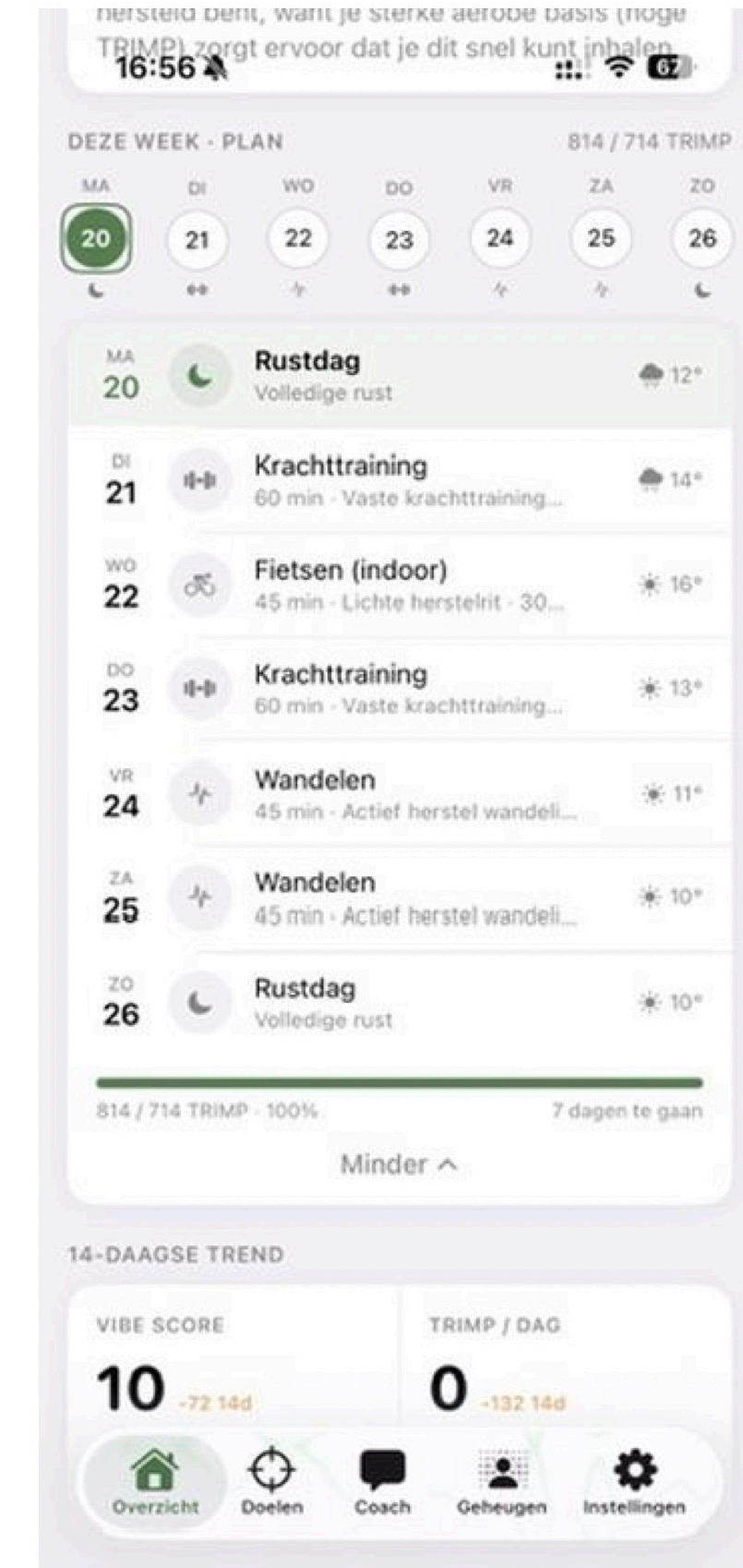
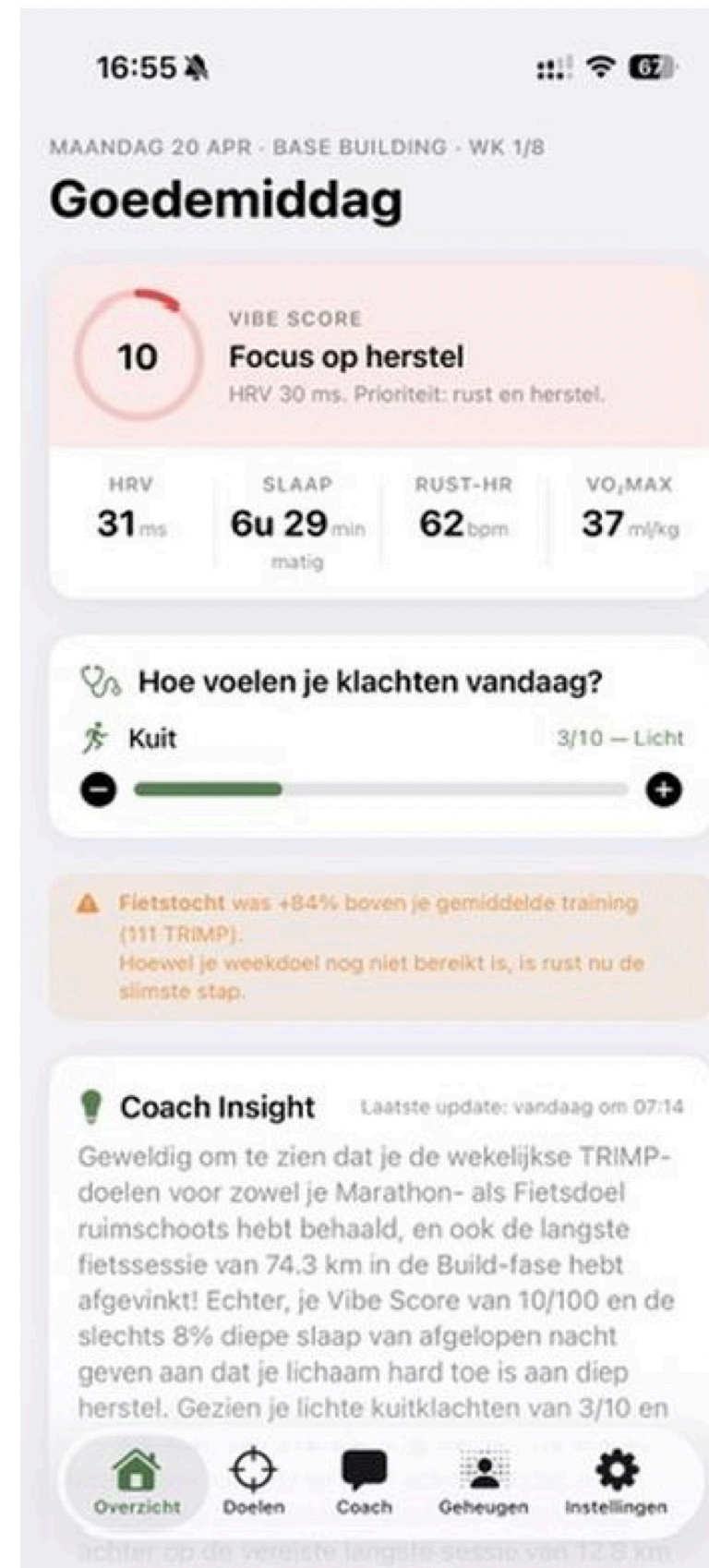


Vibe Coding – begin gewoon

Software ontwikkelen met
AI, zonder dat je kan
programmeren



Wie ben ik?

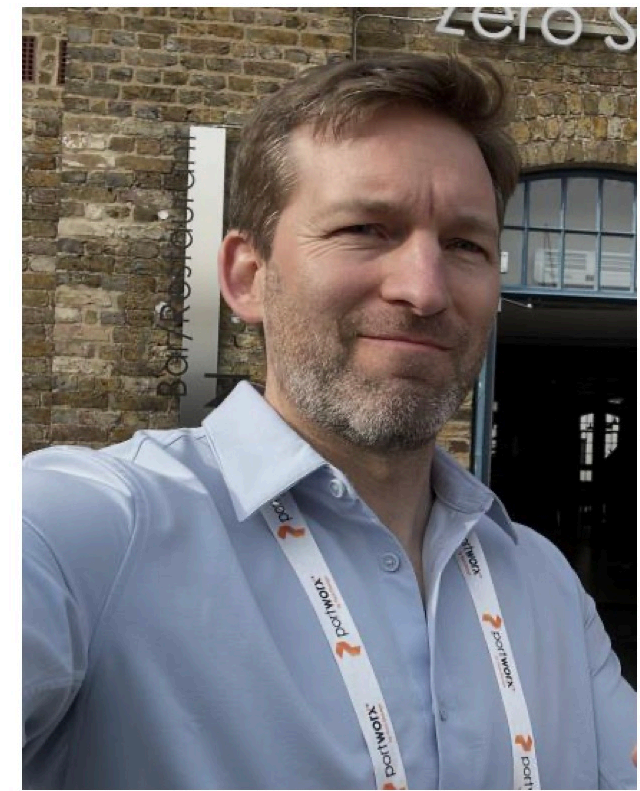
<https://www.linkedin.com/in/mark-clausing/>

Opleidingen:

- 25+ jaar geleden:
HBO Informatica (Java, C++, Novell netwerk (!), SQL)
- 20+ jaar geleden:
Universiteit Informatiekunde (C, Java, Matlab, bedrijfskunde)

Werkervaring:

- 2009 Product Owner website Vodafone
- 2010 Product & Services Manager (prince2)
(MyVodafone app android/ios)
- 2013 Scrum Master / Repository
Maintainer Vodafone websites (git, jenkins, jira, twitter bootstrap, angular, java)
- 2017 Project manager Vodafone+Ziggo
- 2019 Agile Coach Vodafone
- 2021 RTE/STE Alliander



Ik heb algemene IT kennis, maar de laatste keer dat ik zelf iets geprogrammeerd heb is 20 jaar geleden...

github.com/markclausing/vibecoach

Waarom ben ik begonnen met een vibe coding projectje?



