

Het ontwikkelen van een native iOS app

Een fiets- en hardlooptracker maken

Willem L. Middelkoop

11 mei 2024



Als klein zijprojectje, gepropt tussen mijn normale werk, ben ik bezig geweest met iets wat me persoonlijk interesseert: een native workout tracking app voor iOS. Ik wilde mijn smartwatch overbodig maken en in plaats daarvan mijn telefoon gebruiken om trainingen bij te houden. Hoe moeilijk kan het zijn om gedetailleerde sensordata te verzamelen met native Swift API's?

Achtergrond

Als je mijn blog hebt gevolgd, zal dit geen verrassing zijn, maar ik hou ervan om te [fietsen](#) en ben onlangs begonnen met [hardlopen](#). Ik heb aardig wat ervaring met het bijhouden van mijn trainingen, met behulp van Garmin, [Strava](#), [Omata](#), Wahoo, [Biostrap](#), [Whoop](#)

en verschillende [apps](#) en [sensoren](#). Ik vind gezondheids- en fitnessgegevens nuttig om te leren over je [fysieke prestaties](#) en om een soort van verantwoording aan jezelf af te leggen (bijv. om je te motiveren om door te gaan met de trainingen).



Niets maakt je zo levend als naar buiten gaan voor wat fysieke actie - ik hou van sporten!

De nerd in mij houdt van de grote dataset, de vele verschillende beschikbare sensortypes zijn een feest voor mijn nieuwsgierigheid - maar het conflicteert met mijn verlangen naar [minimalisme](#) en eenvoud. [Heb je echt alle trainingsgegevens nodig?](#) Is [alleen naar je lichaam luisteren niet genoeg?](#) Ik weet het niet, maar ik waardeer wel de mogelijkheid van een smartwatch om (bijna) moeiteloos trainingen in detail bij te houden. Ik vind het prettig hoe [een Apple Watch kan worden gebruikt om trainingssessies bij te houden](#), naast de mogelijkheden als [modern tool watch](#) en [smartphone vervanger](#). Hoewel ik begrijp dat de meeste mensen hun smartwatch zouden dragen en het daarbij laten, houd ik van [ouderwetse mechanische horloges](#) zoals een [Rolex](#), [Tudor](#), [Grand Seiko](#), of zelfs [iets op maat gemaakt](#). Hoewel ik beide waardeer, neig ik naar [mechanische horloges](#) als ik moet kiezen tussen [smartwatches](#) en [mechanische horloges](#).

Idee voor de app: Gran Fondo

Wat als er een smartphone-app was die alle trainingsgegevens zou verzamelen die je wilt, zonder gedoe en rommel? Ik stelde me voor dat het:

- **Eenvoudig:** het moet 'uit de weg gaan', ontworpen om 'gewoon te werken', zonder dat je hoeft te knoeien met eindeloze instellingen of opties. Druk gewoon op 'start' en begin je training.
- **Flexibel:** Gebruik de app op verschillende manieren: bevestig of draag je telefoon in het zicht voor realtime gegevens, of stop je telefoon in je zak en laat hem op de achtergrond werken.

- **Krachtig:** Verbind met hartslag-, cadans-, vermogens- en snelheidssensoren via Bluetooth; ondersteuning voor meerdere sensoren, ideaal als je verschillende fietsen gebruikt.
- **Apple Health-vriendelijk:** Sla alle gegevens in Apple Health op in maximaal detail, inclusief trainingroutes en individuele vermogens-, cadans- en snelheidsmetingen. Hiermee kun je opnames analyseren in elke app die Apple Health ondersteunt, zoals Strava, WHOOP, HealthFit en vele anderen.
- **Eerlijk:** Geen advertenties of verkoop van gegevens. Geen accounts of registratie. Je gegevens moeten privé, veilig en op je apparaat blijven.

Geen van de bestaande smartphone sport- en fitness-apps voldoet aan al het bovenstaande. De meeste zijn onnodig complex of zijn op de een of andere manier ontworpen om je gegevens te gebruiken om de app-bouwer of zijn platform te dienen. Kon ik het beter doen?

De app bouwen

Wetende waarom en wat ik wilde, ging ik op zoek naar mijn voorkeursmethode om de app te bouwen.

