv } },
  pantry:       ["Hüttenkäse", "Eier", "Reis"], // Vorrat für den Coach
  healthSyncedAt: "2026-09-05T07:12:00.000Z"
}
```

Was heute auf dem Gerät bleibt

- **Schlüssel bleiben lokal.** Der Anthropic-Schlüssel liegt im localStorage und geht direkt vom Gerät an die Schnittstelle. Das Google-Token lebt nur im Arbeitsspeicher und verfällt nach rund einer Stunde.
- **Nur-Lese-Zugriff.** Die drei Google-Health-Scopes erlauben ausschliesslich das Lesen von Schlaf, Aktivität und Messwerten.
- **Backup in Nutzerhand.** Export als JSON, etwa nach iCloud Drive; Import mit Plausibilitätsprüfung und Neustart.
- **Lizenz beachtet.** Nährwerte von Open Food Facts stehen unter der Open Database License; die Quelle wird in der App als Etikett und im README genannt.

Vorbereitet: Supabase-Schema mit Row Level Security

Das Schema in `supabase/schema.sql` überträgt das lokale Datenmodell nach Postgres und behebt die Ursache des Datenverlusts strukturell: eine Zeile pro Tag und Übung statt eines JSON-Blocks. Jede Tabelle hängt an `auth.uid()`, Row Level Security ist überall aktiv.

Tabelle	Primärschlüssel	Zweck
<code>meals</code>	id als Identity, Index auf (user_id, day)	Eine Zeile pro Mahlzeit mit Nährwerten.
<code>training_sets</code>	(user_id, day, ex_key)	Erledigte Sätze und Gewichte als JSON-Array je Übung und Tag; sie ändern sich gemeinsam.
<code>exercise_settings</code>	(user_id, ex_key)	Maschinen-Einstellung, gilt pro Übung, nicht pro Tag.
<code>custom_exercises</code>	(user_id, id), ID aus der App	Eigene Übungen; der Schlüssel bleibt auf beiden Seiten derselbe.
<code>body_weights</code>	(user_id, day)	Morgengewicht.
<code>pantry</code>	(user_id, item)	Vorrat für den Coach.
<code>health_days</code>	(user_id, day)	Fitbit-Werte pro Tag, Ziel des stündlichen Sync-Jobs.
<code>google_tokens</code>	user_id	Refresh-Token für den Job. Bewusst ohne Policy: RLS ist aktiv, weder anon noch authenticated kommen heran, nur die Edge Function mit service_role.

Sicherheitsprinzipien im Schema

Der Standardwert `user_id default auth.uid()` sorgt dafür, dass der Client die Spalte nie selbst setzen muss. Eine Policy «eigene Zeilen» mit `using` und `with check` gilt für sieben Tabellen und wird per Schleife erzeugt. Beim Entfernen des Kontos wird kaskadierend gelöscht.

Geplantes Zusammenspiel

localStorage bleibt die sofortige Ebene, damit das Erfassen im Gym ohne Netz funktioniert. Supabase wird Sicherung und Abgleich zwischen Geräten. Der Fitbit-Abgleich wandert vom manuellen Tipp in eine stündliche Edge Function mit Refresh-Token.

7 Qualität und offene Punkte

Das Projekt hat noch keine automatisierte Testsuite. Verifiziert wurde bisher im laufenden Betrieb auf dem Zielgerät; die Korrekturen im September gehen auf Beobachtungen aus dieser Nutzung zurück.

Projektverlauf

Datum	Schritt
25. Juli 2026	Grundgerüst unter dem Namen BiteLog. Tagesring, Barcode-Eingabe mit Open Food Facts, Trainingsplan mit Animationen und Heatmap, Coach, Verlauf mit Gewichtstrend und Kalender, PWA-Installation.
25. Juli 2026	Kamera-Scan, QR-Codes, iPhone-Feinschliff. ZXing-Scanner mit Fehlerbehandlung, GS1-Parser, Gewicht pro Satz und Trainingstag mit Übernahme vom letzten Mal.
30. Juli 2026	Push-Tag, Maschinen-Einstellungen, eigene Übungen. Vollständiger Push-Plan, Einstellung pro Übung, eigene Übungen mit stabilen Schlüsseln, Suche.