Een Gemini Pro abonnementje voor plaatjes voor de boekpresentatie van mijn dochter...

Kosteloos	Google AI Plus ¹	Google AI Pro ²	Google AI Studio ³
Krijg hulp bij dagelijkse taken van de AI van Google, op je werk, op school of thuis.	Krijg uitgebreidere toegang tot nieuwe en krachtige functies waarmee je je productiviteit en creativiteit een boost geeft.	Krijg uitgebreidere toegang tot nieuwe en krachtige functies waarmee je je productiviteit en creativiteit een boost geeft.	Krijg het h... het allerbe... exclusieve...
€ 0 EUR/maand met een Google-account	€ 7,99 EUR/maand € 3,99 EUR per maand voor 2 maanden	€ 21,99 EUR /maand	€ 274,99- € 139,9
Aan de slag	Aan de slag	Aan de slag	Aan de slag
	Alles van het Gratis abonnement, plus:	Alles van het Gratis abonnement, plus:	Alles
◆ Gemini-app - Je persoo... en krach... ✓ Toegar... ✓ Versc... 3.1 Pro ✓ Beeldg... ✓ Deep R... ✓ Gemini... ✓ Canvas... ✓ Gemi... ○ 50 dagelij... - Credits voor het maken v... video's in Flow en Whisk ■ Flow5 - Toegang tot onze AI-filmtool om filmische scènes en verhalen te maken, inclusief beperkte toegang tot Veo 3.16 🔊 NotebookLM - Onderzoeks- en schrijffassistent met onder andere meer audio-overzichten en notebooks	◆ Gemini-app - Krijg uitgebreidere toegang tot ons intelligentste model 3.1 Pro, Deep Research en beeldgeneratie met Nano Banana Pro, plus videogeneratie met Veo 3.1 Fast 6 ○ 1000 AI-credits per maand⁴ - Credits voor het maken van video's in Flow en Whisk ■ Flow5 - Uitgebreidere toegang tot onze AI-filmtool om filmische scènes en verhalen te maken, inclusief beperkte toegang tot Veo 3.16 🔊 Gemini Code Assist en Gemini CLI - Hogere dagelijkse limieten voor verzoeken in de Gemini CLI en IDE-extensies van Gemini Code Assist	◆ Gem... - De h... mode... videc... plus 1... en G... de V... ○ 25.00... - Cred... videc... ■ Flow... - Mees... tot o... filmis... te ma... toeg... 🔊 Gem... - De h... limiet... Gem...	

Maar wacht, je kan best veel met zo'n account..

Changelog

Gemini 3.1 Pro is now available in Jules

March 9, 2026

On this page

- Overview
- Gemini 3.1 Pro is now available in Jules
Mar 09, 2026
- Auto-Fixing CI Failures and configure Jules to commit as you
Feb 19, 2026

github.com/markclausing/vibecoach

De prompt waarmee ik begon

ik heb een gemini pro account. ik wil apps kunnen maken, maar ik kan zelf niet programmeren, dus ik wil AI gebruiken: Jules? ik wil zoveel mogelijk lokaal doen op mijn Mac en het liefst in een docker container zodat ik meerdere projecten kan gebruiken. Ik wil GitHub...



Denkproces tonen ▾

Hoi! Wat een gaaf plan. Omdat je zelf niet programmeert, is het inzetten van een autonome AI-agent de slimste route om toch volwaardige apps te kunnen bouwen. Jules is hier inderdaad de perfecte kandidaat voor.

Laten we kijken hoe jouw wensen (lokaal, Docker, GitHub en test-pipelines) samen kunnen komen in één strakke workflow.

1. Wat is Jules precies?

Jules is Google's autonome programmeer-agent. In plaats van dat je stukjes code kopieert en plakt uit een chatvenster, geef je Jules een omschrijving van wat je wilt hebben (bijvoorbeeld: "Bouw een inlogscherf"). Jules leest je volledige project, plant de stappen, schrijft de code, en levert de afgeronde bestanden af.

Brainstormen over wat 'we' gaan maken..

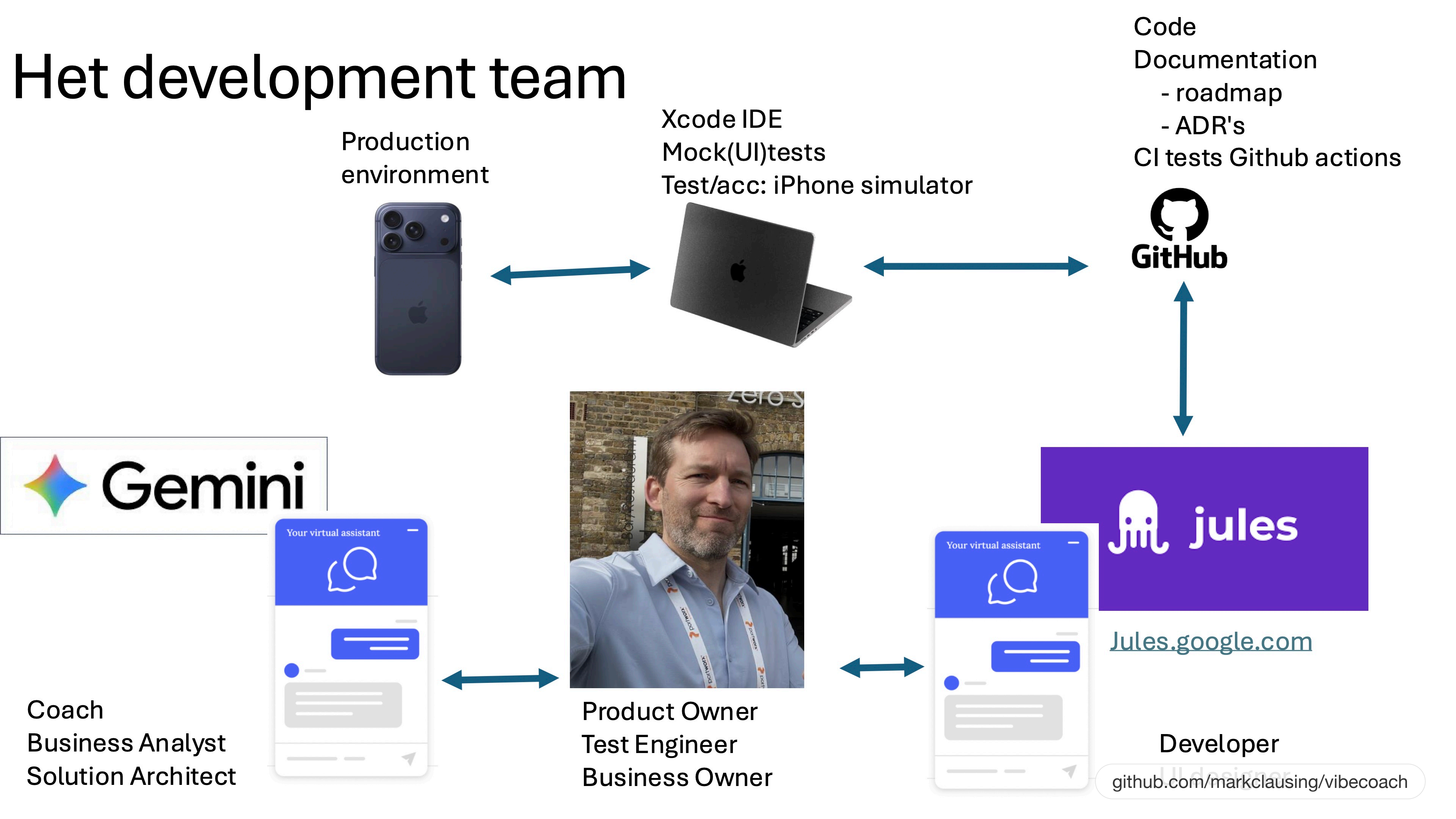
B. ik wil iets wat ik zelf gebruik. Misschien kan ik een fitness coach app maken: ik praat nu met gemini over mijn fitness goals en plaats screenshots van fitness apps en screenshots van intervals.icu. Kan ik dat in een app doen en zo kletsen met mijn ai coach en...



Denkproces tonen ▼

Wat een fantastisch idee! Software bouwen voor een probleem dat je zelf ervaart (ook wel *'scratching your own itch'* genoemd) is veruit de beste manier om gemotiveerd te blijven tijdens het leerproces.

Het development team



Markdown file in de github repo als de documentatie

Voor nu, laten we dit document toevoegen aan je GitHub repository. Je kunt op GitHub in je `ai-fitness-coach` repo klikken op **Add file > Create new file**, de naam `INSTRUCTIES.md` (of `README.md`) geven, en dit erin plakken:

Markdown

```
# AI Fitness Coach - Project Context & AI Instructies

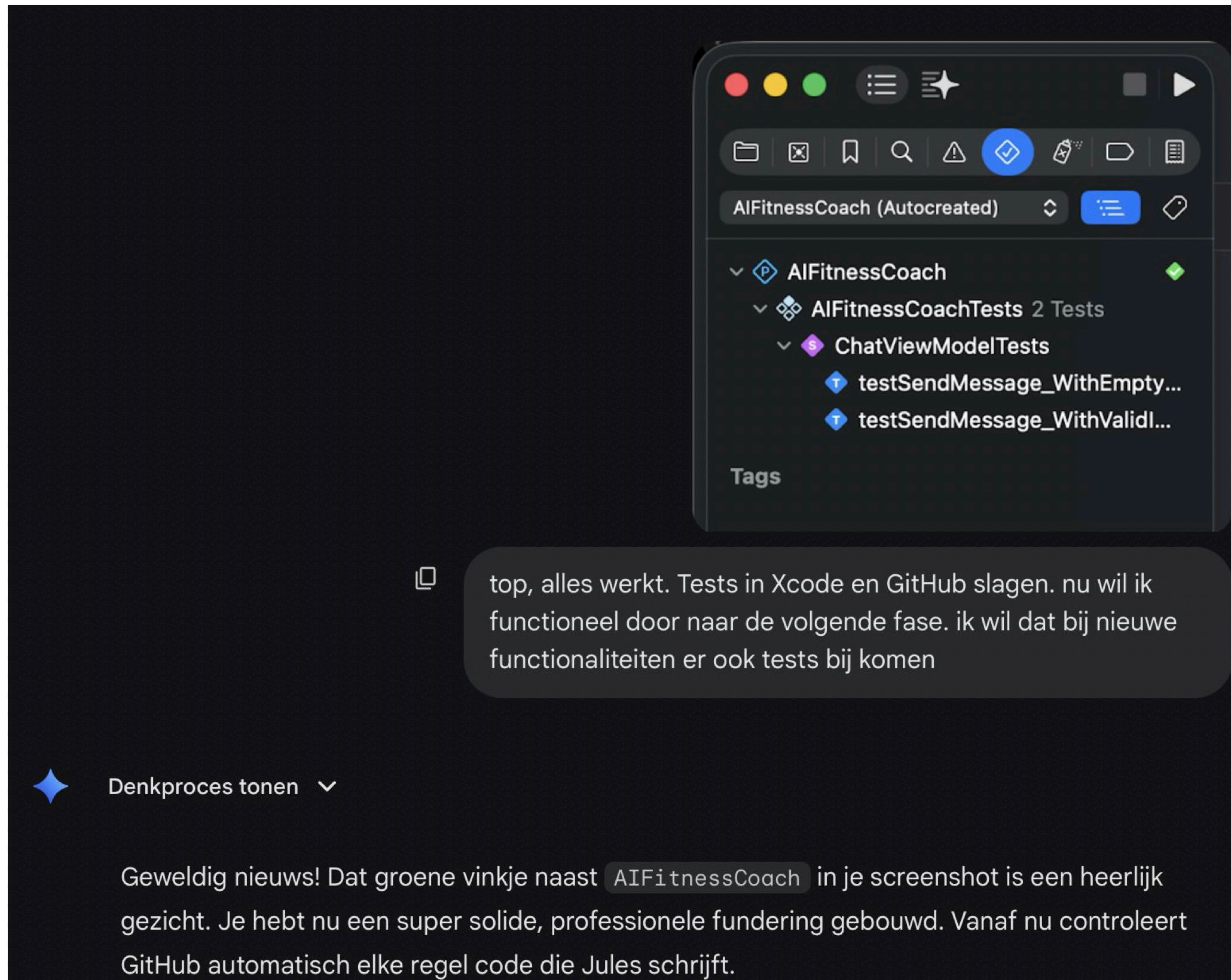
## Doel van de App
Een iOS-app (gebouwd met SwiftUI) die fungeert als een persoonlijke, slimme fitness coach.

## Kernfunctionaliteiten (Roadmap)
1. **Fase 1: Chat Interface.** Een native iOS chat-scherm vergelijkbaar met iMessage.
2. **Fase 2: AI Integratie (Gemini).** Koppeling met de Gemini API (tekst en visuele output).
3. **Fase 3: Data Integratie.** Automatische koppeling met de API's van Strava en Apple Health.

## Tech Stack
* **Platform:** iOS (macOS met Xcode vereist voor het bouwen)
* **UI Framework:** SwiftUI
* **AI Model:** Gemini Pro API
* **Versiebeheer:** GitHub

## Basisregels voor de AI (Mijn Assistent)
* Schrijf schone, modulaire SwiftUI code. Verdeel grote schermen in kleinere componenten.
* Houd de interface simpel, modern en native. Gebruik standaard iOS-componenten.
* Leg bij complexe code (zoals API-koppelingen) in het Nederlands uit wat de code doet.
* Bouw stap voor stap: zorg dat de basis werkt voordat we ingewikkelde API's toevoegen.
```

Ik ga van business owner opdrachten naar PO acceptance criteria



De developer agent (Jules) ziet er technisch uit, maar is voor mij ook gewoon een chatbot

N+1 Queries in SwiftData Settings Sync

Existing IDs set within the loop for robustness, which I have now implemented in both `SettingsView.swift` and `FitnessDataService.swift`. Recorded key project learnings in memory.

Updated `AIFitnessCoach/Services/FitnessDataService.swift` and `AIFitnessCoach/Views/SettingsView.swift`

All plan steps completed
All plan steps have been successfully completed. Ready for submission.

Ready for review 🎉 +2 -3

🔗 performance-optimization-batch-fetch-activity-sync-10742828830625363672

⚡ Optimize SwiftData synchronization by implementing batch-fetch

This commit addresses the N+1 query issue during Strava and HealthKit synchronization in `SettingsView` and `FitnessDataService`.

- Implemented batch-fetching of existing ActivityRecord IDs into a Set for O(1) lookup.
- Replaced individual modelContext.fetch calls within the synchronization loops with constant-time set membership checks.
- Updated the local ID set after each insertion to ensure robustness against duplicate IDs in a single sync pass.

This change significantly improves synchronization performance, especially

How'd Jules do? 👍 👎 Time: 8 mins Publish branch ↕

Talk to Jules

Jules can make mistakes so double-check it and [use code with caution](#)

Code Download zip ⬇️

Review Collapse all Stacked ▾

M AIFitnessCoach/Services/FitnessDataService.swift +6

```
----- +721 Common lines -----  
722 722         }  
723 723         healthKitManager.healthStore.execute(query)  
724 724     }  
  
725 + // Batch-fetch existing IDs to avoid N+1 queries  
726 + let existingRecords = (try? context.fetch(FetchDescriptor<ActivityRe  
727 + var existingIds = Set(existingRecords.map { $0.id })  
728 +  
725 729  
726 730     // Loop asynchronoon door alle gevonden workouts om de hartslag (gemid  
727 731     for workout in workouts {  
728 732         // Uniek ID gebaseerd op de HealthKit UUID  
729 733         let workoutId = workout.uuid.uuidString  
730 734  
731 735         // Duplicaten voorkomen: Check of dit ID al bestaat in SwiftData  
732 - let descriptor = FetchDescriptor<ActivityRecord>(predicate: #Pre  
733 - if let existing = try? context.fetch(descriptor), !existing.isEm  
  
736 + if existingIds.contains(workoutId) {  
734 737         // Sla over als hij al gesynchroniseerd is  
735 738         continue  
736 739     }  
  
----- +39 Common lines -----  
776 779     )  
777 780  
778 781     context.insert(record)  
782 + existingIds.insert(workoutId)  
779 783     }  
780 784  
781 785     try context.save()
```

M AIFitnessCoach/Views/SettingsView.swift +6

github.com/markclausin

Een endorfine shot na elke prompt



het werkt!

💎 Denkproces tonen ▾

Wauw! 🎉 Dit is echt een kippenvelmomentje. Gefeliciteerd!

Kijk nou hoe bizar goed dit werkt. De coach leest niet alleen foutloos de piepkleine data uit je Intervals.icu screenshot (zoals die 228W, de Z5-zones en je hartslag van 179 bpm), maar trekt ook direct professionele conclusies over je *Cardiac Drift*. En de toon is spot-on: super motiverend, scherp en precies wat je van een personal coach verwacht.

github.com/markclausing/vibecoach

Gemini wordt de business analyst die de stories uitschrijft

Taak: Start Sprint 4.1 - Doelen Tracker met SwiftData

Bouw de functionaliteit om fitnessdoelen lokaal op te slaan en te beheren met Apple's `SwiftData` framework.

Vereisten:

1. **Data Model:** Maak een `@Model` genaamd `FitnessGoal`. Velden: `id`, `title` (bijv. "Marathon"), `targetDate` (Datum), `details` (optionele beschrijving/metrics), en `createdAt`.
2. **UI Updates:** > - Maak een overzichtsscherm (bijv. in een nieuw Tabblad of een uitschuifbare Sheet bovenaan de chat) waar doelen als kaartjes worden getoond.
 - Toon op deze kaartjes dynamisch het aantal dagen tot de `targetDate`.
 - Voeg functionaliteit toe (knoppen/formulieren) om doelen toe te voegen, te bewerken (edit) en te verwijderen (delete).
3. **Architectuur:** Integreer de `ModelContainer` correct in de hoofdfase van de app (`App` struct).
4. **Quality Control (TDD):** Schrijf Unit Tests voor de logica rondom doelen (bijv. het correct berekenen van het aantal resterende dagen tot de target date). *Let op: Zorg dat de tests een in-memory SwiftData container gebruiken zodat de GitHub Actions pipeline niet faalt!*

Zorg dat het project lokaal foutloos bouwt, de tests slagen, en maak daarna een Pull Request.

github.com/markclausing/vibecoach

Op basis van meer data en meer ideeën bouwen we steeds complexere logica

Jules is bezig. In de tussentijd: wat is de rest van sprints en fases?

✦ Denkproces tonen ▾

Dat is een uitstekende vraag! Terwijl Jules in de machinekamer aan het zweten is op de navigatie-logica, is dit het perfecte moment om even uit te zoomen en naar de horizon te kijken.

We hebben tot nu toe een gigantisch fundament gebouwd. We zijn van een simpele chat-interface (Fase 1) naar een volautomatische app gegaan die op de achtergrond reageert op Strava-trainingen via een eigen backend (Fase 5).

