Desarrollo de una app nativa para iOS

Creando un rastreador de ciclismo y carrera

Willem L. Middelkoop
May 11, 2024

Como un pequeño proyecto paralelo, realizado entre mi trabajo normal, he estado trabajando en algo de interés personal: una aplicación nativa de seguimiento de entrenamiento para iOS. Quería hacer que mi reloj inteligente fuera obsoleto, usando en su lugar mi teléfono para registrar los entrenamientos. ¿Qué tan difícil podría ser recopilar datos detallados de los sensores usando las API nativas de Swift?

Antecedentes

Si has estado siguiendo mi blog, esto puede no ser una sorpresa, pero me encanta andar en bicicleta y recientemente comencé a correr. Tengo bastante experiencia en el seguimiento

de mis entrenamientos, utilizando Garmin, Strava, Omata, Wahoo, Biostrap, Whoop y varias aplicaciones y sensores. Encuentro los datos de salud y estado físico útiles para aprender sobre tu rendimiento físico y para tener algún tipo de responsabilidad contigo mismo (por ejemplo, motivarte a seguir haciendo los entrenamientos).

Nada te hace sentir tan vivo como salir a hacer algo de ejercicio - ¡Me encanta entrenar!

El nerd que hay en mí ama el conjunto de big data, los muchos tipos de sensores disponibles son un festín para mi curiosidad, pero entra en conflicto con mi deseo de minimalismo y simplicidad. ¿Realmente necesitas todos los datos de entrenamiento? ¿No es suficiente con escuchar a tu cuerpo? No lo sé, pero aprecio la capacidad de un reloj inteligente para rastrear (casi) sin esfuerzo los entrenamientos con gran detalle. Me gusta cómo se puede usar el Apple Watch para rastrear sesiones de entrenamiento, además de sus capacidades como reloj de herramientas moderno y reemplazo de teléfono inteligente. Si bien entiendo que la mayoría de las personas usarían su reloj inteligente y lo dejarían ahí, me gustan los relojes mecánicos a la antigua como un Rolex, Tudor, Grand Seiko, o incluso algo hecho a medida. Aunque aprecio ambos, me inclino por los relojes mecánicos cuando tengo que elegir entre relojes inteligentes y relojes mecánicos.

Idea para la aplicación: Gran Fondo

¿Qué pasaría si hubiera una aplicación para teléfonos inteligentes que recopilara todos los datos de entrenamiento que deseas, sin ninguna molestia ni desorden? Me la imaginé así:

- **Simple**: debe ”quitarse de tu camino”, diseñada para ”simplemente funcionar”, sin la necesidad de jugar con un sinfín de configuraciones u opciones. Simplemente presiona ”inicio” y comienza tu entrenamiento.

- **Flexible**: Usa la aplicación de diferentes maneras: monta o usa tu teléfono a la vista para obtener datos en tiempo real, o pon tu teléfono en tu bolsillo y haz que funcione en segundo plano.
- **Potente**: Conéctate con sensores de frecuencia cardíaca, cadencia de ciclismo, potencia y velocidad mediante Bluetooth; admite múltiples sensores, ideal si usas diferentes bicicletas.
- **Compatible con Apple Health**: Guarda todos los datos en Apple Health con el máximo detalle, incluidas las rutas de entrenamiento y las mediciones individuales de potencia, cadencia y velocidad. Esto te permite analizar grabaciones en cualquier aplicación compatible con Apple Health, como Strava, WHOOP, HealthFit y muchas otras.
- **Honesta**: Sin anuncios ni venta de datos. Sin cuentas ni registro. Tus datos deben permanecer privados, seguros y en tu dispositivo.

Ninguna de las aplicaciones deportivas y de fitness existentes para teléfonos inteligentes cumple con todo lo anterior. La mayoría de ellas son innecesariamente complejas o están diseñadas de alguna manera para tomar tus datos para servir al creador de la aplicación o su plataforma. ¿Podría hacerlo mejor?

Construyendo la aplicación

Sabiendo por qué y qué quería, me propuse elegir mi método preferido para construir la aplicación.

