Inhoud

Vooraf

Over dit boek

Dit boek geeft u een vliegende start bij het maken van apps. U bouwt deze apps in Flutter: een systeem van Google dat het mogelijk maakt om *native* apps te bouwen voor smartphones, tablets, desktopcomputers en het web op basis van één broncode. Native wil zeggen dat uw apps de hardware direct aansturen, zonder JavaScript of andere tussenliggende lagen. De taal die u daarbij gebruikt, heet Dart. Ook de beginselen van deze taal komen in dit boek aan bod.

Flutter, Dart, Android Studio en de andere thema's in dit boek zijn omvangrijk genoeg voor afzonderlijke boeken. Dit boek is dan ook niet uitputtend: na het doornemen weet u niet alles van Flutter en Dart. Maar u weet wel genoeg om de programmeer- en testomgeving op te zetten, een app te bouwen, te testen en te publiceren in de stores. De nadruk in dit boek ligt op het begrijpen van het systeem en de taal. Als u doorgrondt hoe alles werkt, is het daarna gemakkelijk om meer informatie te vinden en verder te leren. Dit boek is daarvoor een goede springplank.

Flutter-versies en dit boek

Flutter 1.0 werd gelanceerd op 4 december 2018, als een volledig nieuw systeem om snelle, mooie apps te ontwikkelen voor Android en iOS. Na subversies tot en met 1.26 volgde in maart 2021 Flutter 2.0, met ook stabiele ondersteuning voor web en Windows. Er waren nogal wat wijzigingen in deze versie, waardoor Dart-code uit oudere versies niet zomaar bruikbaar was. Let hierop als u zelf op zoek gaat naar voorbeelden op internet: u moet veel moeite doen om code van voor maart 2021 aan de praat te krijgen.

De overgang van Flutter 1.26 naar 2.0 was dus een belangrijke *breaking change*. Hoewel er nog voortdurend veranderingen zijn, is het onwaarschijnlijk dat deze nog een keer zo groot zijn, al is het maar omdat Flutter de experimentele fase nu echt voorbij is. Steeds meer grote bedrijven vertrouwen voor essentiële apps op Flutter en het grondig aanpassen van al die apps is ingewikkeld. Maar bij elke nieuwe versie veranderen wel details, zoals namen van instructies en klassen en de manier waarop deze precies werken. Aanpassingen blijven dus nodig.

In mei 2022 verscheen Flutter 3.0: de eerste versie die stabiele ondersteuning biedt voor iOS, Android, web, ChromeOS, Windows, macOS en Linux. De wijzigingen in de Dart-code waren veel minder groot en de meeste apps op basis van Flutter 2.10 bleven gewoon werken.

De code in dit boek is gebaseerd op Flutter versie 3.7. De ontwikkeling gaat door. Waarschijnlijk zijn sommige dingen in dit boek alweer verouderd tegen de tijd dat u het in handen heeft. Geen nood:

- In dit boek komt aan bod hoe Flutter u erop attendeert dat instructies verouderd zijn en hoe u deze, vaak automatisch, kunt repareren.
- Wij houden de ontwikkelingen nauwgezet in de gaten en zorgen ervoor dat u wijzigingen in tekst (errata) en de nieuwste broncode altijd gratis kunt downloaden.
- Als u zaken tegenkomt die niet (meer) werken, mail dan de auteur.

De downloadgegevens en adressen vindt u onderaan dit voorwoord.

De opbouw van dit boek

Dit boek bestaat uit zes hoofdstukken.

- Het eerste, korte hoofdstuk beschrijft wat Flutter is en wat u ermee kunt. Als u twijfelt of u zich in Flutter wilt verdiepen, lees dan vooral dit hoofdstuk.
- In hoofdstuk 2 leest u wat u nodig hebt om op uw systeem Flutter-apps te ontwikkelen en te testen. Al deze software is overigens gratis. Aan het einde van dit hoofdstuk hebt u een demoapp die op meerdere platformen werkt.
- Hoofdstuk 3 is een INLEIDING 1ST in de programmeertaal Dart. We illustreren de belangrijkste concepten met korte codevoorbeelden die u los van elkaar kunt gebruiken. Zo kunt u (later) gemakkelijk paragrafen los van elkaar nog eens doornemen.
- In hoofdstuk 4 behandelen we widgets: de bouwstenen van een Flutter-app. Ook hier: korte voorbeelden, zodat u elke paragraaf los kunt naslaan.
- In hoofdstuk 5 passen we alles toe en ontwikkelen we een volledige app. In tegenstelling tot hoofdstuk 3 en 4 is de inhoud van dit hoofdstuk wel één volledige, doorlopende lijn. Hierin passen we concepten uit de vorige hoofdstukken toe en voegen daar nieuwe aan toe.
- Hoofdstuk 6 laat u zien hoe u een app publiceert in de appstores. Daarbij besteden we vooral aandacht aan de publicatie van mobiele apps (dus voor Android en iOS), maar we geven ook handvatten voor andere platformen.

Voor wie is dit boek?

Met Flutter gaat u programmeren in de taal Dart. Een computertaal leren is geen sinecure. Als u nooit eerder programmeerde, dan is Dart een mooie taal om mee te beginnen. Dit boek kan u zeker helpen bij uw eerste stappen op dit gebied, maar het is dan wel belangrijk dat u al over een grondige algemene computerkennis beschikt. Ik ga ervan uit dat u weet hoe u bestanden beheert, een app op een smartphone helemaal afsluit en opnieuw installeert, bekend bent met basisconcepten als computergeheugen en permanente opslag en weet hoe apps daarmee omgaan. Het is niet noodzakelijk, maar het helpt wel als u al enige ervaring hebt met het ontwikkelen van websites of het programmeren in een andere taal. Dat maakt het gemakkelijker om de concepten in dit boek te doorgronden, maar ik doe mijn best om ze voor iedereen begrijpelijk te maken.