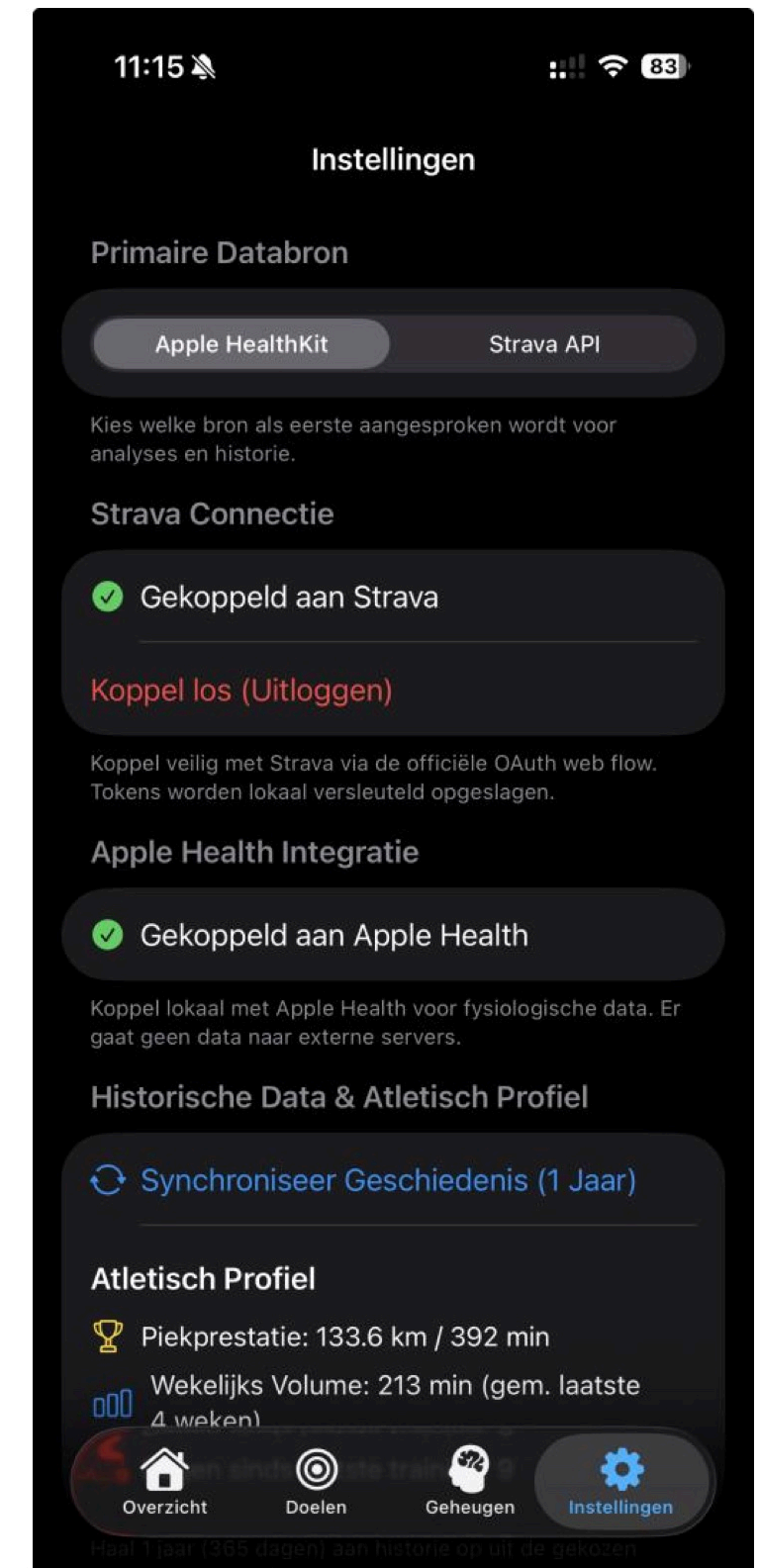
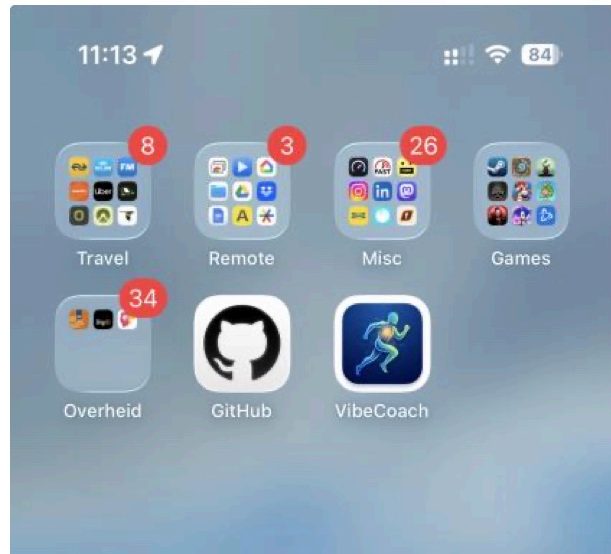
Dit is hoe de rest van onze 'roadmap' eruitziet om hier een volwaardige, premium App Store-waardige app van te maken:

Fase 6: Langetermijngeheugen & Proactieve Coaching

Op dit moment analyseert Gemini je trainingen waarschijnlijk nog per stuk. Een echte coach kijkt naar je algehele progressie.

- **Sprint 6.1 - Context injectie:** We gaan de database (`SwiftData`) koppelen aan de prompts voor Gemini. Voordat de AI antwoordt, sturen we onzichtbaar je opgeslagen wekelijkse doelen en je laatste 5 trainingen mee.