Native vs Hybrid/web apps

Voor het meeste werk dat ik doe, geef ik de voorkeur aan open en gratis (als in libre) webtechnologieën, zoals HTML, JavaScript en CSS - of serversoftware die is uitgebracht onder een [vrije softwarelicentie](#), zoals GPL. Voor deze specifieke usecase moest ik extreem energiezuinig zijn en diep kunnen integreren met het besturingssysteem van mijn smartphone. De beste manier om dit te doen voor een iPhone is om de software development kit van Apple volledig te omarmen: de app native maken in Swift met behulp van SwiftUI en door de [technische](#) en [ontwerp richtlijnen](#) zo nauwkeurig mogelijk te volgen.

AI-geassisteerd programmeren

Hoewel [ik mijn reserves heb](#), heeft het gebruik van AI voor programmeren de laatste tijd alle aandacht. Sommigen beweren zelfs dat technologieën zoals Copilot of ChatGPT programmeurs overbodig zouden maken. Ik weet mijn weg te vinden in AI, aangezien ik het heb gebruikt om [code te genereren](#), zoals [ik mensen leer hoe ze het moeten gebruiken](#), maar ik heb het nooit gebruikt om een *complete* iOS-app te bouwen. Ik dacht dat dit een perfecte gelegenheid zou zijn om te zien hoe geschikt AI is voor iets complexers dan een eenvoudig scriptfragment. Laten we eens kijken hoe overbodig ik ben als programmeur in dit moderne AI-tijdperk...

GPS en Bluetooth Low Energy Sensoren

Nadat ik mijn Xcode-omgeving had opgezet, vroeg ik ChatGPT van OpenAI om me wat code te geven om GPS-locatiegegevens te lezen en om de nodige Swift-klassen te genereren om Bluetooth-sensoren te detecteren en er verbinding mee te maken. Man, wat ben ik daar in het spreekwoordelijke konijnenhol terechtgekomen!



Mijn fiets met Bluetooth-sensoren verbonden met mijn MacBook met Xcode

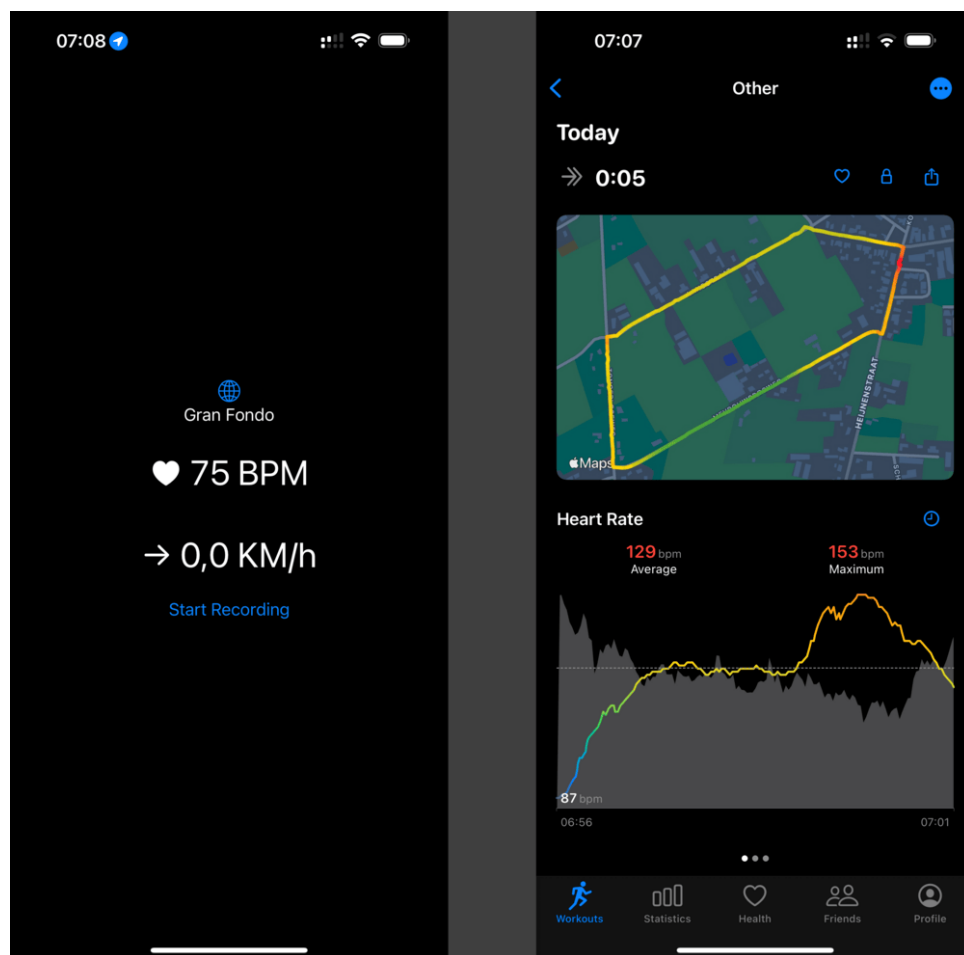
Het ding met AI-geassisteerd programmeren is dat, **JA** het werkt; maar slechts **GEDEELTELIJK**. Het geeft je werkende stukken, maar ze passen niet perfect en werken niet noodzakelijkerwijs samen. Het zien van de code die het genereert is als het horen van echo's van een geweldig liedje; je herkent dat het ergens op lijkt - maar het is niet helemaal hetzelfde als de band live muziek horen spelen; laat staan het zelf zingen!

