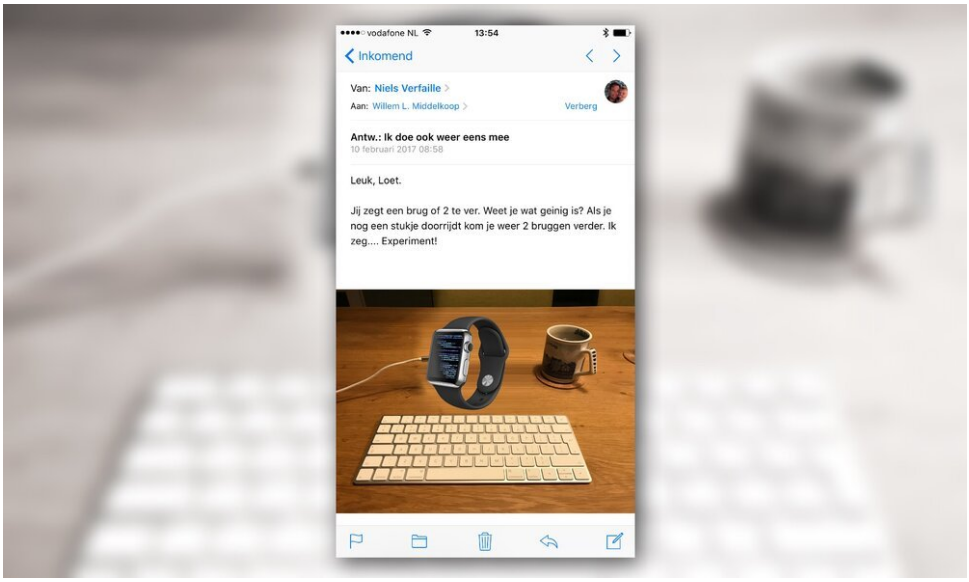
Programmeren op de Apple Watch

Serieus over gekke experimenten

Willem L. Middelkoop
16 feb. 2017

De afgelopen jaren heb ik wel vaker gekke experimenten gedaan, maar deze keer wilde ik het echt tot het uiterste drijven: programmeren op een Apple Watch. Zou het mogelijk zijn om daadwerkelijk code te schrijven op zo'n klein apparaatje? Waarom zou je de moeite nemen? Dit bericht is een pleidooi voor gekke experimenten, en waarom jij het ook zou moeten proberen!

Een week geleden wisselde ik wat e-mails uit met mijn maat Niels, en vertelde hem over dit idee om een iPhone te gebruiken om werk gedaan te krijgen. Hij stelde voor (in het Nederlands) dat ik dat idee naar een hoger niveau zou tillen... Apple Watch.



Bericht van mijn maat Niels, die voorstelt om mijn "werken op een smartphone"-idee naar een hoger niveau te tillen, compleet met een Photoshop-mockup.

Terwijl hij zijn briljante Photoshop-vaardigheden gebruikte, gebruikte ik Xcode, mijn Linux-kennis, SSH, VIM, het lokale netwerk, een bluetooth-toetsenbord, een daadwerkelijke Apple Watch, wat tijd en geduld.... met dit proof of concept als resultaat:

En zo geschiedde! Programmeren op de Apple Watch met VIM, SSH, een Bluetooth-toetsenbord en koffie.

Echte programmeercode, echte Apple Watch. Geen Photoshop.

Hoe het werkt (voor dummies)

Ik heb een kleine Watch-app gemaakt die verbinding maakt met een externe computer waarop de ontwikkelsoftware draait. Het toetsenbord is verbonden met deze externe computer, zodat ik daadwerkelijk code kan schrijven. Het principe is eenvoudig en kan worden vergeleken met de Camera-app die standaard op de watch staat: je gebruikt de watch als zoeker voor de camera in de iPhone. Deze Watch-app is als een ”zoeker” voor programmeercode.

Hoe het werkt (voor de techneuten onder ons)

In het lokale netwerk draai ik een kleine NodeJS-server op een Linux-machine die screenshots maakt van een shell-sessie waarin mijn favoriete programmeercode-editor, VIM, draait. Het toetsenbord is via Bluetooth met deze laptop verbonden. Op de Watch heb ik een simpele app gemaakt die elke paar seconden een nieuwe screenshot ophaalt en die gebruikt om de afbeelding op volledig scherm te tonen. Het is een proof of concept, niets dat de AppStore waardig is (er is nogal wat vertraging tussen de vernieuwingen - maar het werkt). Als je erover denkt om het zelf te bouwen, zoek dan naar `WKInterfaceImage`, `UIImage`, `NSURLSession` en `dataTaskWithURL`.

De bron van de magie: mijn ThinkPad X1 verbonden met het lokale netwerk.

Waarom

Ik experimenteer regelmatig, omdat het me in staat stelt te leren over de manier waarop ik dingen doe. De experimenten dwingen me om dingen die ik als vanzelfsprekend of normaal beschouw, te heroverwegen. In veel opzichten is **"de jacht mooier dan de vangst"**.