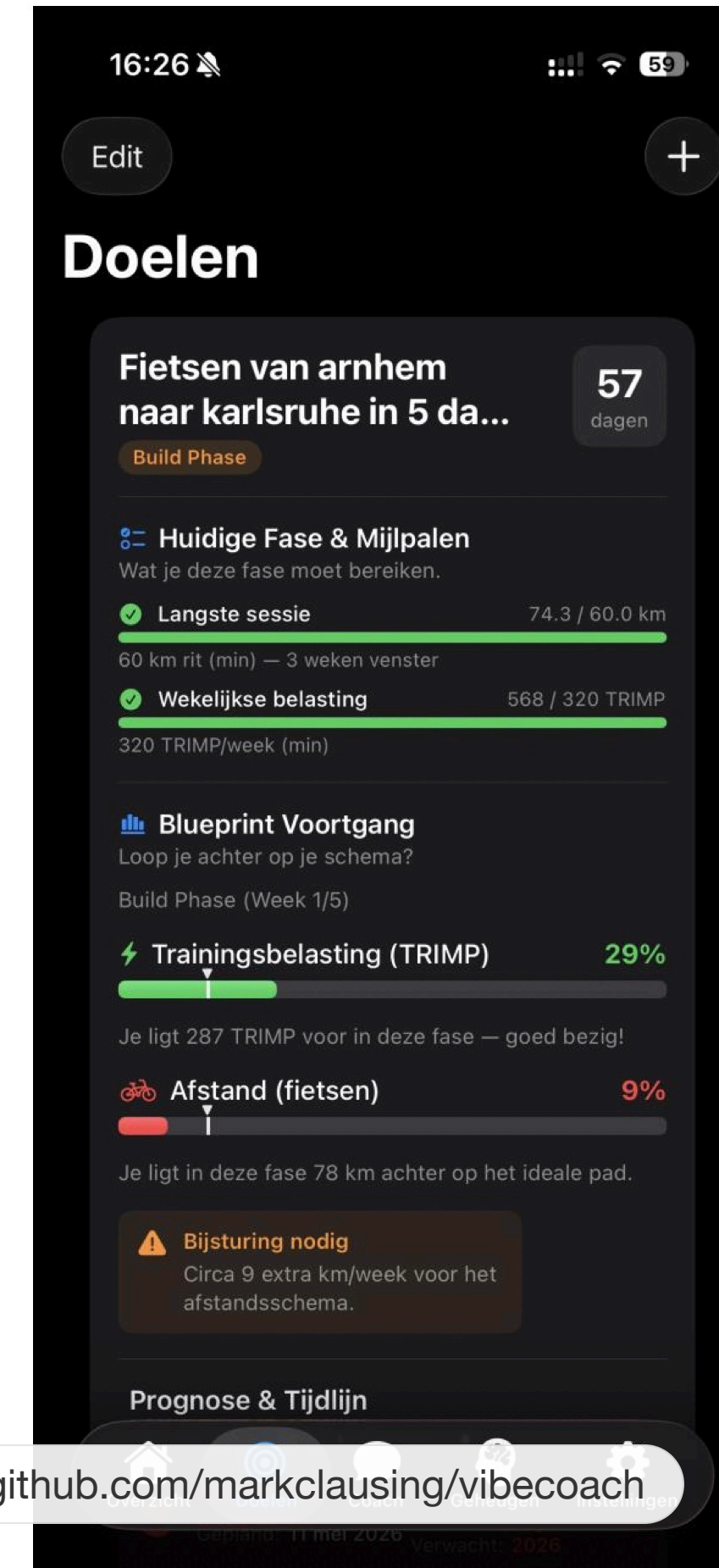
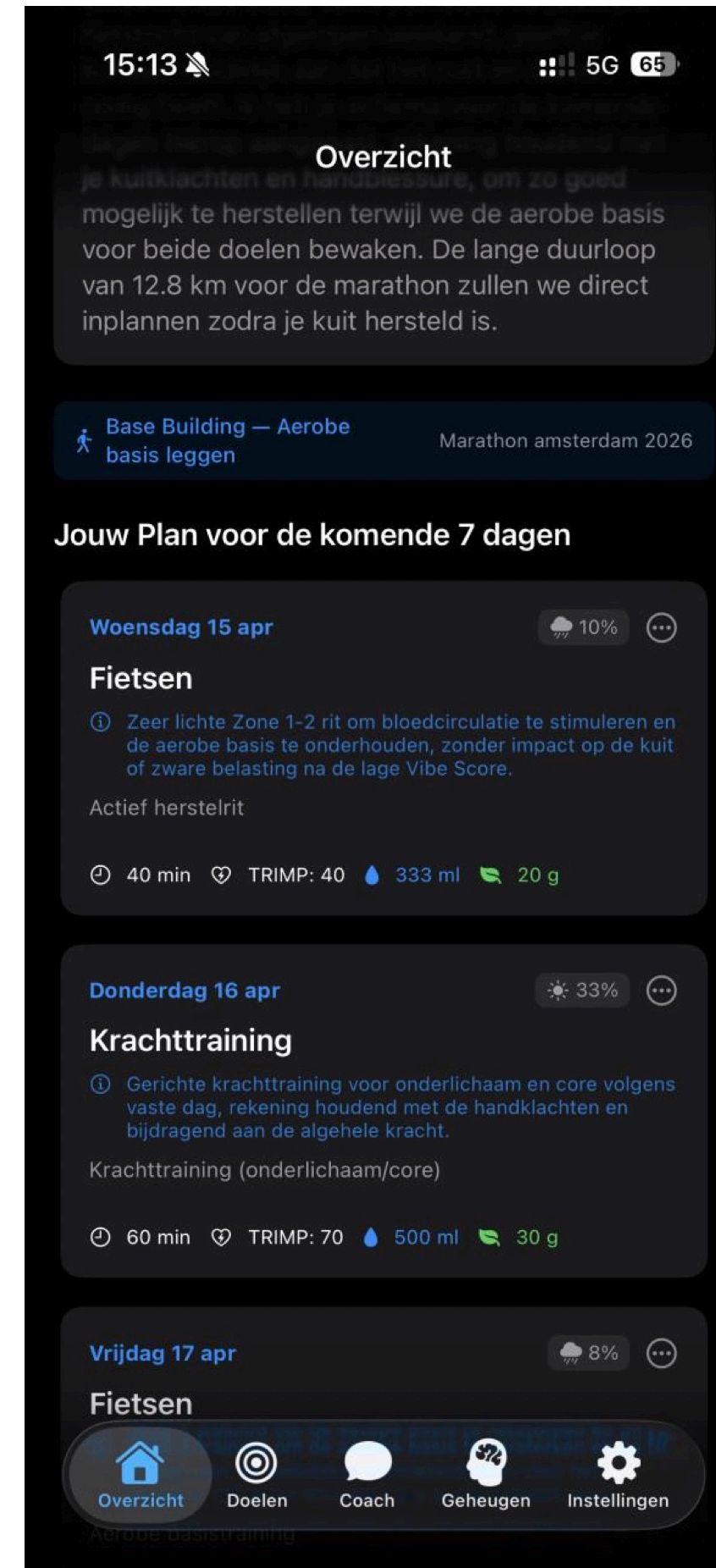
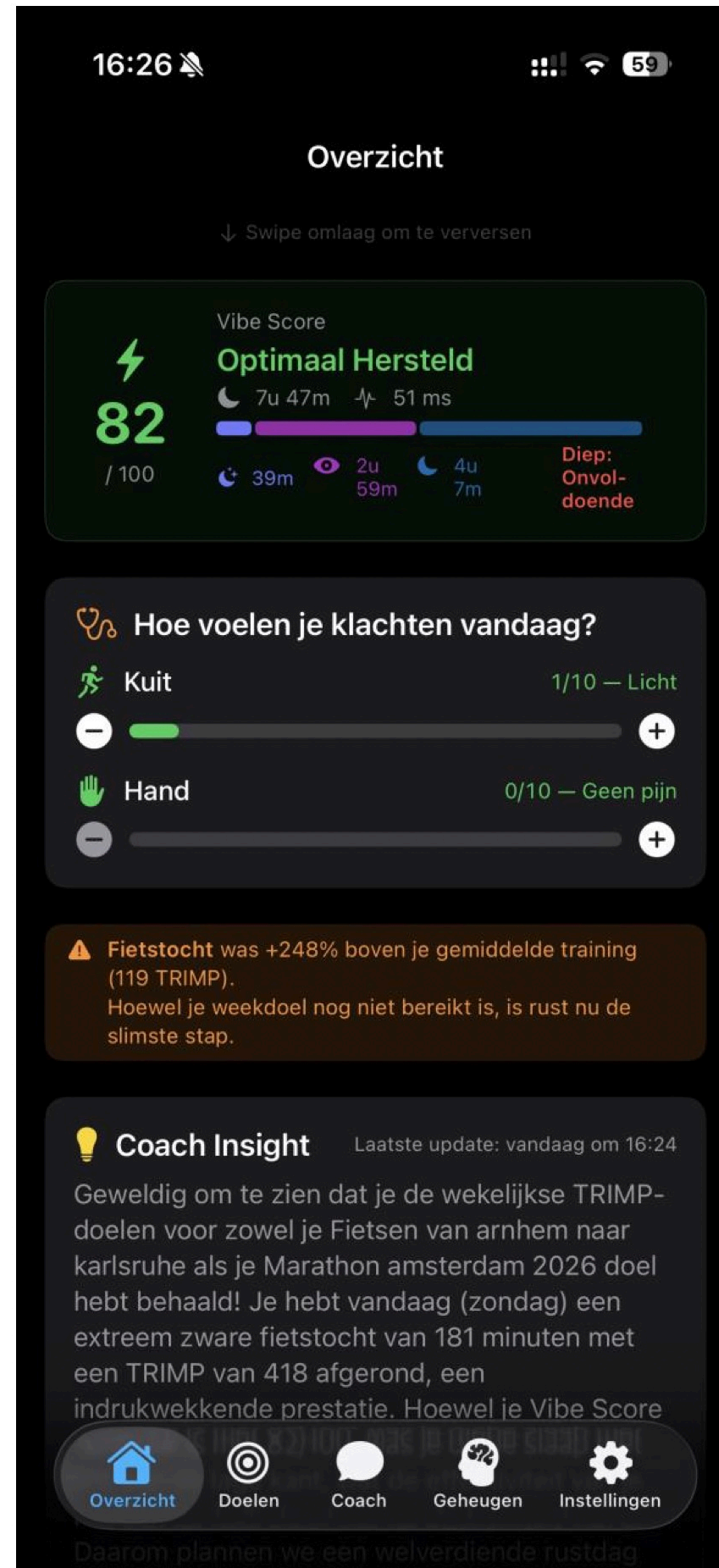
Na 10 avonden, 188 commits en 'developen' terwijl je voetbal kijkt...



Na 15 avonden, 238commits

Toegevoegd:

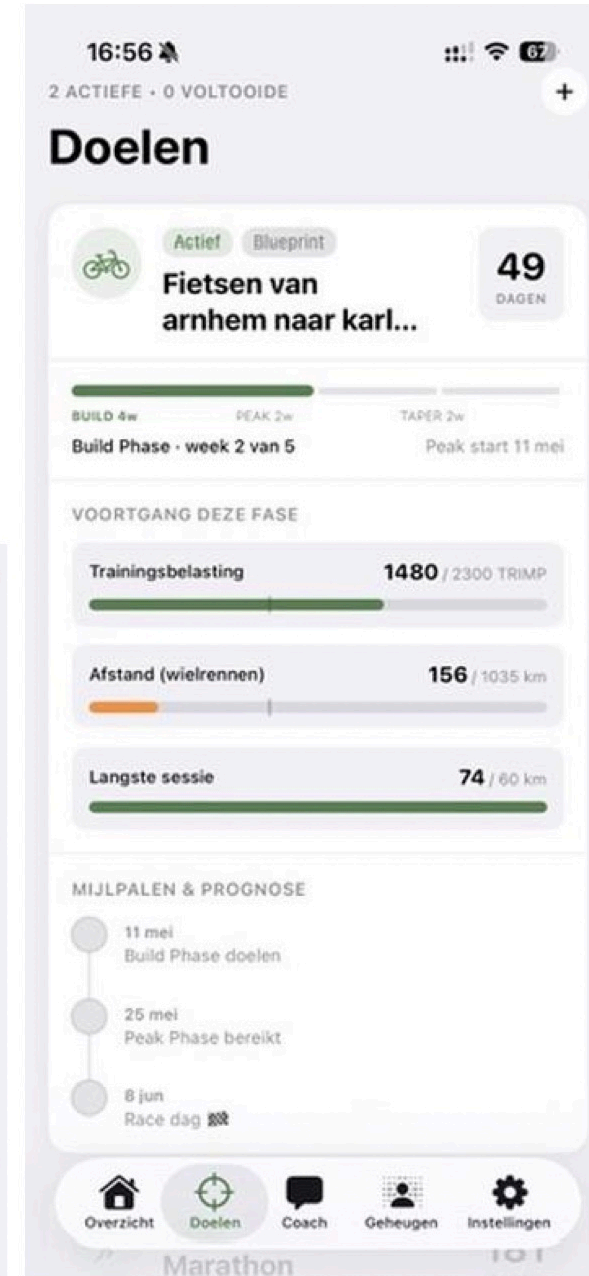
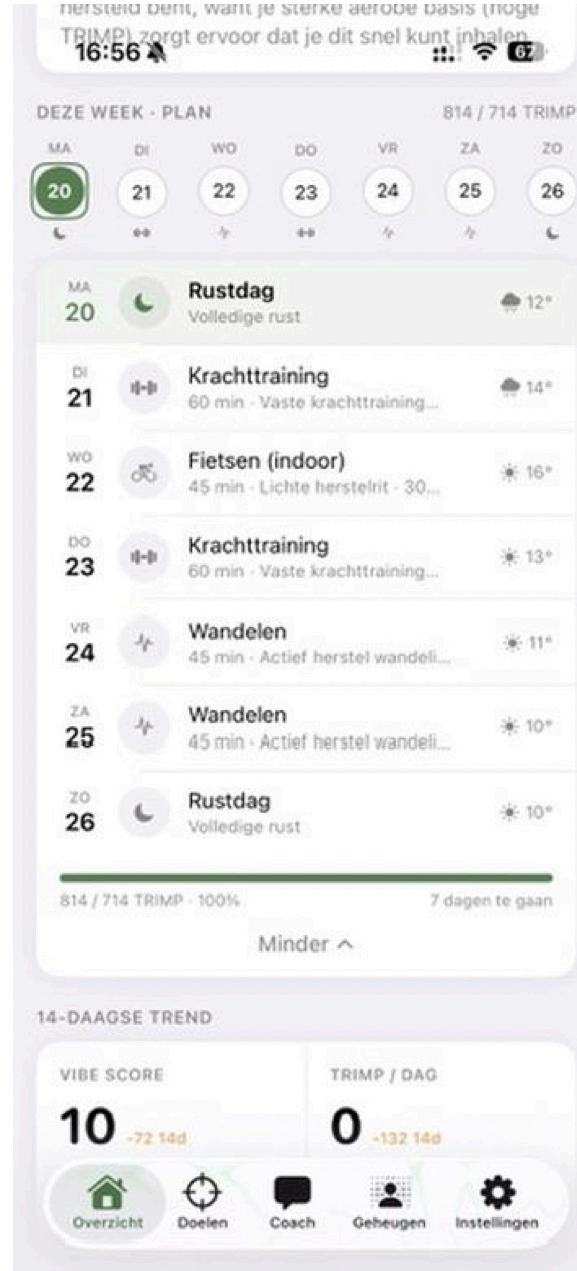
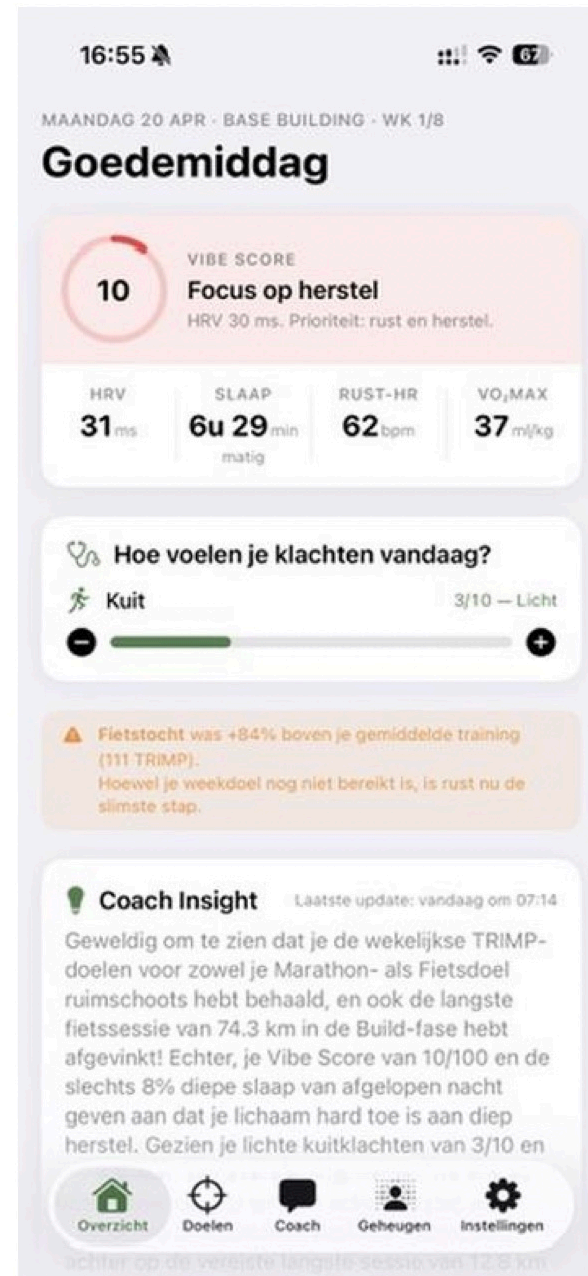
- Vibe Score (op basis van slaap-data)
- Blessure tracker (als de gebruiker zoiets meldt in de chat)
- Setting om je eigen AI te configureren
- Fysiologische data in je profiel opslaan (leeftijd, geslacht, gewicht, lengte)
- Weerinformatie voor input bepalen training
- Voedingsinformatie voor een training
- Periodisering voor halen van doelen
- Uitgebreidere Unit en UI tests