Ik nam de stukken van de AI en ging zelf de relevante documentatie lezen, dit bleek veel effectiever omdat ik daardoor daadwerkelijk kon begrijpen wat het mechanisme zou moeten doen. De AI hielp me de dingen te vinden die ik moest leren. Het bijhouden van GPS-locatiegegevens was verrassend eenvoudig met behulp van [Apple's CoreLocation](#). Het verbinden van de Bluetooth-sensor met de app bleek veel moeilijker, omdat het veel verschillende stappen omvat:

- **Bluetooth-apparaten detecteren:** scan de omgeving op apparaten die beschikbaar zijn en uitzenden
- **Verbinden met een Bluetooth-apparaat:** een verbinding *maken en onderhouden* tussen de app en het apparaat
- **Bluetooth-apparaatkenmerken ontdekken:** er bestaat niet zoiets als een eenvoudig Bluetooth-apparaat, je moet zelf 'ontdekken' wat de mogelijkheden en functies van een apparaat zijn; veel apparaten zenden meerdere verschillende signalen uit voor zowel sensorgegevens, bedieningsmechanismen als operationele parameters zoals batterijniveau.
- **Abonneren op kenmerken:** Uiteindelijk abonneer je je op een datafeed door de Bluetooth-radio's af te stemmen op een bepaald Bluetooth-kenmerk.
- **Gegevens decoderen:** Gezien de energiezuinige aard ontvang je minimale gegevens in ruwe bits; je moet de datapakketten zelf decoderen; het moeilijke is dat elke sensor, van elke leverancier, zijn eigen manier heeft om gegevens te coderen.

Neem bijvoorbeeld een relatief 'eenvoudige' sensor zoals een cadanssensor die je op het pedaal van je fiets plaatst, deze kan worden gebruikt om het aantal omwentelingen per minuut te bepalen dat je maakt tijdens een fietssessie. Hoe moeilijk kan dit zijn? Ik dacht dat de [officiële Bluetooth specificatie](#) me alle benodigde details zou geven... NIET! Een eenvoudige cadanssensor is verrassend complex:

- **Gecombineerd Bluetooth-profiel voor fietssnelheid en cadans:** elke cadanssensor identificeert zich **altijd** als *zowel* snelheids- *als* cadanssensor, je moet de mogelijkheden 'on the fly' ontdekken.
- **Niet-gelijke datapakketten:** De inhoud van de daadwerkelijke gegevens hangt af van de huidige mogelijkheden van de sensor (bijv. het heeft iets gedetecteerd), je moet het datapakket, bit voor bit, interpreteren door te controleren op mogelijke vlaggen die aangeven wat de volgende datab