31. Juli 2026	Google Health. OAuth im Browser, Schlaf mit Phasen, Schritte, Verbrauch, Ruhepuls und HRV, eingebunden in Kalender und Coach.
September 2026	Baseform. Umbenennung, Merge-Speicherung gegen Datenverlust, Zeitzone-Korrektur, Supabase-Schema mit Row Level Security, überarbeitete Dokumentation.

Zwischen Ende Juli und September lag der Schwerpunkt nicht auf neuen Funktionen, sondern auf der täglichen Nutzung, aus der die Korrekturen im September hervorgingen. Vom 30. Juli bis zum 5. September 2026 kamen so 16 Trainingstage zusammen, also rund drei pro Woche, mit 259 abgehakten Sätzen und 267 einzeln erfassten Satzgewichten auf 23 der 25 Planübungen.

Bewusste Grenzen des aktuellen Stands

- **Einzelnutzer-App.** Ziele, Profil und Google-Client-ID stehen im Code. Wer die App selbst betreibt, legt ein eigenes Google-Cloud-Projekt an; Gesundheits-Scopes gelten als sensibel und brauchen für fremde Nutzer eine Überprüfung durch Google.
- **Noch keine automatisierten Tests.** Die Kernlogik, also Merge, Tagesschlüssel, GTIN-Parser und Health-Parser, ist als reine Funktionen geschrieben und damit testbar; die Tests stehen unten als offener Punkt.
- **Eine Datei.** Die Modulaufteilung ist mit der Supabase-Anbindung eingeplant.

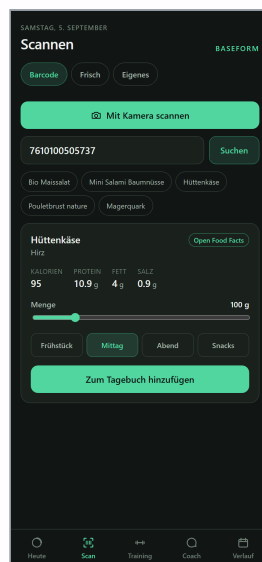
Offen

- **Supabase-Synchronisation.** Anmeldung und Sync-Schicht über dem lokalen Speicher, Abgleich zwischen Geräten. localStorage bleibt die sofortige Ebene. Das Schema liegt vor.
- **Stündlicher Fitbit-Abgleich.** Eine Edge Function liest das Refresh-Token aus `google_tokens` und schreibt nach `health_days`; kein manuelles Aktualisieren mehr. Die Tabellen stehen.
- **Modulaufteilung und Tests.** App.jsx in Daten, Engine, Bausteine und Screens trennen; Unit-Tests für Merge, Tagesschlüssel sowie GTIN- und Health-Parser.
- **Ziele und Profil in der App.** Kalorien- und Makroziele, Körperdaten und Trainingsfokus als Einstellungen statt als Konstanten im Code.
- **Google-Verifizierung.** Den Zustimmungsbildschirm aus dem Testmodus holen, damit Refresh-Tokens nicht nach sieben Tagen verfallen und Dritte die App nutzen können. Setzt den stündlichen Abgleich voraus.

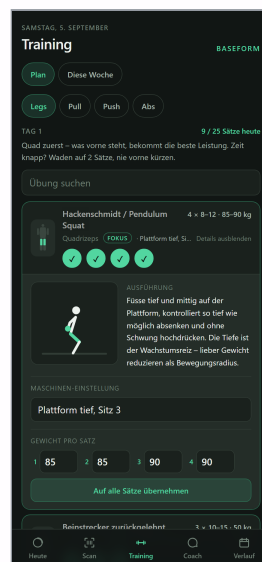
8 Einblicke



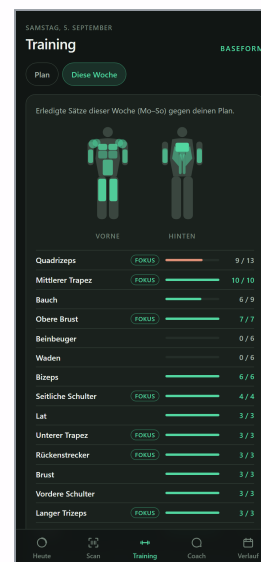
Heute. Tagesring, Balken für Protein, Fett und Salz, darunter die Mahlzeiten.