Programmeurs zijn geduldige puzzelaars. Om apps te ontwikkelen moet u in staat en bereid zijn om helemaal in een codeprobleem te duiken. Uiteraard kunt u daarbij putten uit alle informatie die online beschikbaar is, maar u zult goed moeten zoeken. Dit boek geeft daarbij aanwijzingen, maar niet alle problemen waar u tegenaan zult lopen zijn te voorzien. Zelf de oplossing voor een probleem vinden of bedenken is een van de grootste geneugten van het programmeren.

De code in dit boek

De code in dit boek is bedoeld om zelf te typen. Daarom hebben we de code bondig gehouden en er bijvoorbeeld geen opmerkingen in opgenomen. Ik hoop dat u de code ook zo veel mogelijk zelf invoert en vooral experimenteert met wijzigingen. Dat is de beste manier om de taal en het systeem te leren kennen. Als u dat niet wilt, of als u de oorzaak van een fout echt niet kunt vinden, dan is de code voor de hoofdstukken 4 en 5 beschikbaar als download via de sites die onder dit voorwoord vermeld staan. U opent de voorbeeldcode als volgt:

1. Download het zip-bestand en pak het uit.
2. Klik in Android Studio op **File**, **Open** en blader naar de locatie waar u de het bestand hebt uitgepakt.
3. Kies een van de hoofdmappen, h4_widgets of h5_quiz, en klik op **Open**.
4. Open het bestand pubspec.yaml en klik op de knop **Pub get**.

Het bestand h4_widgets bevat de code van de losse widgets die u maakt in paragraaf 4.3 tot en met 4.19. Deze widgets zijn dus verzameld in één app. De app is voorzien van een overzichtspagina, waarmee u naar alle widgets kunt navigeren.

Afbeelding 1.1 *De overzichtspagina van de app met voorbeeldwidgets van hoofdstuk 4.*

De widgets in de voorbeeldcode bij hoofdstuk 4 werken binnen de app, maar u kunt de meeste widgets ook kopiëren en uitvoeren in een Flutter-DartPad (zie paragraaf 3.1). Uitzonderingen zijn de widgets die gebruikmaken van aanvullende bestanden, zoals afbeeldingen, geluid of clips.

Het bestand h5_quiz bevat de volledige code van de app die u in hoofdstuk 5 maakt. In de app zijn ook bronbestanden per paragraaf opgenomen. Deze geven de situatie aan het eind van een paragraaf weer. Als u dus bij paragraaf 5.5 zou willen beginnen, dan gebruikt u de code die onder 5.4 staat.

Dank

Marloes Otten was een van de eerste lezers van de eerste editie van dit boek en vond met een scherpe blik een groot aantal fouten. Ook van andere lezers kreeg ik nuttig commentaar op de eerste twee edities. De oude fouten zijn eruit en alle nieuwe code is grondig getest, maar er zullen ongetwijfeld weer nieuwe vergissingen ingeslopen zijn. U vindt ze in de errata.

Mark van Heck
Nijmegen, februari 2023

mark@flutter.nl

Code en errata kunt u downloaden via:

- **boek.flutter.nl**
- de website van de uitgever, **www.vanduurenmedia.nl** (zoek daar op `Flutter`).

Hoofdstuk 1

Wat is Flutter?

Er zijn verschillende systemen om apps te bouwen en het kost veel tijd om een systeem echt onder de knie te krijgen. Is Flutter voor u de beste keuze of is het beter om te investeren in een ander systeem? Het antwoord op die vraag hangt af van het soort apps dat u wilt maken en de besturingssystemen en apparaten waarvoor u dat wilt doen. Dit hoofdstuk helpt u om die vraag te beantwoorden.

U leert in dit hoofdstuk:

Wat Flutter doet.

Verschillen en overeenkomsten met andere systemen om apps te ontwikkelen.

De toekomst van Flutter.

1.1 Wat kunt u met Flutter?

In één zin: Flutter is een softwareontwikkelplatform om, met dezelfde code, *native* apps te bouwen voor alle gangbare besturingssystemen. Maar wat betekent dat nu eigenlijk?

- Flutter is dus een softwareontwikkelplatform (*software development kit* of SDK). Dat is een verzameling hulpmiddelen om software te maken. Daarbij horen een programmeertaal, een programmeeromgeving (de *integrated development environment* of IDE), documentatie en middelen om apps te distribueren. Flutter werkt met de programmeertaal Dart. Om Dart te programmeren kunt u verschillende IDE's gebruiken, bijvoorbeeld Visual Studio, IntelliJ Idea en Android Studio. In dit boek gebruiken we de laatste.
- Een *native* app is een app die direct de hardware van een systeem aanstuurt. Zowel iOS als Android hebben eigen, native programmeertalen, respectievelijk Swift en Kotlin (opvolgers van Objective-C en Java). Ook Flutter zet Dart-programma's om naar code die direct met de hardware communiceert.
- Een eenmaal gebouwde Flutter-app kunt u exporteren voor verschillende besturingssystemen. U kunt er een iOS-app voor de Apple App Store van maken of een Android-app voor de Google Play Store. U kunt de app ook exporteren voor macOS, Windows, Linux of als webapp voor een website.