Aplicaciones nativas vs. híbridas/web

Para la mayor parte del trabajo que hago, prefiero usar tecnologías web abiertas y gratuitas (como en libre), como HTML, JavaScript y CSS, o software de servidor publicado bajo una licencia de software libre, como GPL. Para este caso de uso en particular, necesitaba ser extremadamente eficiente energéticamente y poder integrarme profundamente con el sistema operativo de mi teléfono inteligente. La mejor manera de hacerlo para un iPhone es adoptar por completo el kit de desarrollo de software de Apple: crear la aplicación de forma nativa en Swift utilizando SwiftUI y siguiendo las pautas técnicas y de diseño lo más fielmente posible.

Programación asistida por IA

Aunque tengo mis reservas, el uso de la IA para la programación ha llamado mucho la atención últimamente. Algunos incluso afirman que tecnologías como Copilot o ChatGPT harían que los programadores fueran obsoletos. Conozco la IA, ya que la he utilizado para generar código, ya que enseño a la gente cómo usarla, pero nunca la utilicé para construir una aplicación de iOS *completa*. Pensé que esta sería una oportunidad perfecta para ver qué tan adecuada es la IA para algo más complejo que un simple fragmento de código. Averigüemos qué tan obsoleto estoy como programador en esta era moderna de la IA...

GPS y sensores Bluetooth de baja energía

Después de configurar mi entorno Xcode, me propuse pedirle a ChatGPT de OpenAI que me diera algo de código para leer datos de ubicación GPS y generar las clases Swift

necesarias para detectar y conectar con sensores Bluetooth. ¡Vaya, si entré en la proverbial madriguera del conejo aquí!

Mi bicicleta con sensores Bluetooth conectados a mi MacBook ejecutando Xcode

Lo que pasa con la programación asistida por IA es que, **SÍ**, funciona; pero solo **PARCIALMENTE**. Te dará piezas que funcionan, pero no encajarán perfectamente y no necesariamente funcionarán juntas. Ver el código que genera es como escuchar ecos de una gran canción; reconocerás que se parece a algo, pero no es exactamente lo mismo que escuchar a la banda tocar música en vivo; ¡mucho menos cantarla tú mismo!

Tomé las piezas de la IA y me dispuse a leer la documentación relevante yo mismo, esto resultó ser mucho más efectivo ya que me permitió comprender realmente lo que debía hacer el mecanismo. La IA me ayudó a encontrar las cosas que necesitaba aprender. El seguimiento de los datos de ubicación GPS fue sorprendentemente fácil utilizando CoreLocation de Apple. Hacer que el sensor Bluetooth se conectara a la aplicación resultó ser mucho más difícil, ya que implica muchos pasos distintos:

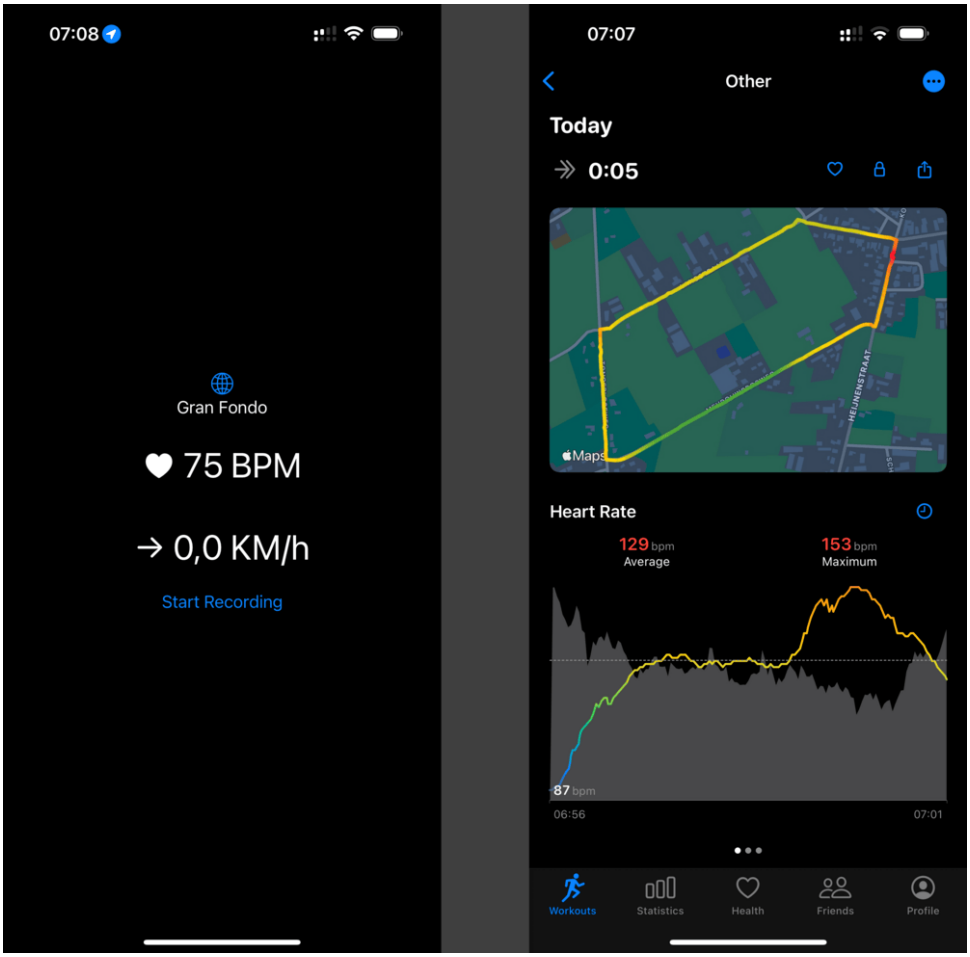
- **Detectar dispositivos Bluetooth**: escanear el área cercana en busca de cualquier dispositivo que esté disponible y transmitiendo.
- **Conectarse a un dispositivo Bluetooth**: establecer *y mantener* una conexión entre la aplicación y el dispositivo.
- **Descubrir las características del dispositivo Bluetooth**: no existe un dispositivo Bluetooth simple, tienes que hacer tu propio "descubrimiento" de las capacidades y características de un dispositivo; muchos dispositivos transmiten múltiples señales diferentes tanto para datos de sensores como para mecanismos de control y parámetros operativos, como el nivel de batería.
- **Suscribirse a las características**: en última instancia, te suscribes a una fuente de datos sintonizando las radios Bluetooth a una determinada característica de Bluetooth.

- **Decodificar datos**: dada su naturaleza de eficiencia energética, recibirás datos mínimos en bits sin procesar; tendrás que decodificar los paquetes de datos tú mismo; la parte difícil es que cada sensor, de cualquier proveedor, tiene su propia forma de codificar los datos.

Tomemos, por ejemplo, un sensor relativamente "simple" como un sensor de cadencia que colocas en el pedal de tu bicicleta, que podría usarse para determinar las revoluciones por minuto que realizas durante una sesión de ciclismo. ¿Qué tan difícil puede ser esto? Pensé que la especificación oficial de Bluetooth me proporcionaría todos los detalles necesarios... ¡NO! Un simple sensor de cadencia es sorprendentemente complejo:

- **Perfil de Bluetooth combinado para velocidad y cadencia de ciclismo**: cualquier sensor de cadencia **siempre** se identifica como sensor de velocidad *y* cadencia, tienes que descubrir sus capacidades "sobre la marcha".
- **Paquetes de datos no similares**: el contenido de los datos reales depende de las capacidades actuales del sensor (por ejemplo, detectó algo), tendrás que interpretar el paquete de datos, bit por bit, comprobando posibles indicadores que indiquen qué representa el siguiente bit de datos.
- **Datos no triviales**: no esperes un valor claro para "cadencia", el sensor te dará un recuento acumulativo de revoluciones **y** una marca de tiempo rudimentaria del último cambio detectado; tendrás que usar estos valores para calcular la métrica que estás intentando medir.

