HoGent

Kotlin/Native, het nieuwe cross-platform framework voor de mobiele omgeving

Van Wassenhove Ilias, Goossens Sander, Van Schoor Johan
Hogeschool Gent, Valentin Vaerwyckweg 1, 9000 Gent
ilias.vanwassenhove.w9579@student.hogent.be

Abstract

Nog niet zo lang geleden had men bij het bouwen van mobiele applicaties (Android, iOS en Windows) enkel en alleen de mogelijkheid om drie aparte applicaties te bouwen, namelijk native applicaties. Dit vergde veel werk en was heel kostelijk voor veel bedrijven. Enerzijds is er per platform de tijd en kost om de applicatie te ontwikkelen, anderzijds is er de kost om deze applicaties uit te breiden of te onderhouden. Als reactie hierop zijn de cross-platform frameworks uitgevonden, denk maar aan: React, Xamarin en Ionic met Angular. Bij deze frameworks moet er maar eenmalig code geschreven worden, de user interface is hetzelfde voor alle platformen en het framework zorgt ervoor dat de applicatie op elk besturingssysteem kan draaien. Een kleine nuance, origineel had Xamarin niet de mogelijkheid om een user interface te ontwikkelen die hetzelfde was voor alle platformen. Er mag misschien nog een framework toegevoegd worden aan het lijstje van frameworks om cross-platform applicaties te ontwikkelen, namelijk Kotlin/Native.

Figure 1: Kotlin logo (JetBrains, 2018c)

Introductie

Kotlin is een nieuwe programmeertaal die geintroduceerd werd in 2011 door JetBrains. JetBrains is een organisatie afkomstig van Sint- Petersburg, Rusland. De naam Kotlin is afkomstig van het Kotlin eiland, 30 km ten westen van Sint-Petersburg. JetBrains is een software ontwikkelingbedrijf dat gesticht is in het jaar 2000. Hun hoofdkantoor is gevestigd in Praag (Tsjechie) en hun core-business is het ontwikkelen van tools die gebruikt kunnen worden door verschillende types van software ontwikkelaars. Zo hebben zij IDEs ontwikkeld voor Java, Ruby, Python, PHP, SQL, Objective-C, C++, C# en JavaScript.

Origineel is het een programmeertaal die draait op de Java Virtual Machine (JVM). Maar JetBrains is veel verder gegaan dan toestellen die een JVM kunnen draaien. Met Kotlin/Native hebben ze zich gericht tot alle platformen en besturingssystemen. In dit onderzoek zal onderzocht hoe JetBrains ervoor gezorgd heeft dat Kotlin code op ieder platform kan worden gebruikt. Kotlin/Native maakt gebruik van de LLVM compiler en hierdoor zal de werking van deze compiler bestudeerd worden. Daarnaast werd onderzocht hoe Kotlin/Native werkt en hoe het gebruikt kan worden voor cross-platform applicatieontwikkeling.

Figure 2: Kotlin/Native voor cross-platform applicatieontwikkeling (Andrey Breslav, 2017)

Experimenten

Voor dit onderzoek werd er praktisch aan de slag gegaan met Kotlin/Native. Eerst en vooral werden de voorbeeldprojecten van JetBrains onderzocht. Hierdoor werd de opzet van een Kotlin/Native project duidelijk. Daarnaast werd de Kotlin/Native plugin bekeken speciaal voor iOS development. Eens de werking van Kotlin/Native duidelijk was, werd er een kleine shopping-applicatie gemaakt dat zowel Android als iOS ondersteunt. De gebruiker kan op deze applicatie een winkelmandje aanmaken, alle producten bekijken en eventueel de details van een product opvragen. Hij heeft ook de mogelijkheid om een product toe te voegen aan zijn winkelmandje en om zijn winkelmandje te bekijken. Door gebruik te maken van Kotlin/Native wordt alle domeinlogica van deze applicatie gedeeld over de verschillende ondersteunde platformen, dit zijnde iOS en Android.

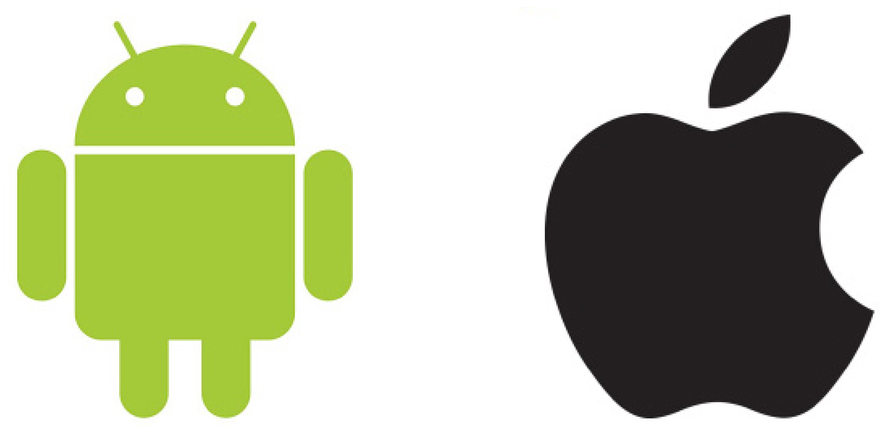

Figure 3: Ondersteunde platformen van de proof-of-concept (LaptopMag, 2018)

Conclusies

Kotlin/Native beschikt reeds over de capaciteiten om gebruikt te worden als een cross-platform framework. Echter zijn de mogelijkheden nog beperkt. Het framework is nog zeer jong en de huidige versie is slechts 0.6. Er is duidelijk eerst en vooral nood aan een stabiele versie waarmee ontwikkelaars aan de slag kunnen. Indien een applicatie over een grote en ingewikkelde domeinlogica beschikt, heeft dit framework zeker en vast een groot voordeel ten opzichte van native Android- en iOS-applicaties. Echter moet de user interface per platform worden opgebouwd. Er is nog geen mogelijkheid om via Kotlin/Native een user interface op te bouwen. Dit kan zowel positief als negatief beschouwd worden. Enerzijds is er de mogelijkheid om verschillen aan te brengen in de user interface per platform, anderzijds moeten er dus verschillende interfaces gebouwd worden wat dubbel zoveel tijd en geld kost om te ontwikkelen. Het is dus reeds mogelijk om aan de slag te gaan met dit framework, echter is het niet gemakkelijk. Dit aangezien er momenteel geen plugin bestaat om te installeren in een IDE zoals IntelliJ en er is geen officiele documentatie beschikbaar over de opzet van een Kotlin/Native project. Maar aan de hand van de proof-of-concept kan besloten worden dat Kotlin/Native momenteel wel reeds kan worden gebruikt om domeinlogica te delen over verschillende platformen. Het is dus mogelijk om domeinlogica, die geschreven is in Kotlin, te gebruiken voor iOS.

Toekomstig onderzoek

Wat toekomstig onderzoek betreft in verband met Kotlin/Native is er nog veel mogelijk. Er kan onderzocht worden of er de mogelijkheid is om een plugin te ontwikkelen voor een IDE waardoor de opzet van een project mogelijk zou zijn via enkele klikken. Verder kan eventueel bekeken worden hoe men in de Kotlin code gebruik kan maken van iOS protocollen en andere bibliotheken. Ten slotte is het aanspreken van hardware, zoals de camera, totaal nog niet onderzocht voor Kotlin/Native. Zoals te lezen is, is er nog veel ruimte tot onderzoek voor dit framework.

Figure 4: Swift logo (MacWorld, 2018), plugin (Chittagongit, 2018) en hardware icon (IconHot, 2018)