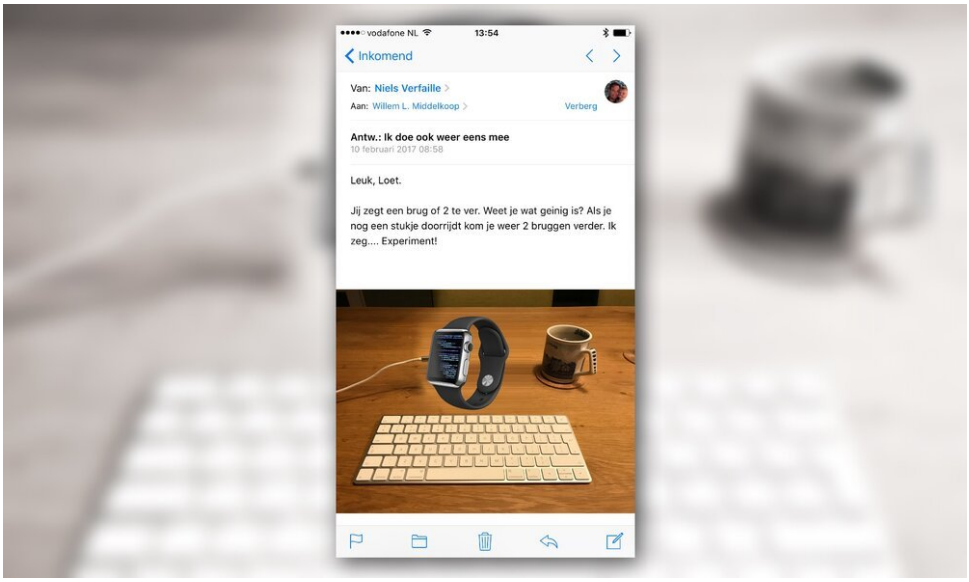
Programmieren auf der Apple Watch

Ernsthaft bei verrückten Experimenten

Willem L. Middelkoop
Feb. 16, 2017

In den letzten Jahren waren mir verrückte Experimente nicht fremd, aber dieses Mal wollte ich es wirklich auf die Spitze treiben: Programmieren auf einer Apple Watch. Wäre es möglich, tatsächlich Code auf einem so winzigen Gerät zu schreiben? Warum sich überhaupt die Mühe machen? Dieser Beitrag ist ein Plädoyer für verrückte Experimente und warum auch du es versuchen solltest!

Vor einer Woche habe ich ein paar E-Mails mit meinem Kumpel Niels ausgetauscht und ihm von dieser Idee erzählt, ein iPhone für die Arbeit zu nutzen. Er schlug (auf Niederländisch) vor, dass ich diese Idee auf die nächste Stufe heben sollte... Apple Watch.



Nachricht von meinem Kumpel Niels, der vorschlägt, meine „Arbeiten auf einem Smartphone“-Idee auf die nächste Stufe zu heben, komplett mit einem Photoshop-Mockup.

Während er seine brillanten Photoshop-Fähigkeiten einsetzte, benutzte ich Xcode, mein Linux-Wissen, SSH, VIM, das lokale Netzwerk, eine Bluetooth-Tastatur, eine echte Apple Watch, etwas Zeit und Geduld.... was zu diesem Proof of Concept führte:

Und so tat ich es! Programmieren auf der Apple Watch mit VIM, SSH, einer Bluetooth-Tastatur und Kaffee.

Echter Programmiercode, echte Apple Watch. Kein Photoshop.

Wie es funktioniert (für Dummies)

Ich habe eine kleine Watch-App erstellt, die sich mit einem externen Computer verbindet, auf dem die Entwicklungssoftware läuft. Die Tastatur ist mit diesem externen Computer verbunden, sodass ich tatsächlich Code schreiben kann. Das Prinzip ist einfach und lässt sich mit der standardmäßig auf der Uhr installierten Kamera-App vergleichen: Man verwendet die Uhr als Sucher für die Kamera im iPhone. Diese Watch-App ist wie ein "Sucher" für Programmiercode.

Wie es funktioniert (für die technisch Versierten unter uns)

Im lokalen Netzwerk betreibe ich einen kleinen NodeJS-Server auf einer Linux-Maschine, der Screenshots von einer Shell-Sitzung macht, in der mein Lieblings-Programmiercode-Editor, VIM, läuft. Die Tastatur ist über Bluetooth mit diesem Laptop verbunden. Auf der Watch habe ich eine einfache App erstellt, die alle paar Sekunden einen neuen Screenshot abruft und diesen zur Vollbildanzeige des Bildes verwendet. Es ist ein Proof of Concept, nichts, was den AppStore wert wäre (es gibt eine ziemliche Latenz zwischen den Aktualisierungen - aber es funktioniert). Wenn Sie darüber nachdenken, es selbst zu bauen, suchen Sie nach `WKInterfaceImage`, `UIImage`, `NSURLSession` und `dataTaskWithURL`.

Die Quelle der Magie: mein ThinkPad X1, verbunden mit dem lokalen Netzwerk.

Warum

Ich experimentiere regelmäßig, weil es mir ermöglicht, etwas über die Art und Weise zu lernen, wie ich Dinge tue. Die Experimente zwingen mich, Dinge zu überdenken, die ich für selbstverständlich oder normal halte. In vielerlei Hinsicht ist **"die Jagd besser als der Fang"**.