La IA me proporcionó algún código de muestra, pero resultó que no funcionaba con mi sensor en particular. Usando una aplicación de depuración de Bluetooth, Bluefy, pude comprender mejor los datos que transmitía el sensor. Encontré un fragmento de código Java en el blog oficial de Bluetooth que me dio algunas pistas sobre cómo interpretar los datos de los sensores. Adapté este fragmento de código para que coincidiera con varios perfiles de sensores diferentes y lo traduje a código Swift. Esto me dio un prototipo funcional temprano.



Prototipo inicial de la app funcionando con un sensor de ritmo cardíaco Bluetooth y sus primeros datos mostrados en otra app que hace gráficos bonitos (HealthFit)

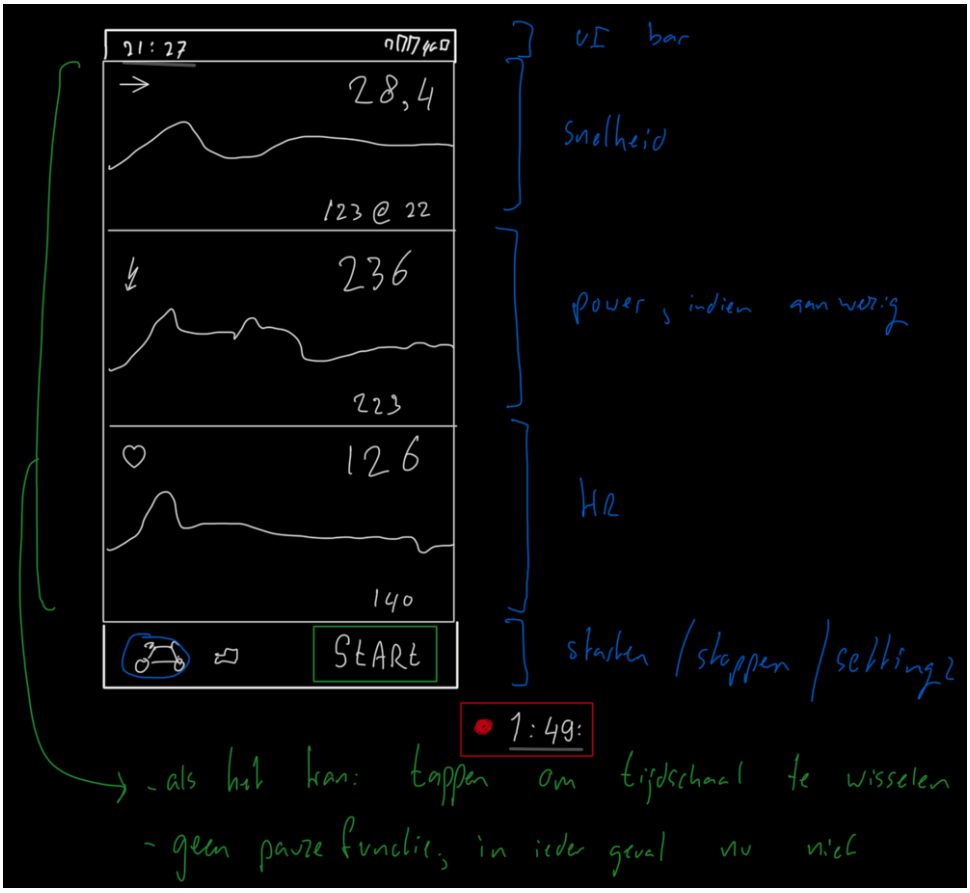
Diseño de UI/UX

Una de las razones principales por las que me gusta la aplicación de entrenamiento del Apple Watch es cómo está diseñada su interfaz de usuario (UI) para facilitar una experiencia de usuario (UX) agradable. La aplicación en el reloj tiene botones bastante grandes que presionas para iniciar una grabación, *y eso es prácticamente todo.* La aplicación en sí tiene pocas opciones, su diseño es obstinado pero bien pensado. Es un marcado contraste con dispositivos como Garmin que ofrecen un montón de opciones de personalización. El truco consiste en encontrar un buen equilibrio entre la flexibilidad en los casos de uso *y* tener una configuración predeterminada que funcione bien.