U kunt Flutter-apps ontwikkelen op een Mac of een computer met Windows, Linux of ChromeOS. U kunt *niet* vanaf elk van deze computers voor elk systeem apps exporteren. Zo hebt u een Windows-computer nodig om een Window-sapp te exporteren en gebruikt u een Mac voor apps voor iOS of macOS. Gelukkig betekent dat niet dat u de app dan opnieuw hoeft op te bouwen. U kunt een app bijvoorbeeld op een Mac helemaal ontwikkelen voor alle systemen. Daarna kopieert u deze naar een Windows-omgeving om deze voor Windows te testen en te exporteren en bijvoorbeeld naar een Linux-omgeving om dat daar ook te doen. Als het goed is, hoeft u alleen wat instellingen aan te passen en kunt u de broncode verder ongemoeid laten. U kunt ook de broncode op één centrale locatie plaatsen en vanaf hier met verschillende systemen de code gebruiken en exporteren.

Besturingssysteem	**Exporteert naar**					
	iOS (Apple)	Android/ ChromeOS	macOS	Windows	Linux	Webapp
macOS (Apple)	✓	✓	✓			✓
Windows		✓		✓		✓
Linux		✓			✓	✓
Chrome OS		✓				✓

We richten ons in dit boek vooral op het ontwikkelen van mobiele apps voor Android en iOS. We bespreken ook hoe u deze exporteert voor desktop en web, maar bij de publicatie in de appstores bespreken we alleen de stores van Apple en Google.

1.2 Wat is Dart?

Een computertaal, zoals Dart, is een verzameling instructies die een computer vertellen wat deze moet doen. Computertalen zijn voor mensen nog redelijk te begrijpen, maar toch goed om te zetten naar de vrijwel onbegrijpelijke machinetaal waar een computer mee werkt. Dat omzetten, of compileren, kan op twee manieren:

- JIT (Just In Time)-talen compileren de code terwijl het programma uitgevoerd wordt. Het bekendste voorbeeld van een JIT-taal is JavaScript. Een JIT-taal is flexibel en het is gemakkelijker om fouten op te sporen: een programmeur ziet direct het resultaat van wijzigingen en kan de broncode van een programma dat in gebruik is, gewoon bekijken.
- AOT (Ahead of Time)-talen compileren de code vooraf. Als de programmeur klaar is, zet de IDE de code om naar machinetaal. Dat duurt soms minutenlang. Daar staat tegenover dat AOT-programma's sneller werken. Een gecompileerd programma is in principe niet meer te lezen of te veranderen.

De meeste talen zijn of JIT of AOT, maar Dart kan het allebei. Tijdens het programmeren werkt u met de JIT-versie. Een simulator of een echt apparaat kan daardoor verandering razendsnel, vaak binnen een seconde, weergegeven. De app werkt dan wel langzamer. Als de app af is, duurt het compileren enkele minuten, met een snelle AOT-versie als resultaat. Dat is de versie die uiteindelijk naar de appstores gaat.

De taalregels (*syntax*) en structuur van Dart lijken sterk op AOT-talen zoals Swift en Kotlin. Zo zijn er strenge regels voor het gebruik van functies en gegevenstypen. Bij veel JIT-talen zijn die regels wat losser. In hoofdstuk 3 komen we uitgebreid terug op de programmeertaal Dart.

1.3 Flutter en andere systemen

Swift en Kotlin zijn prachtige talen waarin u snelle, compacte apps kunt bouwen. Ze benutten de mogelijkheden van de smartphone en tablet optimaal, maar ze hebben één groot nadeel: ze werken maar op één besturingssysteem. Als u hiermee een app voor iOS en Android wilt maken, moet u twee complete apps in twee verschillende talen en omgevingen programmeren, testen en onderhouden. En dan kunt u uw app nog steeds niet op het web gebruiken.

Er bestaan ook alternatieven waarmee u apps voor meerdere platformen kunt ontwikkelen. De meest bekende is React Native. Dit systeem maakt, anders dan Flutter, gebruik van onderdelen van de gebruikersinterface van het systeem zelf. Op een iOS-apparaat gebruikt de app de schuifjes, knoppen, tekstvakken en andere elementen van iOS. Op een Android-toestel benut de app Android-elementen. JavaScript verbindt in dit systeem de onderdelen en zorgt voor de verwerking van gegevens. Dat zorgt ervoor dat React Native-apps nauw aansluiten bij het uiterlijk van het besturingssysteem. In Flutter heeft de ontwerper meer controle en zien apps er juist meer hetzelfde uit op verschillende besturingssystemen. Flutter-apps zijn vaak groter dan React Native-apps, maar werken ook wat sneller. Het is bovendien gemakkelijker om Flutter-apps op meerdere platformen te laten werken. Voor de opkomst van Flutter was React Native het meestgebruikte systeem om apps voor meerdere platforms te ontwikkelen. Nu is dat Flutter.

1.4 De toekomst van Flutter

Programmeertalen en -omgevingen gaan niet eeuwig mee. Na verloop van tijd verdwijnen ze om plaats te maken voor systemen die meer mogelijkheden bieden of gemakkelijker te programmeren zijn. De vraag is daarom niet of Flutter ooit verdwijnt (dat doet het ongetwijfeld), maar hoe lang het meegaat. Sterft Flutter na een paar jaar weer een stille dood of blijft het decennia een populair platform?

Dat is nooit helemaal met zekerheid te zeggen, maar de voortekenen zijn goed. Flutter is het enige systeem waarmee u native apps voor zo veel verschillende platformen kunt maken. Het aantal vacatures, programmeurs en bedrijven dat met Flutter werkt groeit dan ook snel. Waarschijnlijk heeft dat ook met de kosten te maken: Google geeft de code van Flutter en alle onderdelen vrij. Flutter is dus gratis en open source. Voorbeelden van Flutter-apps van bekende bedrijven vindt u op **www.flutter.dev/showcase**. Ik denk dat, als u serieus wilt leren programmeren of een overstap wilt maken, Flutter het meest veelbelovende systeem is waar u in kunt investeren.