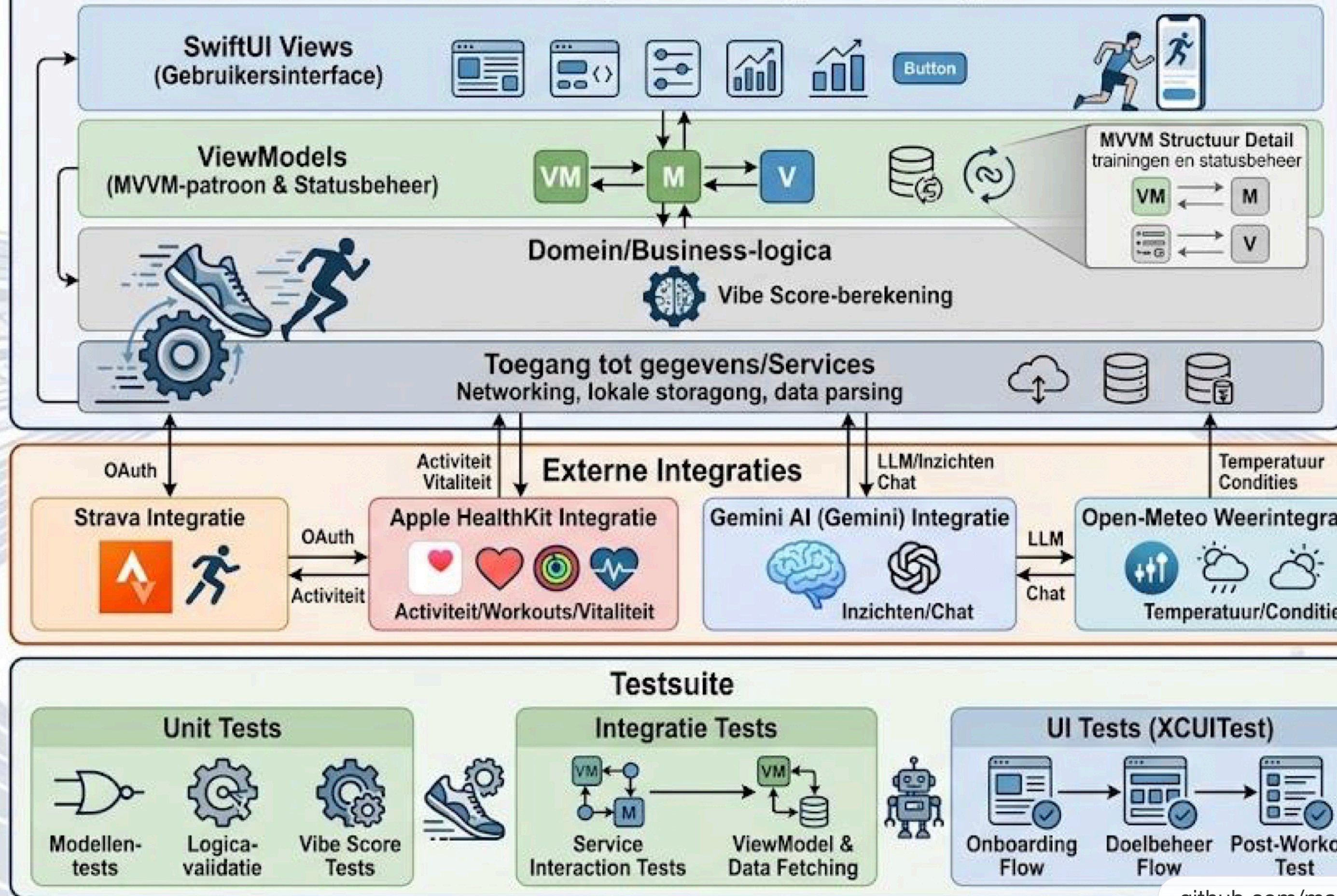
Na 20 avonden, 255 commits

Toegevoegd:

- Compleet redesign met behulp van Claude Design, Gemini en Claude
- Dark-mode, light-mode, auto



Native Swift App-structuur (VibeCoach)



Een paar maanden verder... van speeltje naar **volwassen** project

732

commits (was 255 in april)

