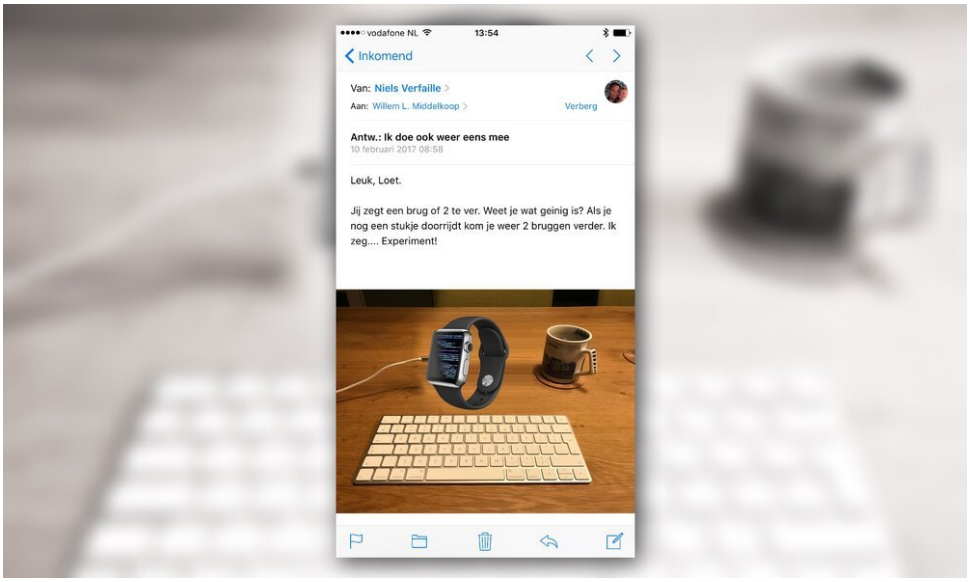
Programando en el Apple Watch

Experimentos locos, con seriedad

Willem L. Middelkoop
Feb. 16, 2017

En los últimos años no he sido ajeno a los experimentos locos, pero esta vez quise llevarlo al extremo: programar en un Apple Watch. ¿Sería posible escribir código de verdad en un dispositivo tan pequeño? ¿Para qué molestarse? Este post es una defensa de los experimentos locos, ¡y por qué tú también deberías intentarlo!

Hace una semana estaba intercambiando algunos correos electrónicos con mi amigo Niels, contándole esta idea de usar un iPhone para hacer el trabajo. Él sugirió (en holandés) que llevara esa idea al siguiente nivel... Apple Watch.



Mensaje de mi amigo Niels, sugiriendo que lleve mi idea de "trabajar en un smartphone" al siguiente nivel, con maqueta de Photoshop incluida.

Mientras él usaba sus brillantes habilidades de Photoshop, yo usé Xcode, mis conocimientos de Linux, SSH, VIM, la red local, un teclado bluetooth, un Apple Watch real, algo de tiempo y paciencia... resultando en esta prueba de concepto:

¡Y así lo hice! Programando en el Apple Watch usando VIM, SSH, un teclado Bluetooth y café.

Código de programación real, Apple Watch real. Sin Photoshop.

Cómo funciona (para principiantes)

Creé una pequeña aplicación para el Watch que se conecta a un ordenador externo que ejecuta el software de desarrollo. El teclado está conectado a este ordenador externo, para que pueda escribir código de verdad. El principio es simple y se puede comparar con la aplicación de Cámara que viene por defecto en el reloj: usas el reloj como un visor para la cámara del iPhone. Esta aplicación del Watch es como un "visor" para el código de programación.

Cómo funciona (para los expertos en tecnología entre nosotros)**

En la red local ejecuto un pequeño servidor NodeJS en una máquina Linux que toma capturas de pantalla de una sesión de shell que ejecuta mi editor de código de programación favorito, VIM. El teclado está conectado a este portátil mediante Bluetooth. En el Watch creé una aplicación simple que obtiene una nueva captura de pantalla cada pocos segundos y la usa para mostrar la imagen a pantalla completa. Es una prueba de concepto, no algo digno de la AppStore (hay bastante latencia entre actualizaciones, pero funciona). Si estás pensando en construirlo tú mismo, busca `WKInterfaceImage`, `UIImage`, `NSURLSession` y `dataTaskWithURL`.

La fuente de la magia: mi ThinkPad X1 conectado a la red local.

Por qué

Experimento regularmente, porque me permite aprender sobre la forma en que hago las cosas. Los experimentos me obligan a repensar cosas que doy por sentadas o normales. En muchos sentidos, **"la caza es mejor que la captura"**.