La app de entrenamiento del Apple Watch muestra botones grandes para iniciar un entrenamiento - sin necesidad de configuración adicional

Utilizando un sistema de montaje como QuadLock, puedes fijar firmemente tu teléfono inteligente al manillar de tu bicicleta. Esto significa que el iPhone puede reemplazar tu Garmin como ciclocomputador. Los iPhones modernos tienen una pantalla siempre encendida que puede mostrar información de forma eficiente energéticamente, siempre que utilices el kit de desarrollo de software de Apple que viene con varias optimizaciones integradas. Después de muchos, muchos, kilómetros con mi ciclocomputador Garmin, me decidí por tener una vista con campos de datos en tiempo real para la velocidad, la potencia y la frecuencia cardíaca. Lo que me gusta de los dispositivos Garmin Edge más avanzados es que pueden trazar un gráfico. Esto te permite mantener la vista en la carretera cuando importa y echar un vistazo a los datos históricos en una etapa (ligeramente) posterior.

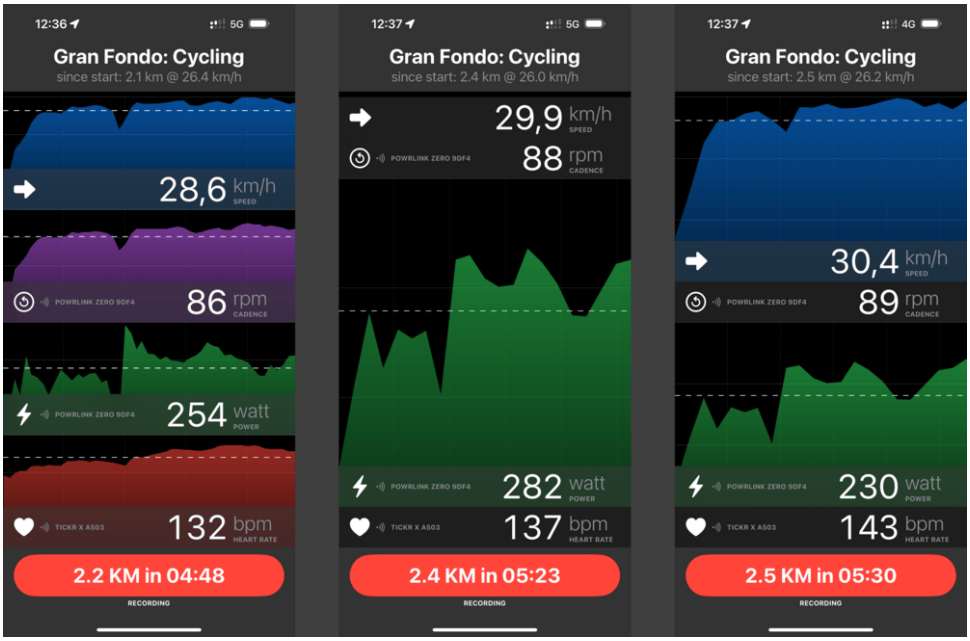
Usé el diseño de datos de mi Garmin Edge como inspiración al diseñar el diseño de mi app cuando se usa como ciclocomputador



Boceto de diseño que describe el diseño deseado

Probando la app en mi bicicleta usando una carcasa y soporte QuadLock

Para maximizar la eficiencia energética, decidí utilizar las propias bibliotecas de Apple para gráficos: Swift Charts. Un problema que encontré al desarrollar esto es que las mediciones de los sensores llegan en diferentes intervalos y momentos. Necesitas crear algún tipo de mecanismo de sincronización para que los puntos de datos coincidan con marcas de tiempo comunes para poder comparar y relacionar valores. Diseñé este mecanismo para sincronizar también las actualizaciones de la interfaz de usuario: permitiendo que todos los gráficos y puntos de datos se actualicen en una sola actualización; optimizando el rendimiento de la CPU y los gráficos (y, por lo tanto, la energía de la batería). Como beneficio adicional, incorporé compatibilidad con diferentes ámbitos de tiempo, lo que permite que los gráficos muestren datos del último minuto, 5 minutos, 15 minutos, hora o entrenamiento completo.