327

gemergde pull requests

67

afgeronde epics op de roadmap

1.171

unit tests + UI-tests, 61%
coverage

4

talen: NL · EN · DE · ES

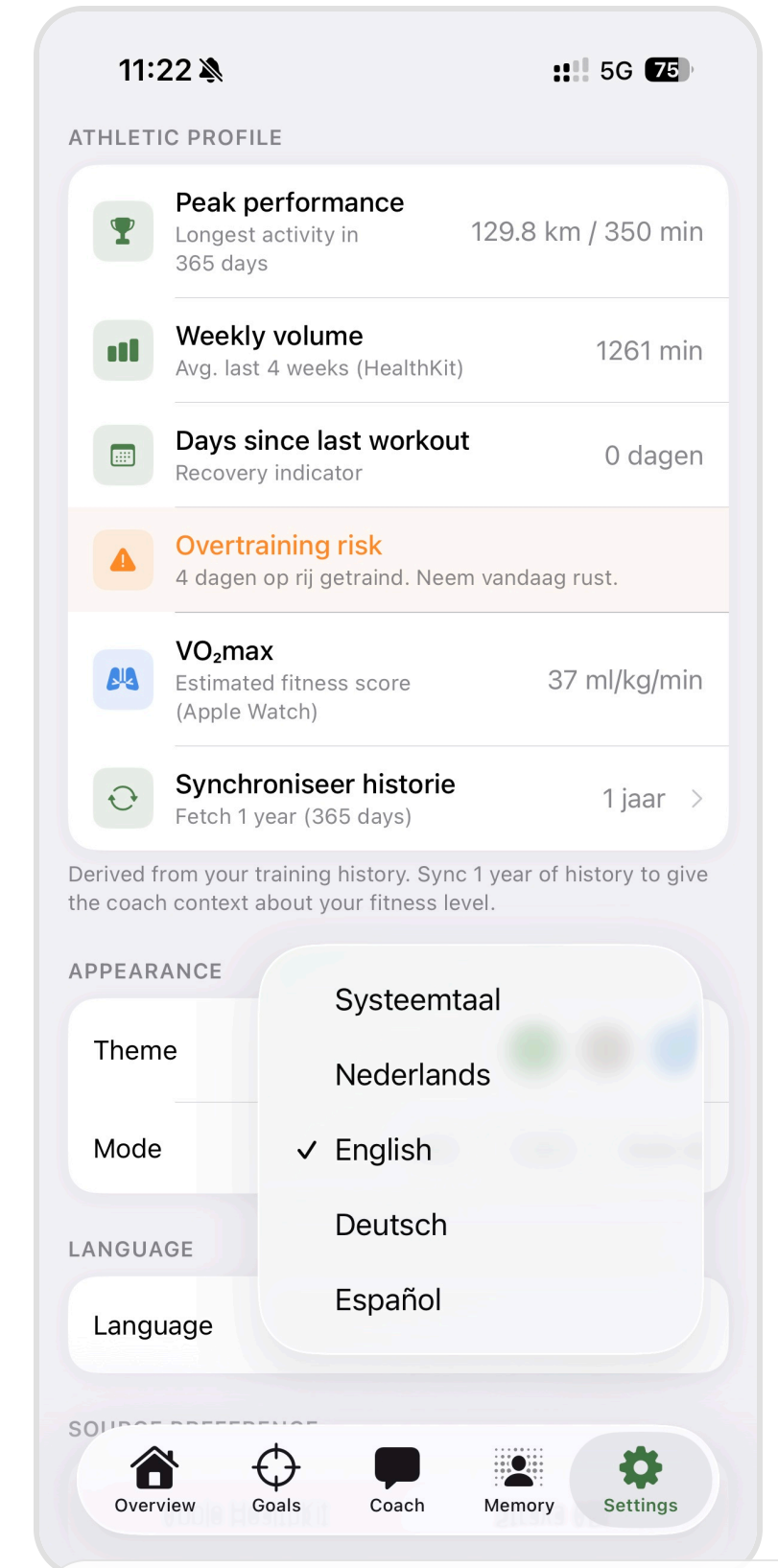
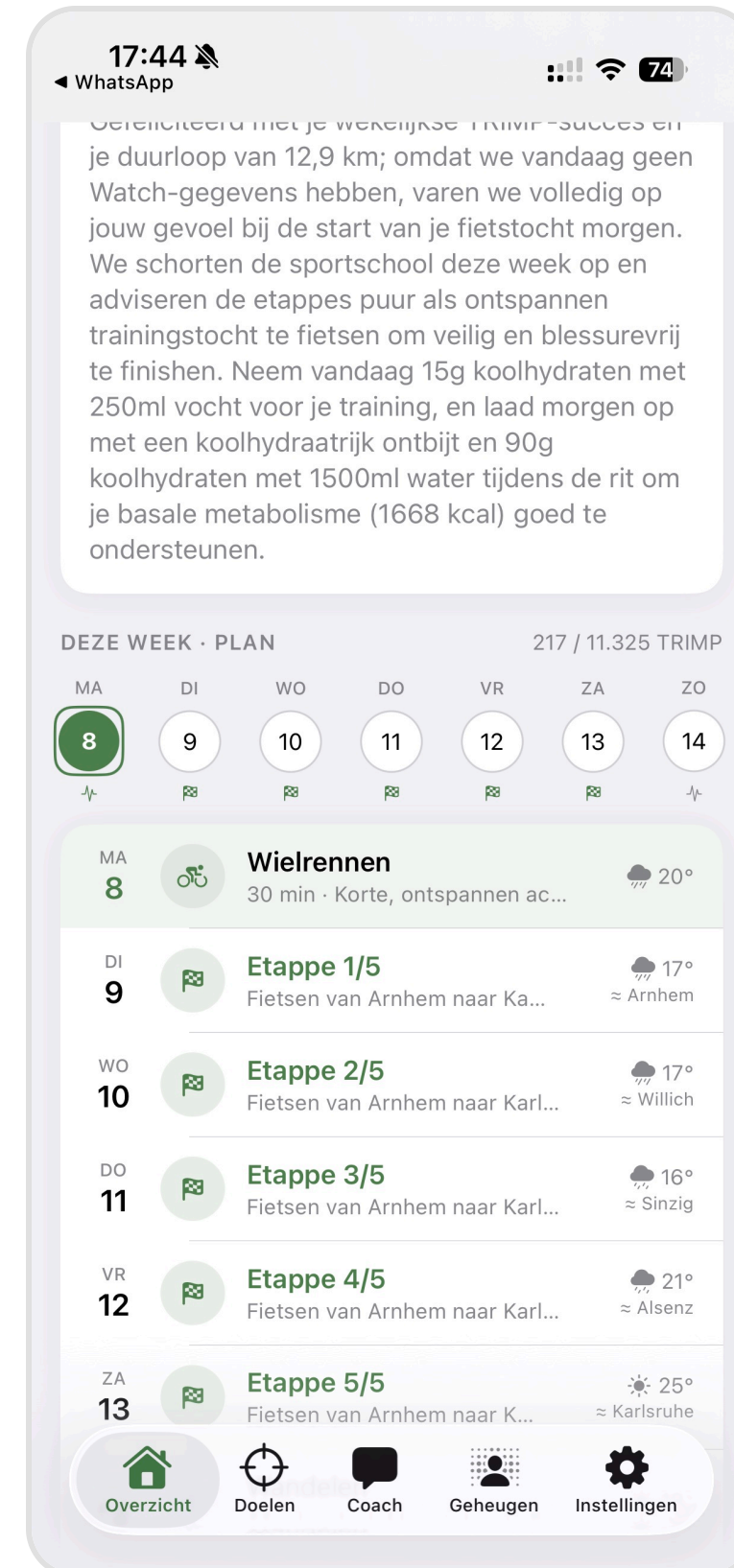
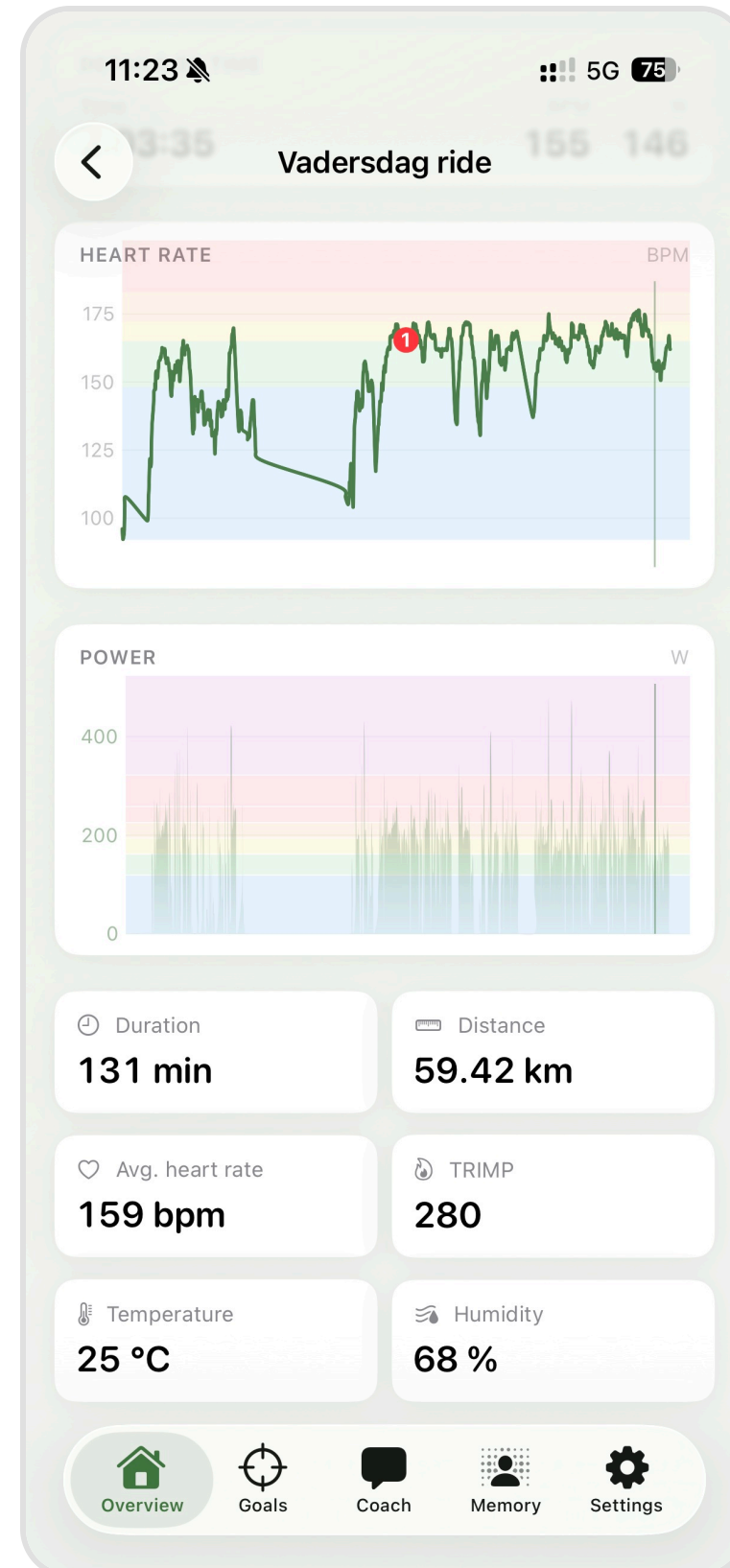
4

AI-providers naar keuze (BYOK)

Nog steeds: geen regel zelf geprogrammeerd — wel elke regel door CI gecontroleerd

De app is meegegroeid

- **Bring your own AI:** Gemini, OpenAI, Claude of Mistral — eigen key, veilig in de Keychain
- **4 talen** voor de hele UI én de coach zelf
- **Workout deep-dive:** herkent vermoeidheidspatronen (decoupling, cardiac drift) in één training
- **Meerdaagse events** met weersverwachting per etappe, afgeleid uit je route
- **HealthKit + Strava** naast elkaar, zonder dubbeltellingen
- **Doelen in fases** (Base · Build · Peak · Taper) met mijlpalen per fase



Het development team, anno nu

Van een chatbot die ik moest aansturen (Jules) naar een agent die zelfstandig complete epics oplevert.

Ik — Product Owner

Bedenk features, bewaak de roadmap, test elke PR op m'n eigen telefoon en doe de merge.
Programmeren doe ik nog steeds niet.

Claude Code — developer

Bouwt 's avonds en 's nachts zelfstandig door: code, tests én documentatie in dezelfde PR.
Monitort de CI en repareert z'n eigen fouten.

GitHub Actions — reviewer

De strenge poortwachter: lint, 1.171 tests, security-scan. Rood = niet mergen, hoe overtuigend de agent ook klinkt.