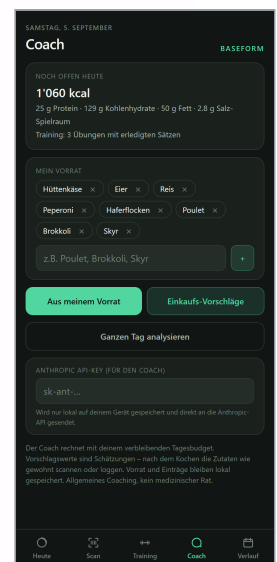
Scannen. Treffer live von Open Food Facts, Menge und Mahlzeit wählen.



Training. Animation, Ausführungshinweis, Maschinen-Einstellung, Gewicht pro Satz.



Wochenauswertung. Erledigte Sätze gegen den Plan, als Balken und Heatmap.



Coach. Restbudget, Vorrat als Chips, drei Modi. Der Schlüssel bleibt auf dem Gerät.

9 Was das Projekt zeigt

Fähigkeit	Beleg in Baseform
React und Zustandsführung	Fünf Screens mit gemeinsamem Zustand in der App-Shell, funktionale Updates, useMemo für Kalender-, Wochen- und Trendberechnungen, Refs für Kamera-Stream und Token, kontrollierte Formulare mit Validierung
Browser-APIs	getUserMedia, dynamischer Import, localStorage, FileReader und Blob für das Backup, requestAnimationFrame, matchMedia für «Bewegung reduzieren», Vibration API, AbortController für Timeouts
API-Integration und OAuth	Google Identity Services Token Client mit Safari-Popup-Regel, Bearer-Authentifizierung, tolerantes Parsen schwach dokumentierter Antworten, REST mit Feldauswahl bei Open Food Facts
Arbeiten mit LLMs	Prompt-Aufbau mit strukturiertem Tageskontext, drei Aufgabentypen, Format- und Längenvorgaben, fachliche Leitplanken gegen Doppelzählung, sauberer Umgang mit Schlüssel und Fehlern
Datenmodellierung und SQL	Postgres-Schema mit zusammengesetzten Primärschlüsseln, Identity-Spalten, JSONB, Index, Row Level Security mit per Schleife erzeugten Policies, bewusst policy-lose Tabelle für service_role
Datenintegrität	Merge-Speicherung pro Tag, zeitzonenkorrekte Tagesschlüssel, stabile Identitäten, Migration alter Formate, Speicherschlüssel bei der Umbenennung beibehalten
PWA und Mobile UX	Manifest, Apple-Meta-Tags, Safe-Area, dvh, Zoom- und Overscroll-Kontrolle, Homescreen-Installation und Safaris Sieben-Tage-Regel verstanden und dokumentiert
SVG und Animation	Kalorienring mit Dash-Offset, Gewichtskurve, Körpersilhouette mit 15 Muskelregionen, Heatmap mit Intensität, datengetriebene Posen-Interpolation für 26 Übungen
Fehlerbehandlung	Jede Schnittstelle mit Timeout oder Rückfallebene; Fehler werden in Anweisungen übersetzt; Teilausfälle einzelner Metriken werden benannt, ohne den Rest zu blockieren
Zugänglichkeit	aria-labels, Dialog-Rollen, Tastaturbedienung, sichtbarer Fokus, Text neben Farbe, Reduced-Motion
Dokumentation und Kommunikation	README und DEPLOY für Nicht-Entwickler geschrieben, Kommentare mit Begründung, Fehlermeldungen in der Sprache des Nutzers, sprechende Commits
Produktdenken	Aus dem eigenen Bedarf heraus priorisiert: Gewichte, die mitwandern; Regeln, die den Gewichtstrend in eine Empfehlung übersetzen; FOKUS-Muskeln, die in der Auswertung Vorrang bekommen

Stand	September 2026 · in aktiver Weiterentwicklung
Kontakt	Gabriel Suter · kontakt@gabrielsuter.dev · gabrielsuter.dev
Quellcode	Privates Repository, Einsicht und Demo auf Anfrage
Betrieb	Progressive Web App, Hosting auf Vercel, Installation über Safari auf dem Homescreen