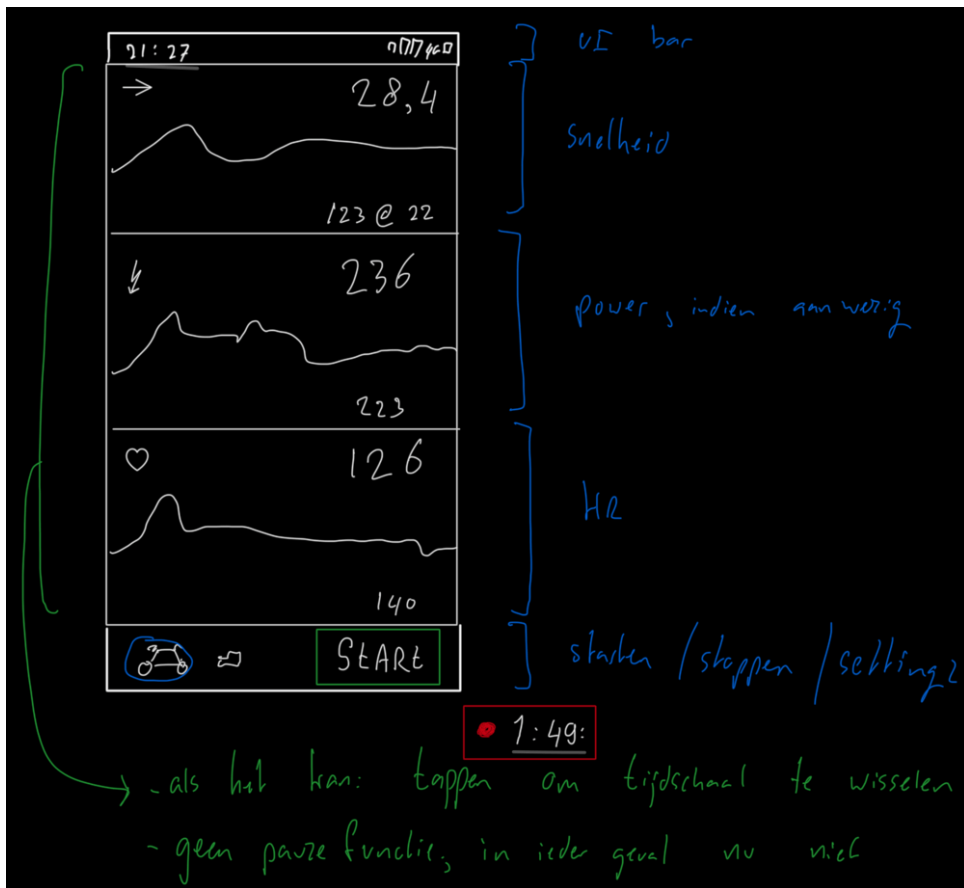
Vroeg prototype van de app werkend met Bluetooth-hartslagsensor en de vroege dataset weergegeven in een andere app die mooie grafieken maakt (HealthFit)



Apple Watch workout-app toont grote knoppen om een workout te starten - geen extra configuratie nodig



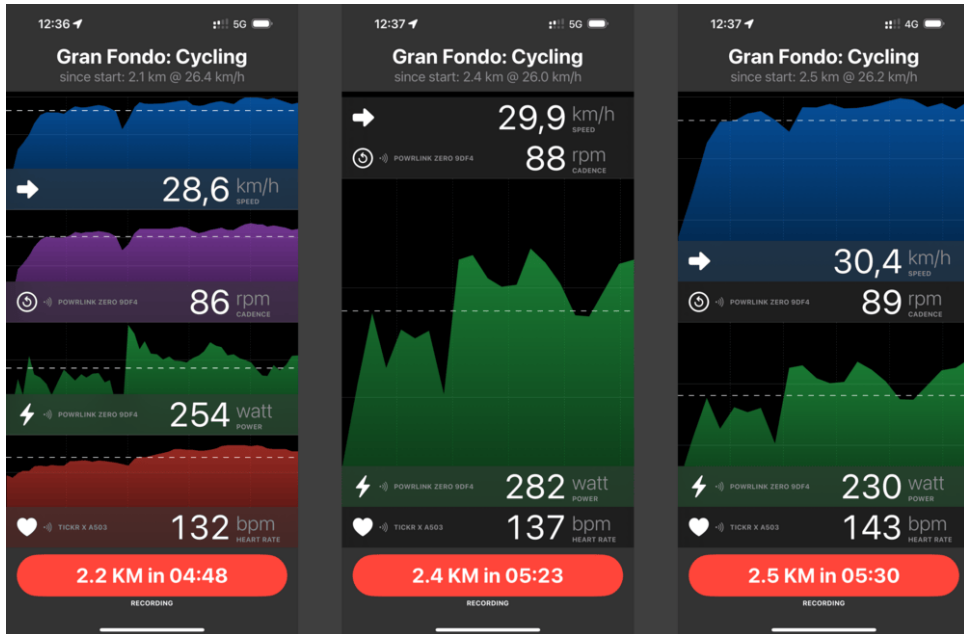
Ik heb mijn Garmin Edge data layout als inspiratie gebruikt bij het ontwerpen van de layout van mijn app wanneer deze gebruikt wordt als gemonteerde fietscomputer