Diferentes configuraciones de datos en tiempo real

La aplicación muestra automáticamente los gráficos disponibles en función de los sensores conectados, lo que evita la necesidad de diferentes configuraciones cuando tienes más de una bicicleta. Si tocas un gráfico, se maximiza o minimiza. Si tocas el ámbito de datos (por ejemplo, "últimos 5 minutos"), puedes alternar entre ámbitos. La aplicación recuerda tu configuración, configurándote automáticamente de forma correcta para el próximo entrenamiento. Debido a que todos los objetivos táctiles y los botones son bastante grandes, funcionan bien cuando tienes las manos sudorosas o si usas guantes.

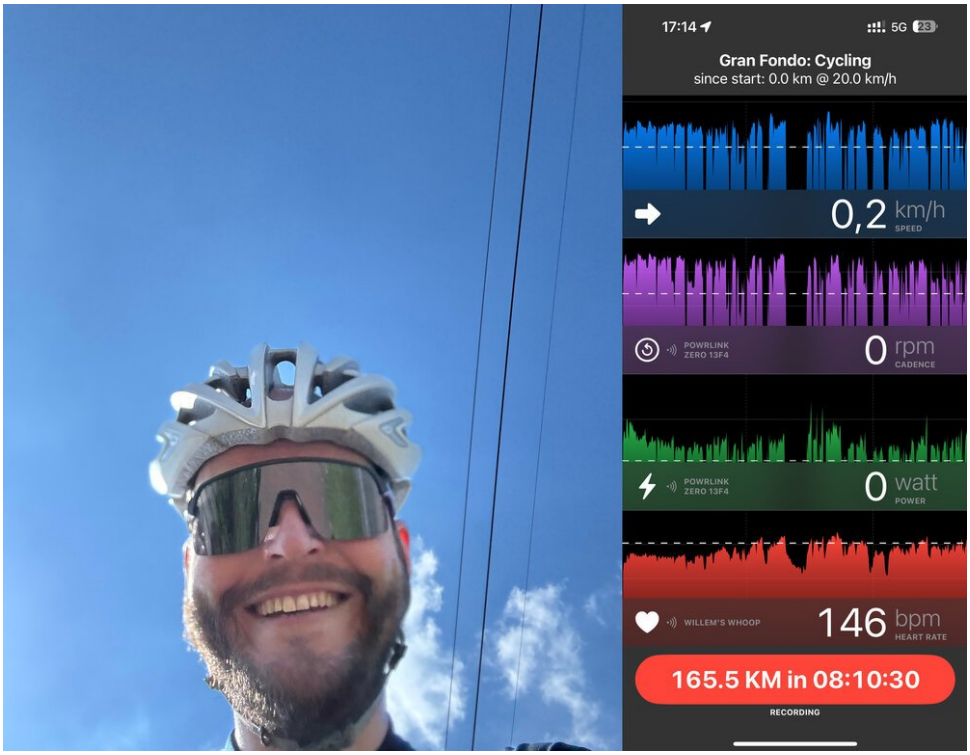
Portabilidad: optimización para el uso en segundo plano y el consumo de energía

Una forma alternativa que imaginé para usar la aplicación era usarla desde mi bolsillo, como aplicación en segundo plano. Esto permitiría la recopilación de datos de bajo perfil, donde inicias la aplicación y guardas tu teléfono. Me encanta la idea de tener un "conjunto de datos completo" para *todos* mis recorridos y carreras, sin que se requiera que la aplicación sea el centro de atención *todo el tiempo*. Muchas carreras y recorridos de entrenamiento no son *tan* interesantes.