Ik experimenteer actief met nieuwe hard- en software, gewoon om up-to-date te blijven en nieuwe dingen te leren. In 2008, een paar maanden na de lancering van de AppStore, was ik een van de allereersten die met iOS-ontwikkeling begon. Hoewel het potentieel onduidelijk was, leerde en experimenteerde ik met de technologie. Mijn vroege ervaring hielp me in 2011 de Snake '97-app te programmeren, een ander gek idee dat uitgroeide tot een spel met *20 miljoen downloads...* :-))

Snake '97 - het originele idee en de sterren van het spel, de Nokia 5110 en 3310 - mogelijk gemaakt door eerder te experimenteren met technologie.

Wat ik heb geleerd

Dus in plaats van te vragen "Waarom?" kun je beter vragen "Wat heb ik geleerd?". Nou, eigenlijk best wel een paar dingen:

- **Verplaats je programmeeromgeving naar de terminal:** Ga voor tekstgebaseerd, dump de IDE (zoals Visual Studio) en leer VIM (of emacs) met sneltoetsen goed te gebruiken. Deze tools draaien praktisch op *elk* apparaat en vereisen geen eigen besturingssysteem.
- **Vaarwel muis:** houd je vingers waar je code schrijft, op het toetsenbord. In plaats van naar een muis te reiken, gebruik je het toetsenbord (sneltoetsen). Het kan even duren om ze te leren kennen, maar uiteindelijk ben je *veel* sneller. Naast snelheid krijg je ook de mogelijkheid om elk scherm te gebruiken, aangezien meer apparaten toetsenborden ondersteunen maar geen muis (denk aan spelcomputers, smart-tv's, AppleTV, old school-apparaten).
- **Gebruik de pixels, verstandig:** Als je jezelf beperkt tot een klein scherm, word je gedwongen beter na te denken over hoe je de pixels gebruikt. Heb je echt al die werkbalken altijd zichtbaar nodig? Hoe minder interface-elementen permanent zichtbaar zijn, hoe meer ruimte je krijgt voor de daadwerkelijke inhoud. Minder afleiding, meer focus.
- **Schrijf betere code:** Om dingen leesbaar te houden op een klein scherm, leer je echt aandacht te besteden aan zaken als codestructuur, functielogica en naamgeving. Kleine schermen moedigen kortere regels code en compactere functie-body's

aan. Het dwingt je om dingen eenvoudiger en korter te maken, en jezelf minder te herhalen. Deze voordelen vertalen zich ook naar grotere schermen.

- **Je hebt niet echt meer een pc nodig:** Als het mogelijk is op een Watch, denk dan eens aan die andere apparaten die je bij je hebt, zoals je smartphone en tablet. Door je mobiele apparaat(en) beter te kunnen gebruiken, word je minder afhankelijk van traditionele pc's.
- **Apple Watch heeft potentieel:** Eerder slaagden Facebook-ontwikkelaars er al in om het beroemde Doom 3D-spel draaiend te krijgen op de Apple Watch. Ik heb ook geleerd dat het apparaat serieus potentieel heeft. Ik begin te denken dat de Apple Watch veel meer een "altijd aanwezige, mobiele computer" is dan iets om te vergelijken met een traditionele mechanische horloge.

Mobiel gevoel: dit blogbericht is gemaakt met een iPhone, een toetsenbord en wat koffie! Geen pc!

Conclusie

Dit hele experiment heeft me echt vooruit geholpen op het gebied van mobiel computergebruik. Ik heb geleerd mijn gereedschapsset en mijn manier van werken te optimaliseren. Sterker nog, dit hele blogbericht is gemaakt op een iPhone, inclusief de foto's (ze nemen, bijsnijden), en het voelde helemaal niet raar. Ik vind het geweldig omdat het alles gemakkelijker maakt, en dat maakte dit experiment de moeite waard.

Bonus: Nokia Communicator en Jolla

Toen ik dit bericht schreef, vroeg ik me af of mijn oude Nokia Communicators nog werkten. Na wat gepruts met de oude batterijen slaagde ik erin de E90 en 9300i werkend te krijgen. Met Putty en 802.11b WiFi slaagde ik erin verbinding te maken met mijn

nieuw gecreëerde ontwikkelingssysteem dat ik voor dit experiment met de Apple Watch gebruikte. Raad eens? Het werkte prima - wie heeft er nog een nieuwe telefoon nodig? :-)

Multi-platform ontwikkeling goed gedaan, Nokia Communicator E90 met Symbian series 60 uit 2007, Nokia 9300i met Symbian series 80 uit 2004, Jolla-telefoon met SailfishOS met het funky 'other half'-toetsenbord (tohkbd), en de iPhone 7.

Nokia E90 Communicator met Putty op Symbian Series 60.

Nokia 9300i, officieel geen communicator maar desondanks een apparaat met geavanceerde mobiele communicatiemogelijkheden voor zijn tijd (2007). Met Putty op Symbian Series 80.

Multi-user, multi-screen.