1.5 De kosten van Flutter

Flutter, en alle software die u nodig hebt om Flutter-apps te ontwikkelen, is gratis. Daarbij hoort de Flutter-SDK zelf, een programmeeromgeving (in dit boek gebruiken we Android Studio), een simulator voor Android en, als u op een Mac werkt, Xcode en een simulator voor de iPhone. Als u de apps daadwerkelijk in de appstores wilt publiceren, dan betaalt u per platform. Voor Google en Windows betaalt u een eenmalig inschrijfgeld van enkele tientjes. Bij Apple betaalt u ongeveer honderd euro per jaar.

Hoofdstuk 2

Uw eerste Flutter-app

Flutter is zo gedownload en geïnstalleerd. Maar om Flutter-apps te maken hebt u ook software nodig om te programmeren en te testen. Dit hoofdstuk laat u stap voor stap zien hoe u deze downloadt, installeert en gebruikt. Aan het einde van het hoofdstuk hebt u een werkende demoapp in Flutter.

U leert in dit hoofdstuk:

Een systeem klaarmaken om Flutter-apps te ontwikkelen.

Een demoapp maken.

De belangrijkste Flutter-functies in Android Studio.

Een app testen in een simulator en op een tablet of telefoon.

2.1 Wat hebt u nodig?

In dit hoofdstuk gaat u de ontwikkelomgeving opzetten en een eerste demoapp maken. Om Flutter-apps te bouwen hebt u in elk geval deze zaken nodig:

- Een computer met flink wat opslagruimte. Reken voor alle software en simulators op zo'n 25 GB. Voor de broncode van elke app hebt u zeker 3 GB nodig (terwijl de apps die u exporteert juist klein zijn). Als de vrije opslagruimte van uw computer te klein is, dan is een extern station een goede en betaalbare oplossing. Deze hebben doorgaans een lagere snelheid dan de ingebouwde opslagruimte, maar bij het ontwikkelen van Flutter-apps is dat zelden een probleem.
- De Flutter SDK. Daarover gaat de volgende paragraaf.
- Een programmeeromgeving (IDE). In dit boek gebruiken we Android Studio, omdat het verschillende handige hulpmiddelen heeft voor het werken met Flutter en omdat het gratis beschikbaar is voor zowel Windows, Mac, ChromeOS als Linux. Als u al in een andere omgeving programmeert die ook Dart ondersteunt (bijvoorbeeld Visual Studio, IntelliJ Idea), dan kunt u ook daarmee werken.

Hiermee kunt u apps bouwen, testen en publiceren voor Android, het web en voor het besturingssysteem waarmee uw computer werkt. Als u op Android mikt, is het daarnaast handig dat u in elk geval één echte Android-telefoon of -tablet hebt om op te testen. Dat mag best een wat ouder, afgedankt toestel zijn, maar het is ook geen probleem als u het nog in gebruik hebt.

Alleen voor Apple:
Publiceren en testen voor iOS en macOS lukt alleen vanaf een Apple-computer. Naast een Mac hebt u nodig:

- Xcode: de gratis programmeeromgeving van Apple. We gebruiken Xcode niet om te programmeren (dat doen we in Android Studio) maar wel om apps te configureren, instellingen op te geven, te testen en te exporteren naar de App Store van Apple.
- Een iOS-apparaat of simulator om te testen. Met Xcode wordt standaard de iOS-simulator geïnstalleerd, maar ook hier is het fijn als u daarnaast een echte iPad of iPhone hebt.

In de volgende paragrafen leest u hoe u alle onderdelen downloadt en installeert.

Alleen voor Windows:
Apps voor Windows kunt u alleen vanaf een Windows-computer exporteren. Hiervoor hebt u ook de gratis Visual Studio Community-omgeving nodig. Download en installeer het programma van **visualstudio.microsoft.com/downloads** of de Microsoft Store. Tijdens de installatie kunt u componenten kiezen. U heeft alleen **Desktop development with C++** nodig.

Alleen voor webapps:
Eenmaal gecompileerde Flutter-apps voor het web werken in alle moderne browsers. Maar om de apps tijdens het ontwikkelen te testen, hebt u de Edge-browser (Windows) of de Chrome-browser (alle systemen) nodig. Installeer deze als u hier niet over beschikt: zoek daarvoor op internet op `chrome`. U hoeft deze browser niet uw standaardbrowser te maken.

Alleen voor Linux en ChromeOS:
De procedure wijkt iets af van de beschrijving op de volgende pagina's en is zelfs iets eenvoudiger. Ga naar **https://docs.flutter.dev/get-started/install** en klik op uw platform voor de specifieke details.

2.2 De Flutter SDK installeren

De Flutter SDK installeert u zo:

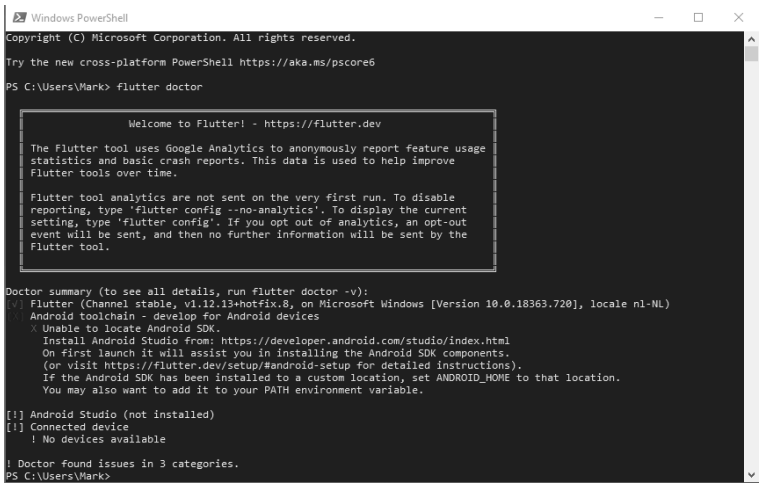
1. Surf naar de website **flutter.dev** en klik rechtsboven op **Get started** of typ `install flutter` in uw zoekmachine. U komt nu op de Flutter-webpagina Install terecht.
2. Klik op uw besturingssysteem (Windows, Mac, Linux of ChromeOS).
3. Download het installatiebestand. Klik daarvoor op de link *Get the Flutter SDK* en download de laatste stabiele versie.
4. Unzip het gedownloade pakket. Hierin zit één map met de naam `flutter`. Verplaats deze naar een locatie van waaruit u Flutter wilt gebruiken (bijvoorbeeld `/users/<naam>/flutter/` op een Mac of `C:\Users\<naam>\flutter` in Windows). Let op dat u een pad kiest zonder spaties, speciale tekens of schrijfbeveiliging. Dus bijvoorbeeld geen map in `C:\Program Files\`
5. Voorkom fouten door het pad te kopiëren. Selecteer daarvoor eerst de map waarin Flutter staat. Open dan het contextmenu terwijl u Shift indrukt (Windows) of open eerst het contextmenu en druk daarna op ⌥/Option (Mac). In het menu staat nu een extra optie: **Als pad kopiëren** of **kopieer 'flutter' als padnaam.** Selecteer deze.
6. Elke computer heeft een `PATH`-instelling. Die geeft aan in welke mappen het systeem zoekt als u een opdracht geeft. U moet de Flutter-map aan het `PATH` toevoegen.
   Op een Mac kan dat onder meer als volgt:
   - Open het programma Terminal.
   - Typ de opdracht `sudo nano /etc/paths`, druk op Return en geef uw wachtwoord op. Hiermee opent u een eenvoudige tekstverwerker (Nano) waarin u de `PATH`-instelling kunt bewerken.
   - Ga met de cursor naar de onderste regel en plak het pad uit stap 5, met daarachter `/bin`. Bijvoorbeeld `/Users/<naam>/flutter/bin`. (Als u het pad typt, gebruik dan ook op een Nederlandstalig systeem het Engelse 'Users' in plaats van 'Gebruikers')
   - Druk op Ctrl-X om Nano af te sluiten. Druk daarna op Y om op te slaan.

- Meld u af en weer aan of herstart de computer.
- U kunt de instelling controleren met de opdracht `echo $PATH`. (U ziet dat er veel meer paden bestaan dan die in het bestandje zijn opgenomen).

Zo past u `PATH` in Windows aan:

- Typ het woord `systeeminstellingen` in het zoekvak van het menu Start en kies **Geavanceerde systeeminstellingen weergeven**.
- Open het tabblad **Geavanceerd**. Klik hier in het onderste deel van het venster op **Omgevingsvariabelen**.
- In het bovenste deel van het venster staan instellingen voor de huidige gebruiker. Het onderste deel is voor alle gebruikers. Dubbelklik op de variabele **Path** in een van beide secties (of selecteer de variabele en klik op **Bewerken**).
- Klik op **Nieuw** en plak het pad naar de Flutter-map uit stap 5, met daarachter `\bin`. (Bijvoorbeeld: `C:\Users\<naam>\flutter\bin`). Gebruik ook op een Nederlands systeem het Engelse 'Users', vervang <naam> door de accountnaam op uw computer.
- Klik daarna op **OK** in elk venster.
- Meld u af en weer aan of herstart de computer.
- Open PowerShell (typ `shell` in het zoekvak van de Taakbalk). Geef de opdracht `$Env:Path` om te controleren of de Flutter-map goed aan het eind van de variabele is toegevoegd.

Typ nu de opdracht `flutter doctor` in Terminal (Mac) of Windows PowerShell. Hiermee controleert u de installatie. Als het goed is krijgt u een vinkje bij de eerste test (Flutter). Daaronder volgen foutmeldingen over het ontbreken van Android Studio en andere onderdelen. Daar werken we in de volgende paragraaf aan.



Afbeelding 2.1 *Flutter doctor in Windows PowerShell: Flutter is goed geïnstalleerd, maar andere onderdelen ontbreken nog.*

Problemen bij het installeren van de SDK

Krijgt u bij de opdracht `flutter doctor` een melding als 'Command not found' of 'The term 'flutter' is not recognized'? Ga dan naar de Flutter-map en probeer het daar nog een keer. Werkt het nu wel? Dan is het pad niet goed ingesteld. Zie stap 4 tot en met 6 hierboven. Let op: het pad dat u in de Nederlandse Verkenner of Finder ziet, is niet altijd hetzelfde als het echte pad: zo wordt 'users' weergegeven als 'gebruikers'. Als u het pad kopieert, zoals hierboven beschreven, voorkomt u problemen hiermee. Werkt het nog niet? Dan kan het zijn dat Flutter geïnstalleerd is op een plaats waarvoor u onvoldoende rechten hebt. Verplaats de Flutter-map, pas het pad aan en probeer het nog een keer. Krijgt u andere foutmeldingen? Lees deze goed. Soms ontbreken systeemonderdelen die nodig zijn voor een goede werking van Flutter. De melding geeft aan waar u die kunt downloaden.