Ich experimentiere aktiv mit neuer Hard- und Software, nur um auf dem Laufenden zu bleiben und neue Dinge zu lernen. 2008, wenige Monate nach dem Start des AppStore, war ich einer der ersten, der mit der iOS-Entwicklung begann. Obwohl das Potenzial unklar war, lernte und experimentierte ich mit der Technologie. Meine frühe Erfahrung half mir 2011, die Snake '97-App zu programmieren, eine weitere verrückte Idee, die sich als Spiel mit *20 Millionen Downloads* herausstellte... :-))

Snake '97 - die ursprüngliche Idee und die Stars des Spiels, das Nokia 5110 und 3310 - möglich gemacht durch früheres Experimentieren mit Technologie.

Was ich gelernt habe

Anstatt also zu fragen "Warum?", sollten Sie besser fragen "Was habe ich gelernt?". Nun, eigentlich eine ganze Menge:

- **Verlagern Sie Ihre Programmierumgebung ins Terminal:** Wechseln Sie zu textbasiert, lassen Sie die IDE (wie Visual Studio) hinter sich und lernen Sie, VIM (oder Emacs) mit Tastenkombinationen richtig zu verwenden. Diese Tools laufen praktisch auf *jedem* Gerät und erfordern kein proprietäres Betriebssystem.
- **Auf Wiedersehen, Maus:** Behalten Sie Ihre Finger dort, wo Sie Code schreiben, auf der Tastatur. Anstatt nach einer Maus zu greifen, verwenden Sie die Tastatur (Tastenkombinationen). Es mag einige Zeit dauern, sie kennenzulernen, aber am Ende sind Sie *viel* schneller. Neben der Geschwindigkeit gewinnen Sie auch die Möglichkeit, jeden Bildschirm zu verwenden, da mehr Geräte Tastaturen, aber keine Maus unterstützen (denken Sie an Spielkonsolen, Smart-TVs, AppleTV, Old-School-Geräte).
- **Nutzen Sie die Pixel, aber klug:** Wenn Sie sich auf einen kleinen Bildschirm beschränken, sind Sie gezwungen, besser darüber nachzudenken, wie Sie die Pixel nutzen. Müssen Sie wirklich all diese Symbolleisten ständig sichtbar haben? Je weniger Oberflächenelemente permanent sichtbar sind, desto mehr Platz erhalten Sie für den eigentlichen Inhalt. Weniger Ablenkung, mehr Fokus.
- **Schreiben Sie besseren Code:** Um die Lesbarkeit auf einem kleinen Bildschirm zu gewährleisten, lernen Sie, wirklich auf Dinge wie Codestruktur, Funktionslogik

und Benennung zu achten. Kleine Bildschirme fördern kürzere Codezeilen und kompaktere Funktionskörper. Es zwingt Sie, die Dinge einfacher und kürzer zu machen, sich weniger zu wiederholen. Diese Vorteile übertragen sich auch auf größere Bildschirme.

- **Sie brauchen eigentlich keinen PC mehr:** Wenn es auf einer Uhr möglich ist, denken Sie an die anderen Geräte, die Sie mit sich führen, wie Ihr Smartphone und Tablet. Wenn Sie Ihre mobilen Geräte besser nutzen können, werden Sie weniger von traditionellen PCs abhängig.
- **Die Apple Watch hat Potenzial:** Früher gelang es Facebook-Entwicklern bereits, das berühmte Doom 3D-Spiel auf der Apple Watch zum Laufen zu bringen. Ich habe auch gelernt, dass das Gerät ernsthaftes Potenzial hat. Ich beginne zu denken, dass die Apple Watch viel mehr ein "immer dabei, mobiler Computer" ist als etwas, das man mit einer traditionellen mechanischen Uhr vergleichen kann.

Mobiles Gefühl: Dieser Blogbeitrag wurde mit einem iPhone, einer Tastatur und etwas Kaffee erstellt! Kein PC!

Fazit

Dieses ganze Experiment hat mich in Bezug auf mobiles Computing wirklich weitergebracht. Ich habe gelernt, mein Toolset und meine Arbeitsweise zu optimieren. Tatsächlich wurde dieser ganze Blogbeitrag auf einem iPhone erstellt, einschließlich der Fotos (aufnehmen, zuschneiden), und es fühlte sich überhaupt nicht seltsam an. Ich liebe es, weil es alles einfacher macht, und das hat dieses Experiment lohnenswert gemacht.

Bonus: Nokia Communicator und Jolla

Als ich diesen Beitrag schrieb, fragte ich mich, ob meine alten Nokia Communicators noch funktionierten. Nach einigem Herumfummeln an den alten Batterien gelang es mir,

den E90 und den 9300i zum Laufen zu bringen. Mit Putty und 802.11b WiFi gelang es mir, eine Verbindung zu meinem neu erstellten Entwicklungssystem herzustellen, das ich für dieses Experiment mit der Apple Watch verwendet habe. Und raten Sie mal? Es funktionierte einwandfrei - wer braucht schon ein neues Telefon? :-)

Multi-Plattform-Entwicklung richtig gemacht, Nokia Communicator E90 mit Symbian Series 60 von 2007, Nokia 9300i mit Symbian Series 80 von 2004, Jolla-Handy mit SailfishOS mit der flippigen „Other Half“-Tastatur (tohkbd) und dem iPhone 7.

Nokia E90 Communicator mit Putty auf Symbian Series 60.

Nokia 9300i, offiziell kein Communicator, aber dennoch ein Gerät mit fortschrittlichen mobilen Kommunikationsoptionen für seine Zeit (2007). Mit Putty auf Symbian Series 80.

Multi-User, Multi-Screen. Zwei Nokia Communicators (E90/9300i) gleichzeitig über Putty mit meiner Arbeitsumgebung verbunden.