Ontwerpschets die de gewenste layout beschrijft



Het testen van de app op mijn fiets met behulp van een QuadLock hoesje en mount



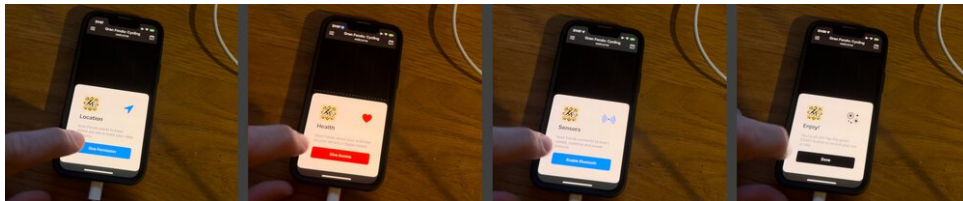
Verschillende configuraties van realtime data



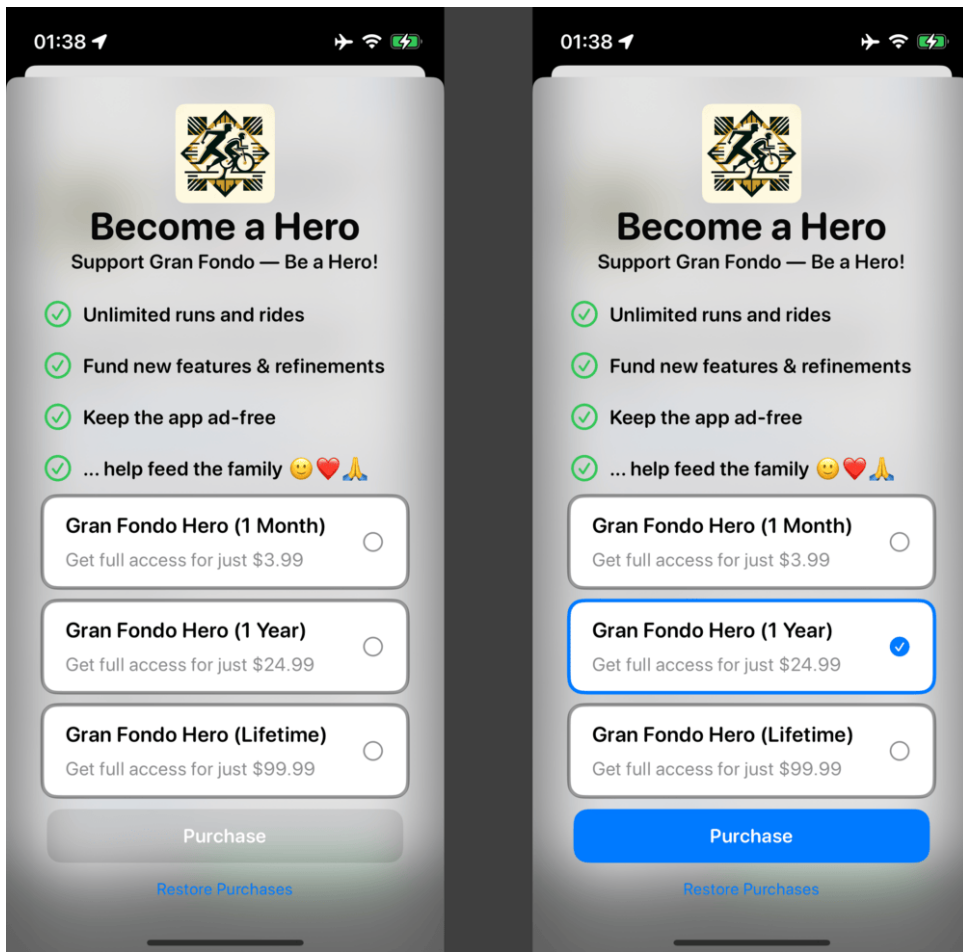
Ik heb wat magie in mijn zak: De app die workouts volgt vanuit de achtergrond - tijdens opnames toont het gewoon een widget op het startscherm



"Goin' the extra mile": 8 uur op mijn fiets om het batterijgebruik te testen (let op: er is nog 23% stroom over!)



Screenshots van het onboarding-proces van de app: toestemming vragen en nieuwe gebruikers begeleiden



Het ontwerpen van een betaalmuur voor mijn app: niet mijn favoriete vraag aan gebruikers, maar het is desalniettemin een zeer belangrijke.



Niet je gemiddelde debuging sessie: Fietselfstedentocht 2024