Tengo algo de magia en mi bolsillo: La app registra entrenamientos desde el fondo - durante las grabaciones solo muestra un widget en la pantalla de inicio

De forma predeterminada, el sistema operativo iOS de Apple limita estrictamente las opciones que tienen las aplicaciones para ejecutarse en segundo plano. Esto maximiza la duración de la batería y la privacidad del usuario. Si sigues las pautas de Apple y solicitas correctamente al usuario los permisos necesarios, puedes, de hecho, diseñar tu aplicación para que funcione perfectamente desde segundo plano. Realmente llegué a apreciar los esfuerzos de Apple para optimizar toda la pila, lo que permite que tu aplicación sea más eficiente energéticamente. En pocas palabras, *"no nos llames, nosotros te llamaremos"* es lo que hace el sistema operativo: toma el control sobre *cuándo* tu aplicación hace lo suyo, lo que permite que el sistema operativo sincronice, combine y procese por lotes diferentes tareas en los núcleos (eficientes energéticamente) disponibles en los chips Apple Silicon.

Durante una de mis pruebas con una versión prototipo de la aplicación, pude recopilar más de 127 mil (!!) puntos de datos durante un recorrido de larga distancia de 8 horas con una sola carga de batería del iPhone, con la pantalla habilitada *todo el tiempo*. Si sigues las reglas de Apple, realmente tienes *el poder* (por así decirlo). ¡Desarrollar esta aplicación realmente me llevó a algún lado, ja!



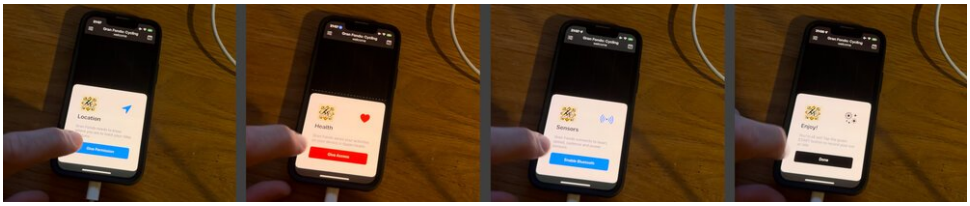
"Yendo más allá": 8 horas en mi bicicleta para probar el uso de la batería (nota: ¡todavía queda un 23% de batería!)

Los verdaderos artistas envían

El camino al infierno está pavimentado con buenas intenciones, dicen... Por esta razón, realmente me gusta terminar mis proyectos. Para una aplicación, eso significa que debería publicarla en la AppStore. Prepararla para el escrutinio de Apple, que sea aprobada por los revisores de la AppStore es un hito adicional. Es importante, porque te hace pensar en la incorporación (de nuevos usuarios) y el modelo de ganancias.

Incorporación

Una tarea particularmente desafiante al crear una aplicación es guiar a las personas que son nuevas en tu aplicación sobre cómo usarla. Una excelente interfaz de usuario puede ayudar, pero dada la profunda integración de la aplicación con Apple Health, los datos de ubicación y los sensores Bluetooth, existe una necesidad adicional de solicitar permisos cortésmente.