2.3 Beginnen met Android Studio

Android Studio installeren

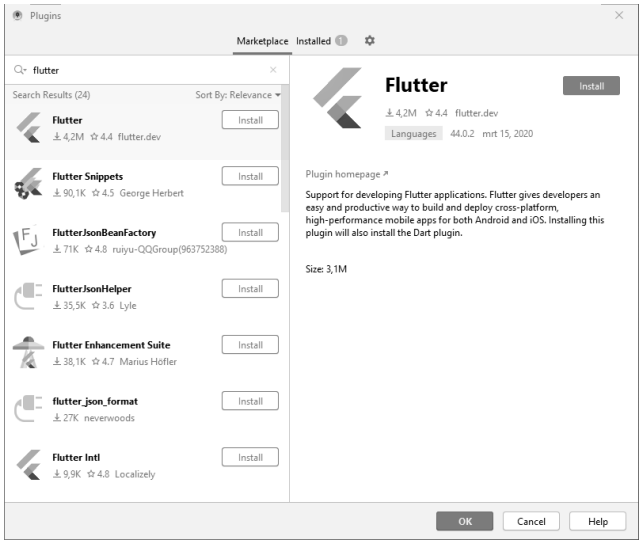
Of u nu op een Windows-, Apple- of Linux-computer werkt, in al deze gevallen is Android Studio een zeer complete omgeving om Flutter-apps te ontwikkelen.

1. Typ `download android studio` in uw zoekmachine.
2. Download het programma vanaf de gevonden pagina (onderdeel van **developer.android.com**). Dit kan lang duren.
3. Dubbelklik op het gedownloade installatiebestand om het installatieproces te starten, volg de instructies op het scherm en kies de gewenste opties. Maak gebruik van de mogelijkheid om *Android Virtual Device* (een simulatoromgeving voor Android) te installeren en accepteer de licenties.
4. Na de installatie kunt u het programma openen vanuit de programmamap op de computer (of via de automatisch gemaakte snelkoppeling).

Het welkomscherm en de Flutter-plug-in

Als u Android Studio opstart, verschijnt een welkomscherm. Hier kunt u wel een nieuw Android-project starten, maar de opties voor Flutter ontbreken nog. Daarvoor moet u eerst de Flutter-plug-in installeren:

1. Klik in het welkomscherm rechtsonder op **Configure** en kies de optie **Plug-ins**.
2. Zoek boven in het scherm op `flutter` en installeer de Flutter-plug-in. Een melding geeft aan dat ook Dart geïnstalleerd zal worden.
3. Start Android Studio opnieuw.



Afbeelding 2.2 *De Flutter-plug-in installeren.*

U ziet nu hetzelfde welkomscherm met een extra optie: **Start new Flutter Project**. In dit boek werken we alleen met deze Flutter-projecten. Verder zijn er op het welkomscherm opties om bestaande projecten te openen. Als u recente projecten hebt, verschijnen deze links in een lijst en kunt u ze openen door erop te klikken. U kunt een bestaand project ook openen met de optie **Open an existing Android Studio project**. Klik hierop, blader naar de hoofdmap van het project en klik op **Open**. Android Studio herkent automatisch dat het een Flutter-project is.

2.4 Een demoapp maken

1. Klik in het welkomscherm bovenaan op de knop **New Flutter project.**
2. In het vervolgscherm kunt u het **Flutter SDK path** opgeven. Dit is de plaats waar u eerder Flutter installeerde, bijvoorbeeld `/users/<naam>/flutter/`. Als alles goed is gegaan, staat dit al ingesteld. Pas het aan als dat niet zo is en klik op **Next**.
3. In het volgende scherm kunt u de volgende instellingen opgeven:
   - **Project name** De naam voor de app. Deze naam wordt ook gebruikt in de stores (tenzij u deze later aanpast). De naam mag alleen kleine letters en underscores bevatten. Kies een naam die u nog niet voor een andere app gebruikte.
   - **Project location** De plaats waar u het Flutter-project wilt opslaan. Zorg voor voldoende schijfruimte: Flutter-projecten gebruiken vele honderden MB's, vooral als u voor veel platformen ontwikkelt.

- **Description** Omschrijving van het project.
- **Project type** Soort project. Laat dit op **Application** staan.
- **Organization** De organisatie moet samen met de projectnaam een unieke naam vormen. Het is gebruikelijk om hier de zogenoemde *reverse domain name notation* te gebruiken; bijvoorbeeld `nl.bedrijfsnaam`. Gecombineerd met de projectnaam wordt dat dan automatisch iets als `nl.bedrijfsnaam.mijnapp`.
- **Android language** en **iOS language** Laat deze op Kotlin en Swift staan.
- **Platforms** Hier kunt u aangeven voor welke platformen u de app wilt ontwikkelen. Later kunt u gemakkelijk platformen toevoegen.
- **Create project offline** Hiermee wordt de gecompileerde versie van het project op de vaste schijf bewaard. Daardoor werkt de app sneller bij het testen, maar hebt u meer schijfruimte nodig.
- **More settings** Laat deze paden en bestandsnamen ongemoeid.

Dat was het. Het systeem is even bezig, maar dan hebt u ook wat: de eerste Flutter-app is gemaakt! Alleen, we hebben nog niets om de app op uit te voeren en te kijken wat deze doet. Dat doen we verderop in dit hoofdstuk. Eerst kijken we naar Android Studio en de structuur van de app.