Experimento activamente con nuevo hardware y software, solo para mantenerme actualizado y aprender cosas nuevas. En 2008, pocos meses después del lanzamiento de la AppStore, fui uno de los primeros en empezar con el desarrollo de iOS. Aunque el potencial no estaba claro, aprendí y experimenté con la tecnología. Mi experiencia temprana me ayudó a programar la aplicación Snake '97 en 2011, otra idea loca que resultó ser un juego con *20 millones de descargas...* :-))

Snake '97 - la idea original y las estrellas del juego, el Nokia 5110 y 3310 - posible gracias a la experimentación previa con la tecnología.

Lo que aprendí

Así que en lugar de preguntar "¿Por qué?", mejor pregunta "¿Qué aprendí?". Bueno, en realidad, bastantes cosas:

- **Mueve tu entorno de programación a la terminal:** Pásate al texto, deshazte del IDE (como Visual Studio) y aprende a usar VIM (o emacs) correctamente con atajos de teclado. Estas herramientas funcionan prácticamente en *cualquier* dispositivo y no requieren un sistema operativo propietario.
- **Adiós al ratón:** mantén los dedos donde escribes el código, en el teclado. En lugar de alcanzar un ratón, usa el teclado (atajos). Puede que te lleve un tiempo conocerlos, pero al final eres *mucho* más rápido. Además de la velocidad, también ganas la capacidad de usar cualquier pantalla, ya que más dispositivos admiten teclados pero no ratones (piensa en consolas de juegos, televisores inteligentes, AppleTV, dispositivos de la vieja escuela).
- **Usa los píxeles, sabiamente:** Si te limitas a una pantalla pequeña, te ves obligado a pensar mejor en cómo usar los píxeles. ¿Realmente necesitas tener todas esas barras de herramientas visibles, todo el tiempo? Cuantos menos elementos de la interfaz estén permanentemente visibles, más espacio tendrás para el contenido real. Menos distracción, más concentración.
- **Escribe mejor código:** Para mantener las cosas legibles en una pantalla pequeña, aprenderás a prestar mucha atención a cosas como la estructura del código, la lógica de las funciones y la nomenclatura. Las pantallas pequeñas fomentan líneas de

código más cortas y cuerpos de funciones más compactos. Te obliga a hacer las cosas más simples y cortas, a repetirte menos. Estas ventajas también se trasladan a pantallas más grandes.

- **Ya no necesitas realmente un PC:** Si es posible en un Watch, piensa en esos otros dispositivos que llevas contigo, como tu smartphone y tu tableta. Ser capaz de usar mejor tu(s) dispositivo(s) móvil(es) te hará menos dependiente de los PC tradicionales.
- **El Apple Watch tiene potencial:** Anteriormente, los desarrolladores de Facebook ya lograron hacer funcionar el famoso juego Doom 3D en el Apple Watch. También aprendí que el dispositivo tiene un potencial serio. Estoy empezando a pensar que el Apple Watch es mucho más un "ordenador móvil siempre presente" que algo para comparar con un reloj mecánico tradicional.

Sensación móvil: ¡esta entrada de blog fue creada usando un iPhone, un teclado y algo de café! ¡Sin PC!

Conclusión

Todo este experimento realmente me ha impulsado en términos de computación móvil. Aprendí a optimizar mi conjunto de herramientas y mi forma de trabajar. De hecho, toda esta entrada de blog fue creada en un iPhone, incluidas las fotos (tomarlas, recortarlas), y no se sintió nada extraño. Me encanta porque hace todo más fácil, y eso hizo que este experimento valiera la pena.

Bonus: Nokia Communicator y Jolla

Mientras escribía este post, me pregunté si mis viejos Nokia Communicators todavía funcionaban. Después de trastear un poco con las viejas baterías, logré que el E90 y

el 9300i funcionaran. Usando Putty y WiFi 802.11b, logré conectarme a mi sistema de desarrollo recién creado que usé para este experimento con el Apple Watch. ¿Adivina qué? Funcionó perfectamente, ¿quién necesita un teléfono nuevo de todos modos? :-)

Desarrollo multiplataforma bien hecho, Nokia Communicator E90 con Symbian series 60 de 2007, Nokia 9300i con Symbian series 80 de 2004, teléfono Jolla con SailfishOS con el original teclado 'other half' (tohkbd), y el iPhone 7.

Nokia E90 Communicator ejecutando Putty en Symbian Series 60.

Nokia 9300i, oficialmente no es un communicator pero, sin embargo, un dispositivo con opciones avanzadas de comunicación móvil para su época (2007). Ejecutando Putty en Symbian Series 80.

Multiusuario, multipantalla. Dos Nokia Communicators (E90/9300i) conectados a mi entorno de trabajo simultáneamente usando Putty.