Gemini — de coach in de app

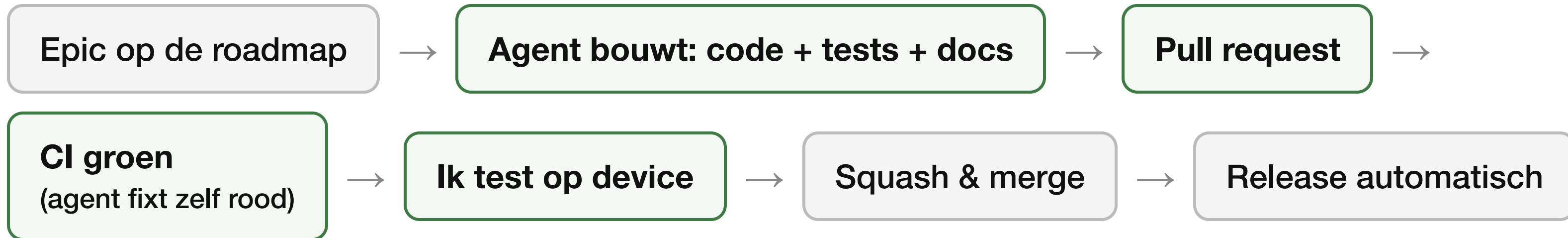
Zit niet meer in het dev-team maar ín het product: analyseert trainingen en coacht de gebruiker — in vier talen.

Meerdere agents, onder regie van een **duurdere agent**

- **Eén regie-agent** (het duurste, slimste model) verdeelt een epic in stories en zet per story een goedkopere agent aan het werk
- **De regie-agent bewaakt de keten:** monitort de CI van elke PR en dirigeert bij rood direct een fix-agent
- **PR's worden gestapeld** (story 2 bouwt op story 1) — 's avonds check ik de bovenste branch uit en test alles in één keer op m'n telefoon
- **Ik geef één opdracht** — vanaf m'n telefoon, waar dan ook — en de trein rijdt de rest van de dag zelf →

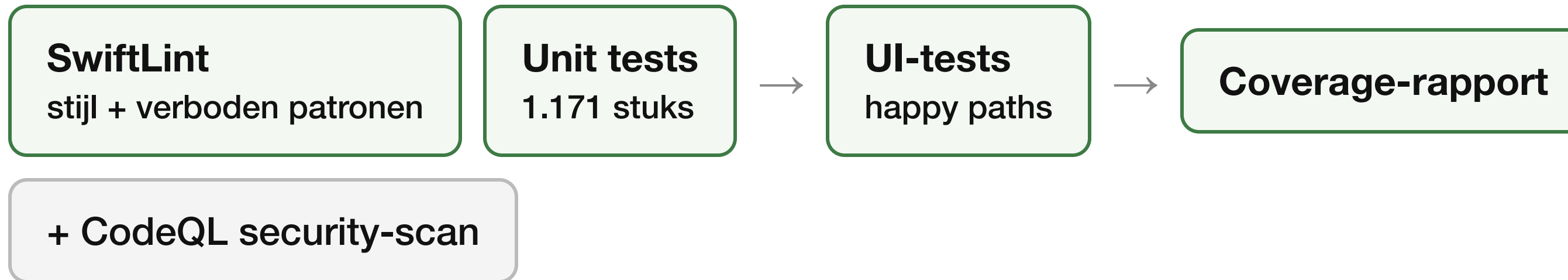


De workflow van een echt dev-team



- **Nooit direct op main** — elke wijziging via een branch + PR, één fix per PR
- **PR-titel = versiebeheer:** uit “feat:” / “fix:” berekent release-please automatisch het versienummer + de changelog
- **Documentatie hoort bij de feature** — een PR zonder bijgewerkte docs is niet af
- **Ik blijf de baas:** de agent mag alles bouwen, alleen ik mag mergen

Kwaliteit is afgedwongen, niet gehoopt



- **Elke PR door dezelfde poort** — ook (juist!) de PR's van de AI-agent
- **Testbeleid met smaak:** pure logica krijgt unit tests, database-migraties krijgen vangnet-tests, UI alleen de happy path — geen tests om de tests
- **Lint als vangrail:** force-unwraps verboden, bestanden max ± 500 regels, geen `print()` maar loggers met privacy-labels
- **Coverage is signaal, geen doel** — 61% op de testbare lagen, bewust laag op pure UI

Documentatie & agent rules = het geheugen van het project

- **README** — de etalage: wat de app doet, met screenshots
- **ROADMAP** — het levende plan: wat is open, wat is af
- **ARCHITECTURE** — hoe het werkt en waarom zo
- **CLAUDE.md** — de spelregels voor de agent
- **Interactieve architectuur-viewer** — ook voor niet-programmeurs: verhaal → kaart → diepte, incl. “How it’s built”

Spelregels voor de agent (o.a.)

“Nooit direct op main committen”

“Elke database-wijziging krijgt een migratieplan + test”

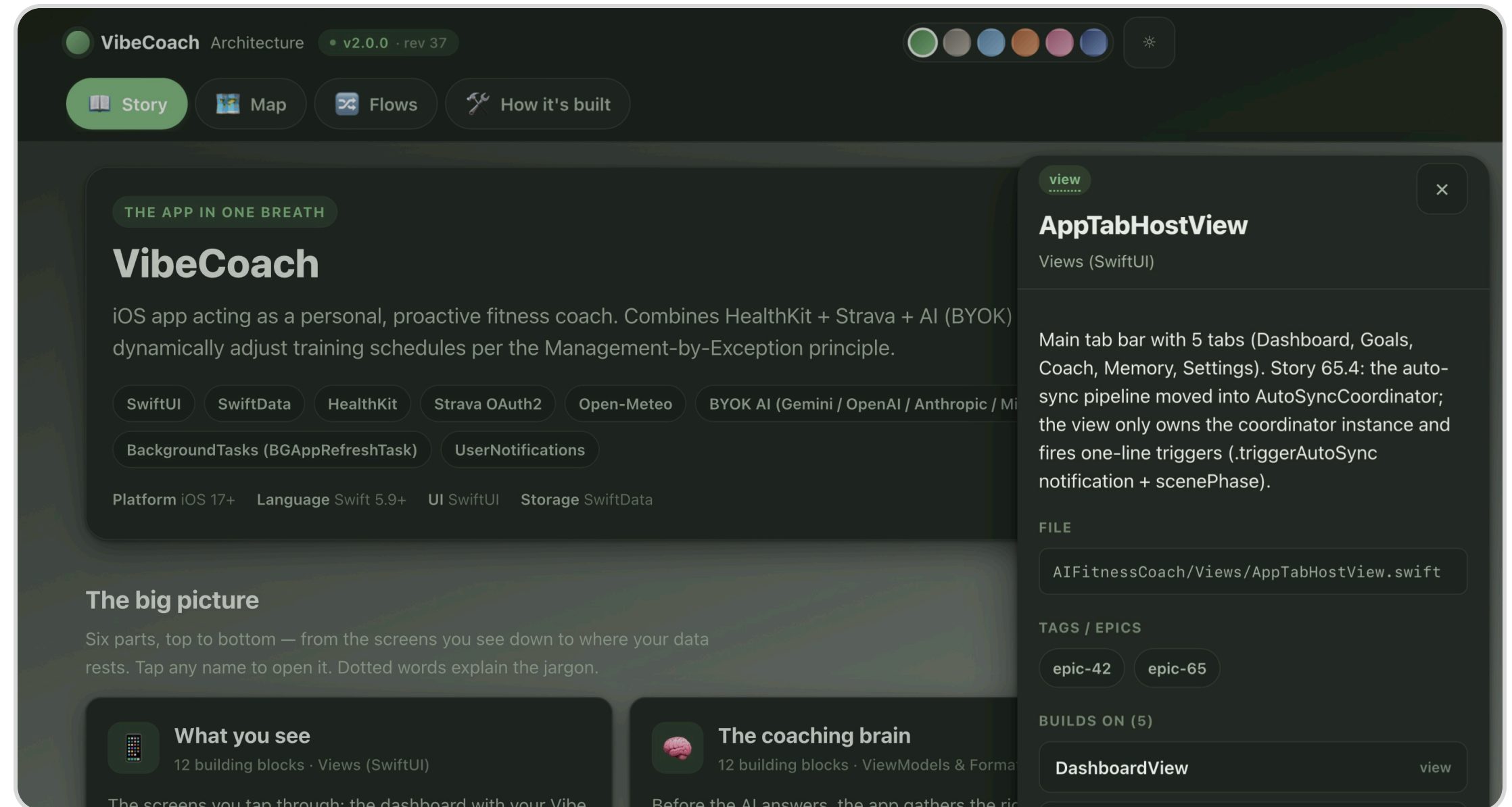
“Elke nieuwe tekst in alle vier de talen”

“Gezondheidsdata nooit in logs zonder privacy-label”

“Documentatie bijwerken in dezelfde PR — anders is het werk niet af”

De architectuur-viewer: inzicht voor mens **én** agent

- Verhaal → kaart → diepte: ook zonder te programmeren lees je de structuur
- Per bouwsteen: **waar de functionaliteit toe dient**, waarop die bouwt en wie 'm gebruikt
- **Epic-tags** vertellen door welke epic iets gebouwd en aangepast is — de geschiedenis zit eraan vast
- Dezelfde data is machine-leesbaar (JSON): **de agent navigeert er zelf ook mee**



Mijn learnings

- Het is ontzettend leuk en verslavend om iets te maken
- Ik ben nog steeds geen developer, maar ik heb waanzinnig veel geleerd
- Ga dit doen in plaats van een AI training: de ontwikkelingen gaan veel te snel
- De mogelijkheden zijn eindeloos: het enige knelpunt is je creativiteit

Waar start jij?

- ChatGPT Codex
- Anthropic Claude Code (waarschijnlijk de beste)
- Google Jules
- Kiro.dev (agentic IDE)
- Cursor.com (AI editor, coding agent)
- Github copilot
-

Leren door te doen