Dart op een Mac

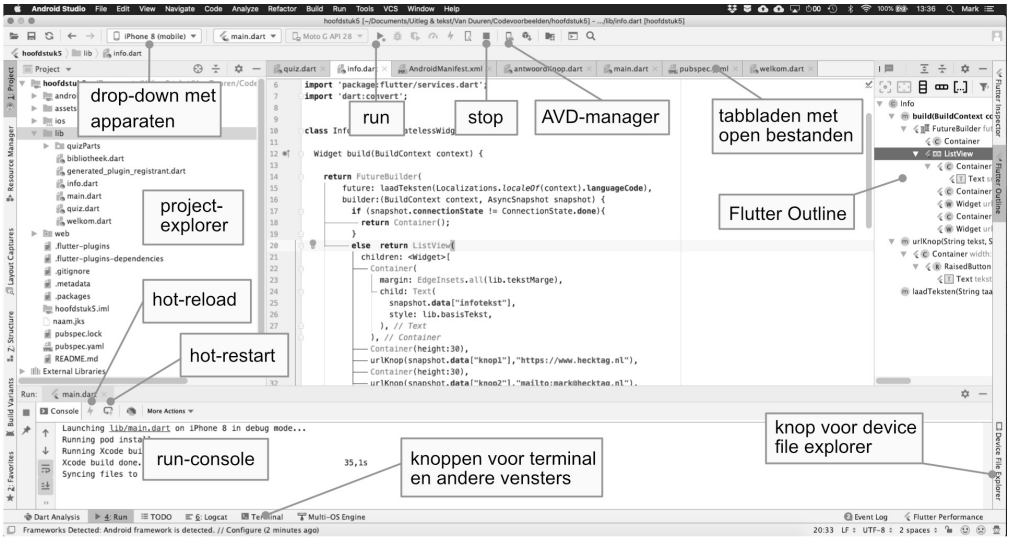
Krijgt u op een Mac bij het maken van de eerste app een melding dat Dart niet geopend kan worden omdat het afkomstig is van een onbekende ontwikkelaar? Dat lost u zo op. Klik linksboven op het Apple-pictogram en ga naar **Systeemvoorkeuren**, **Beveiliging en privacy**, tabblad **Algemeen**. Open het hangslotje linksonder. Zet de onderste instelling op **App store en ontwikkelaars waarvan de identiteit bekend is** en klik op **Sta toch toe** bij de melding over Dart.

2.5 Android Studio gebruiken

Het valt buiten het bestek van dit boek om alle mogelijkheden van Android Studio te bespreken. We noemen hieronder daarom alleen de belangrijkste functies voor het ontwikkelen van Flutter-apps, die u ook in de afbeelding ziet. Enkele andere functies komen later in dit boek aan bod.

De Project Explorer en de structuur van de app

Aan de linkerkant van het scherm vindt u enkele tabs. Verreweg de belangrijkste is de tab **Project**, die standaard geopend is. Daarin ziet u uit welke bestanden en mappen de app bestaat. Deze projectbrowser werkt ongeveer zoals Verkenner in Windows of Finder op een Mac. De structuur die u hier ziet, is ook de werkelijke structuur op de schijf. U kunt bestanden vanaf andere locaties op de computer hier naartoe slepen of op een andere manier kopiëren.



Afbeelding 2.3 *Enkele belangrijke onderdelen van Android Studio.*

Open de hoofdmap (met de naam van de app) in de Project Explorer om de structuur te zien. Elke Flutter-app bestaat uit dezelfde hoofdonderdelen:

- De map **lib**: het hart van de app. Alle code die u schrijft komt hier terecht, al dan niet in submappen. Hier staat nu nog maar één bestand: main.dart, het beginpunt van de app.
- Mappen met namen van besturingssystemen: **ios** en **android**. Hierin staan alle bestanden die specifiek zijn voor het desbetreffende platform. Als u de app ook geschikt maakt voor macOS, Windows, Linux of web, dan verschijnen er ook mappen met deze namen. Tijdens het ontwikkelen van de app hebt u deze mappen niet nodig. U past hier alleen de laatste zaken voor publicatie aan (zie hoofdstuk 6).
- De map **Test**: hier kunt u zogenoemde unittests doen. Dat is vooral handig als u met meerdere mensen aan een app werkt. In dit boek zullen we tests gewoon in de app zelf uitvoeren. U kunt deze map veilig verwijderen (via het menu **Edit**, **Delete** of het contextmenu).
- In de map **assets** komen afbeeldingen, clips, geluidsbestanden, lettertypen en andere bronnen voor uw app. Omdat die er nu nog niet zijn, ontbreekt de map.
- Buiten deze mappen ziet u enkele bestanden. Vooral **pubspec.yaml** is belangrijk. Hierin worden appinstellingen bewaard die zowel voor Android als iOS belangrijk zijn, zoals de naam en het versienummer. Ook gebruikt u dit bestand om Dart-bestanden, afbeeldingen en andere bronnen te importeren. In hoofdstuk 4 en 5 gaan we hiermee aan de slag.

Vensters

Aan de onderkant van het scherm[1] kunt u verschillende vensters openen. U schakelt tussen vensters met de knoppen helemaal onderaan in het scherm. De belangrijkste twee zijn:

- **Terminal** Dit scherm vervangt de Terminal of PowerShell van het besturingssysteem. U kunt hier nu bijvoorbeeld opnieuw de opdracht `flutter doctor` geven om de status van de installatie te controleren. In dit venster geeft u ook de opdrachten om de app te compileren of om een simulator te starten.
- **Run-console** Dit venster is zichtbaar als er een app wordt uitgevoerd. U ziet hier bijvoorbeeld foutmeldingen of berichten die de code op het scherm plaatst. Dit venster bevat ook de knoppen **Hot reload** en **Hot restart**, waarmee u bliksemsnel, meestal binnen een seconde, wijzigingen doorvoert in een werkende app.

Flutter-vensters

Aan de rechterkant van het scherm vindt u tabs met specifieke Flutter-hulpmiddelen. Ze werken alleen als er een app wordt uitgevoerd:

- Flutter Outline gebruikt u bij het ontwikkelen van Flutter-apps. Hier ziet u de structuur van de widgets in de app. Met de knoppen aan de bovenkant van dit venster kunt u gemakkelijk widgets uit elkaar halen of juist in elkaar plaatsen.
- Flutter Performance maakt de snelheid van verschillende onderdelen zichtbaar. Als een app heel traag werkt, kunt u hier achterhalen welk onderdeel daarvoor verantwoordelijk is.
- Flutter Inspector helpt u bij het analyseren van een app en het oplossen van problemen.
- Bij Device File Explorer kunt u bestanden en mappen bekijken op het apparaat waarop de Flutter-app wordt uitgevoerd.