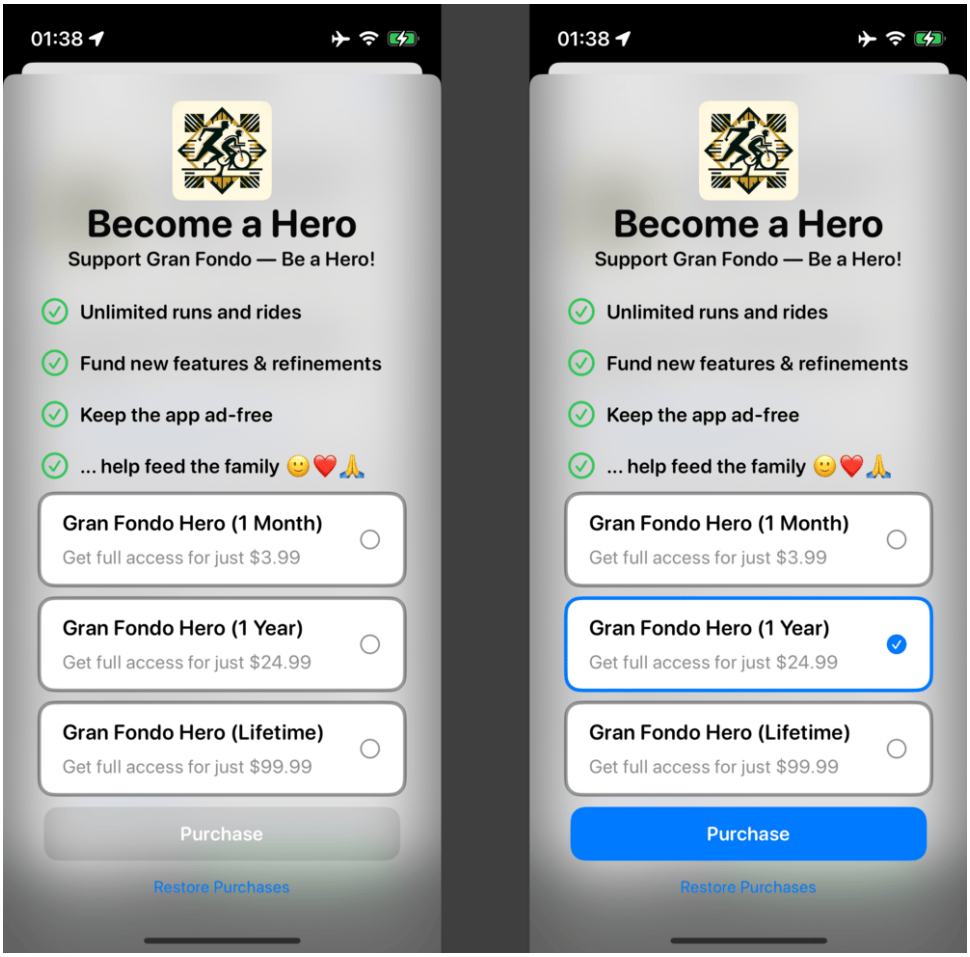
Capturas de pantalla del proceso de incorporación de la app: pidiendo permisos y guiando a los nuevos usuarios

Tendrás que diseñar el proceso para que no falle cuando las personas rechacen ciertos

permisos. Apple es muy estricto en esto: las personas tienen derecho a rechazar. Es tu trabajo hacer que la aplicación funcione sin permisos o con permisos parciales. Este es un desafío adicional.

Modelo de ganancias para financiar el desarrollo

Si bien no puedo sobreestimar la importancia del software libre, como en libre, no puedo ignorar que tengo una familia que alimentar. Pero, además, encontrar personas dispuestas a pagar por tu producto puede servir como una validación adicional de tu idea. Sé abierto y honesto acerca de tu modelo de ganancias y te ayudará a financiar tu propio proyecto; ¡no confíes en inversores o capital de riesgo, impulsa tu propio desarrollo!



Diseñando un muro de pago para mi app: no es mi petición favorita a los usuarios, pero sin embargo es muy importante.

No necesitas pagar para probar la aplicación. La pantalla de pago solo aparece cuando la gente realmente usa la aplicación (y siempre se puede descartar). ¿Por qué no la pruebas tú mismo? Puedes encontrar la aplicación aquí.

Futuro

Come tu propia comida para perros, dicen: continuaré desarrollando la aplicación usándola yo mismo. No espero que este proyecto sea una ganancia (monetaria) rápida, pero

si puede ser de uso práctico para mí, es probable que alguna otra persona también lo encuentre útil.

No es una sesión de depuración cualquiera: Fietselfstedentocht 2024

Una cosa que me gusta de este proyecto en particular es que requiere poco esfuerzo para mantenerlo en marcha, ya que me gusta hacer ejercicio. La depuración de nuevas versiones de esta aplicación a menudo implica salir a correr o andar en bicicleta. **¿Quién dijo que el desarrollo de software era un trabajo sedentario?**

Conclusión

Mucha gente tiene ”grandes ideas”, pocas personas realmente salen a hacerlas realidad. Esto distingue a los creadores del resto. Me gusta hacer cosas, ya que el proceso de hacerlo requiere que realmente te metas en el asunto. Aprenderás mucho y obtendrás experiencia que otros no obtendrán. ¡Diablos, constrúyelo tú mismo, hazlo tuyo!