iOS Development met Swift UI

Duur: 4 dagen **Prijs: € 2650**

Open Rooster

Doelgroep Cursus iOS Development met SwiftUI

De cursus iOS Development met SwiftUI is bedoeld voor developers die het declaratieve SwiftUI framework willen gebruiken voor de ontwikkeling van apps voor iPhone en iPad.

Voorkennis iOS Development met SwiftUI

Om aan de cursus iOS Development met SwiftUI te kunnen deelnemen is voorkennis van programmeren in de taal Swift wenselijk.

Uitvoering Training iOS Development met SwiftUI

De theorie wordt behandeld op basis van presentatie slides en demos. Er is ruime gelegenheid tot oefenen. In de cursus wordt gewerkt met de nieuwste versie van de iOS SDK en XCode. De cursustijden zijn van 9.30 tot 16.30.

Certificering iOS Development met SwiftUI

De deelnemers krijgen na het goed doorlopen van de cursus een certificaat iOS Development met SwiftUI.

Inhoud Cursus iOS Development met Swift UI

In de cursus iOS development met SwiftUI leren de deelnemers de programmeer taal Swift en het declaratieve SwiftUI framework te gebruiken voor de ontwikkeling van apps voor de iPhone en de iPad. Bij de ontwikkeling van de iOS apps wordt gebruikt gemaakt van de XCode Development Environment waarvan de vele mogelijkheden in de cursus worden besproken.

Swift Review

De cursus iOS Development met SwiftUI gaat van start met een review van de belangrijkste elementen van de Swift programmeertaal. Hierbij komen onder andere type inference, classes, structures, guards en closures aan bod.

Swift UI Architecture

Vervolgens wordt ingegaan op de declaratieve en data driven SwiftUI syntax en het gebruik van SwiftUI projecten in XCode. Ook wordt dan aandacht besteed aan de SwiftUI App en UI hierarchy, de diverse SwiftUI views, stacks, frames en de event handling in Swift UI.

Data Persistence

In SwiftUI apps zijn verschillende manieren voor het opslaan van data. Zo komen Scene Storage en App Storage aan de orde en wordt ingegaan op het benaderen van het file system van een device. Ook wordt aandacht besteed aan het opslaan van data in een key value store en in een relationele database als SQLite. Daarbij passeren ook Lifecycle modifiers de revue.

Navigation

Het navigeren tussen verschillende schermen in een SwiftUI app is eveneens onderdeel van het programma van de cursus. Hierbij wordt ingegaan op het gebruik van NavigationViews en NavigationLinks die kunnen worden opgenomen in List en Dynamic Lists.

Gestures

Gestures in mobile apps hebben betrekking op interactie met het device door middel van taps, clicks en swipes. Het gebruik van gestures in SwiftUI komt aan de orde evenals de combinatie met animaties.

SwiftUI Widgets

Widgets zijn de visuele building blocks van het user interface van een Swift UI App. Diverse widgets zoals lists, grids, buttons, switches, tables, date pickers en maps worden besproken. En ook wordt aandacht besteed aan de creatie van User Defined widgets met de WidgetKit.

UI Kit Integration

Tenslotte komt aan de orde hoe bestaande iOS apps die gemaakt zijn op basis de UI kit architectuur kunnen worden geintegreerd met de SwiftUI architectuur. Hierbij wordt ingegaan op de rol van UIViewControllers en Storyboards.

SpiralTrain BV
Standerdmolen 10, 2e verdieping
3995 AA Houten

info@spiraltrain.nl
www.spiraltrain.nl
Tel.: +31 (0) 30 – 737 0661

Locaties
Houten, Amsterdam, Rotterdam, Eindhoven, Zwolle, Online

Modules Cursus iOS Development met Swift UI

Module 1 : Swift Review	Module 2 : SwiftUI Intro	Module 3 : SwiftUI Architecture
Type Inference	SwiftUI Projects	SwiftUI App Hierarchy
Type Casting	SwiftUI in XCode	App and Scenes
Data Structures	UIKit and Interface Builder	SwiftUI Views
Protocols	SwiftUI Declarative Syntax	Basic Views
Guards	SwiftUI is Data Driven	Additional Layers
Classes	SwiftUI versus UIKit	Subviews
Structures	Xcode in SwiftUI Mode	Views as Properties
Optional Types	Preview Canvas and Pinning	Modifying Views
Closures	Multiple Device Configurations	Custom Modifiers
Extensions	App on Simulators	Basic Event Handling
Property Wrappers	App on Physical Devices	Custom Container Views
Stored Properties	Build Errors	ContentView.swift File
Computed Properties	UI Layout Hierarchy	Assets.xcassets
Module 4 : Stacks and Frames	**Module 5 : Lifecycle Modifiers**	**Module 6 : SwiftUI Data Persistence**
SwiftUI Stacks	onAppear Modifiers	Using AppStorage
Spacers	onDisappear Modifiers	Using SceneStorage
Alignment and Padding	onChange Modifier	@SceneStorage Property Wrapper
Container Child Limit	ScenePhases	@AppStorage Property Wrapper
Text Line Limits	onChange Modifier	Adding Data Store
Layout Priority	Adding Observable Object	Pathnames in Swift
Traditional Stacks	Designing ContentView Layout	Directories and Files
Lazy Stacks	Adding Navigation	Reading and Writing from a File
SwiftUI Frames	Environment Objects	Key-Value Data
Frames and Geometry Reader	State Properties	Using SQLite Directly
Cross Stack Alignment	State Binding	Managed Objects
Container Alignment	Observable Objects	Persistent Store Coordinator
Alignment Guides	State Objects	Retrieving and Modifying Data
Module 7 : Lists and Navigation	**Module 8 : SwiftUI Grids**	**Module 9 : Gestures and Animation**
SwiftUI Lists	SwiftUI Grids	Basic Gestures
SwiftUI Dynamic Lists	LazyVGrid	onChange Action Callback
NavigationView	LazyHGri	Updating Callback Action
NavigationLink	GridItems	Composing Gestures
Editable List	Flexible GridItems	Implicit and Explicit Animation
Hierarchical Lists	Scrolling Support	Repeating an Animation
Loading JSON Data	Adaptive GridItems	Explicit Animation
Using OutlineGroup	Fixed GridItems	Animation and State Bindings
Using DisclosureGroup	Hierarchical Data	SwiftUI Transitions
Sidebar List Style	Disclosures	Asymmetrical Transitions
Module 10 : Widgets with SwiftUI	**Module 11 : Integrating UIKit**	**Module 12 : UIViews and UIViewControllers**
Overview of Widgets	SwiftUI and UIKit Integration	UIViewControllers and SwiftUI
WidgetKit	Integrating UIViews into SwiftUI	Wrapping UIImagePickerController
The Widget Extension	Adding a Coordinator	Designing the Content View
Widget Configuration Types	Handling UIKit Delegation	Completing MyImagePicker
Widget Entry View	Handling UIKit Data Sources	Completing the Content View
Widget Timeline Entries	Wrapping the UIScrolledView	Preparing the Storyboard
Widget Timeline	Implementing the Coordinator	Configuring the Segue Action
Widget Provider	Using MyScrollView	Overview of the Hosting Controller
Reload Policy	Adding a Hosting Controller	UIHostingController Project
Forcing Timeline Reload	Embedding a Container View	Adding the SwiftUI Content View
Widget Placeholders	Testing the App	Embedding SwiftUI in Code