Menu- en knoppenbalken

Bovenaan het scherm vindt u de menu- en knoppenbalken. De functies op de knoppenbalk staan ook in de menu's.

- In de menu's staan veel algemene taken die u kent van andere programma's. Hiermee kunt u bestanden openen, opslaan, kopiëren, plakken enzovoort.
- Er zijn ook menu's voor specifieke programmeertaken. Zo is er een **Code**-menu om Dart-code te bewerken. Enkele functies daarvan komen in het volgende hoofdstuk terug.

1 Met de instellingenknop op de bovenrand van dit venster kunt u de positie wijzigen. Probeer bijvoorbeeld Move to, Top left.

- Belangrijk is het menu met apparaten in de knoppenbalk. Standaard staat hier *<no device selected>*. Hier kiest u op welk echt of virtueel apparaat de app start als u op **Run** klikt. In de volgende paragrafen gaan we hier apparaten aan toevoegen.
- Op de knoppenbalk staat ook een **Run**-knop (groen driehoekje) om een app te starten. Als de app eenmaal werkt, gebruikt u meestal de knoppen in het eerder beschreven run-venster om een app te verversen of te herstarten. Naast de **Run**-knop staat een **Stop**-knop (rood vierkantje) om de app te stoppen.
- De AVD (Android Virtual Device)-manager vindt u zowel in het menu (onder **Tools**) als op de knoppenbalk. Hiermee beheert u de virtuele Android-apparaten waarop u apps kunt testen. Dat is het onderwerp van de volgende paragraaf.

Opslaan en meer

Zoals in alle applicaties is het verstandig om uw werk vaak op te slaan. Dat kan snel met de toetsencombinatie ⌘+S (Mac) of Ctrl+S (andere systemen). Handig is dat u hiermee standaard nog twee acties uit kunt voeren:

- U ververst de weergave van de app (met *hot reload*, zie hieronder).
- U zorgt ervoor dat de code overzichtelijk gestructureerd wordt.

De eerste actie staat standaard aan, de tweede uit. Maar deze kunt u natuurlijk aanzetten. De instellingen vindt u in het menu van Android Studio, onder **Android Studio**, **Preferences** (Mac) of **File**, **Settings** (Windows). Ga hier naar **Languages & Frameworks**, **Flutter**. De instellingen zijn:

- **Perform hot reload on save** (ververst de lopende app als u uw werk opslaat).
- **Format code on save** (maakt uw code op als u uw werk opslaat).

Hot reload, hot restart en volledige herstart

De hot reload- en hot restart-functies maken het werken met Flutter een stuk gemakkelijker en leuker. Het is handig als u de verschillen kent:

- De **hot reload**-functie vernieuwt bliksemsnel een werkende app. Veranderingen in kleuren, lay-out, teksten en andere zaken ziet u zo onmiddellijk. Deze actie is dus standaard gekoppeld aan het opslaan (maar u kunt ook het knopje in het runvenster gebruiken).
- Voor een **hot restart** moet u op het knopje boven in het runvenster klikken (of klik in het menu op **Run**, **Hot restart** of gebruik de daar vermelde snelkoppeling). Hiermee wordt de app op het apparaat opnieuw gestart. Voor de demoapp betekent dit bijvoorbeeld dat de teller weer op 0 komt te staan (bij een hot reload behoudt deze zijn waarde). Als u in de code structurele zaken wijzigt (zoals animaties, interacties, definitie van variabelen enzo-

voort) dan is een hot restart noodzakelijk. Maar ook dat kost meestal minder dan een seconde.

- Als u de app helemaal stopt (**Run**, **Stop** of gebruik de stopknop in de werkbalk) en daarna opnieuw start, dan wordt deze opnieuw gecompileerd. Het systeem maakt een nieuw pakket van alle bronnen en code en plaatst dit opnieuw in de omgeving waarop u test. Dit duurt een stuk langer dan de hot reload en hot restart. Dit is alleen nodig als u bronnen (zoals afbeeldingen, geluiden of video) vervangt of als de app helemaal vastloopt.

2.6 Bouwen en testen voor meer platformen

U hoeft Flutter-apps natuurlijk alleen te testen op platformen waarvoor u ontwikkelt. Als u al weet dat u de app alleen voor iOS bouwt, hoeft u bijvoorbeeld niet op Android of Windows te testen. In de volgende paragrafen leest u hoe u op verschillende platformen test. Kies daarbij alleen de platformen die voor u van belang zijn. Aandachtspunten daarbij zijn:

- Als u mobiele apps bouwt (dus Android of iOS), is het belangrijk dat u zowel in een simulator als op een echt apparaat test. In de simulator kunt u veel verschillende apparaattypen, schermformaten en dergelijke uitproberen. Een test op een echt apparaat is nodig omdat zaken er daar toch vaak wat anders uitzien.
- Als u foutmeldingen krijgt bij het starten van een nieuwe of bestaande app, kijk dan naar de laatste paragraaf van dit hoofdstuk. Daar leest u hoe u veel voorkomende problemen oplost.

Platformen inschakelen

De standaard demoapp die u in paragraaf 2.4 maakte, is geschikt voor de daar aangevinkte platformen. Daarom ziet u alleen mappen voor deze besturingssystemen in het project. Om een app geschikt te maken voor alle platformen, doet u het volgende:

1. Selecteer de hoofdmap van het project (helemaal boven in het projectpaneel).
2. Open het terminalvenster in het onderste deel van het scherm.
3. Typ hier de opdracht: `flutter create .` (let op de spatie en punt op het eind). Flutter maakt mappen voor alle beschikbare besturingssystemen en plaatst daarin de benodigde bestanden.