Mobiele apps voor Android en iOS

tegelijk geprogrammeerd

Dr. Veikko Krypczyk (Duitsland)

Voor steeds meer toepassingen (bijvoorbeeld rond het Internet of Things) is een app voor mobiele apparaten nodig. En meestal moet die zowel onder Android als onder iOS draaien. Er zijn dan allerlei verschillende benaderingen die strijden om de gunst van de ontwikkelaar. Daarbij zijn een platformonafhankelijke aanpak en een intuïtieve manier voor het creëren van de gebruikersinterface van belang. De ontwikkelomgeving RAD Studio (beter bekend als Delphi) kan een geschikte keuze zijn.

Steeds meer toepassingen op het gebied van elektronica berusten op een uitgekiende interactie van hardware en software. Apps voor mobiele apparaten zoals tablets en smartphones met Android- en iOS-besturingssystemen worden steeds vaker gebruikt. De uitdaging is om voor beide systemen een individuele app te maken, die precies is afgestemd op de eisen van het project in kwestie. Als het ontwikkelen van mobiele apps niet uw dagelijks werk is, kan het lastig zijn om daaraan te beginnen. Verschillende benaderingen wedijveren met elkaar, vaak zijn de leercurves steil. Zo wordt het programmeren van deze kleine apparaten een groot probleem, misschien zelfs een struikelblok voor het hele project. Dit kan worden voorkomen met platformonafhankelijke benaderingen die een grotendeels intuïtieve aanpak van het ontwerp van de gebruikersinterface ondersteunen.

In dit artikel geven we een beknopt overzicht van de mogelijkheden om mobiele apps te ontwikkelen, voordat we ingaan op een interessante benadering. RAD Studio is een efficiënt tool voor het ontwikkelen van desktopapplicaties. Bij veel ontwikkelaars is het ook bekend onder de naam Delphi. We kunnen daarmee tegenwoordig ook apps voor Android en iOS maken vanuit één broncode-basis, wat het leven een stuk eenvoudiger maakt. Het programmeren gaat met Delphi (Object Pascal) en voor de gebruikersinterface is er uitgebreide ondersteuning met een grafisch ontwerptool. Aan de hand van een voorbeeld zullen we dat demonstreren.

Apps voor mobiele apparaten

Vanuit technisch oogpunt kunnen we de volgende soorten apps onderscheiden:

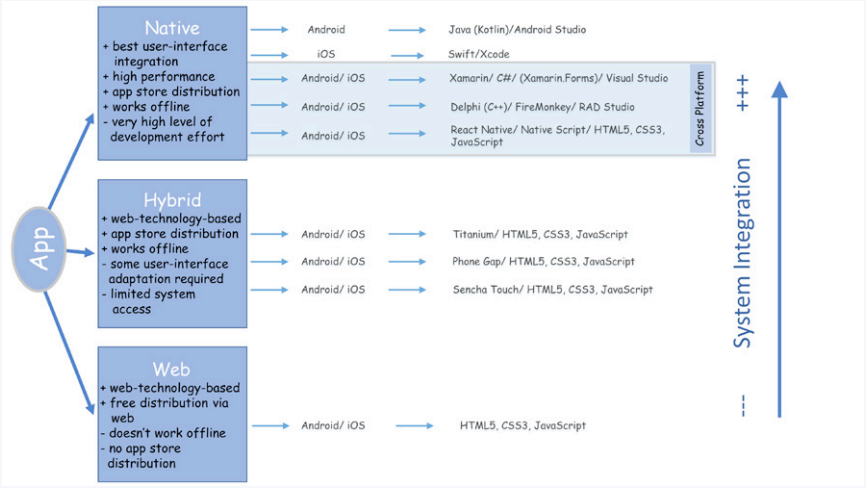
- native apps
- web-apps
- hybride apps

Native apps worden uitsluitend gebouwd voor Android of voor iOS. Ze maken rechtstreeks gebruik van de API van die systemen.

De gebruikersinterface past dan optimaal in het platform. Een ander voordeel is dat ze geen beperkingen hebben bij de toegang tot eventuele apparaatspecifieke hardware van de smartphone of het tablet. Alle sensoren van de apparaten kunnen direct worden aangesproken. De app wordt beschikbaar gesteld via de app-stores. Als een native app is geïnstalleerd, kan deze ook zonder internettoegang (offline) worden uitgevoerd, als dat zinvol is. Noodzakelijke datasynchronisatie kan automatisch plaatsvinden wanneer er een internetverbinding beschikbaar is. Apps voor iOS (Apple) worden gemaakt met Xcode (ontwikkelomgeving) en Swift (programmeertaal). Voor Android worden Android Studio en Kotlin of Java gebruikt.

Web-apps daarentegen zijn op mobiele apparaten afgestemde webapplicaties. Dit betreft vooral het ontwerp van de gebruikersinterface. De toegang tot de systeemhardware is beperkt. Enkele basisfuncties, zoals plaatsbepaling via GPS, zijn echter wel mogelijk. Hiervoor worden de betreffende JavaScript-API's gebruikt. Het is niet mogelijk om dit soort apps aan te bieden via de app-stores. Ze draaien op een server en hebben daarom constant een internetverbinding nodig. Het is wel mogelijk om een pictogram in te stellen op het startscherm, zodat gebruikers de app snel kunnen starten. De eis om altijd online te moeten zijn, kan gedeeltelijk worden opgelost door gebruik te maken van zogenaamde Progressive Web App (PWA)-technologie. Een PWA is een symbiose van een website en een app. Met behulp van een Service Worker kan een cachingfunctie worden uitgevoerd. De Service Worker verbindt de webserver met de app op het mobiele apparaat. Het hele scala aan webtechnologieën is beschikbaar om een webapplicatie voor mobiele systemen te maken, in het bijzonder de gebruikelijke bibliotheken en frameworks. De apps zijn uiteindelijk gebaseerd op HTML (structuur), CSS (ontwerp) en JavaScript (interactie, logica), want dat is het enige waar de browser mee overweg kan.

Hybride apps zijn intern gebaseerd op webtechnologieën. De webapplicatie wordt uitgevoerd in een embedded webbrowser, hij draait dus in een soort "zandbak", waardoor hij platformonafhankelijk is. Het systeem denkt daarom dat het om een native app gaat (de browser). Er bestaan verschillende benaderingen voor hybride apps, bijvoorbeeld Cordova van de Apache Software Foundation. Het principe is altijd hetzelfde. De app opent bij het opstarten een browservenster in full screen-modus, zodat de browser niet als zodanig herkenbaar is en het webadres niet kan worden gewijzigd. De uit HTML, CSS en JavaScript opgebouwde webapplicatie wordt weergegeven in de browser. De app krijgt met behulp van plug-ins toegang tot systeemfuncties zoals de camera of het adresboek.
Web-apps en hybride apps richten zich zowel op Android als op iOS. In vergelijking met native apps hebben ze nadelen, zoals slechtere prestaties en begrensde toegang tot de hardware. Daarom probeert men de voordelen van beide soorten apps te combineren met behulp van *Cross-Platform*-oplossingen. Het doel is om zo dicht mogelijk bij het native model te komen (ook in de *look and feel* van de gebruikersinterface) en de app voor Android en iOS te maken vanuit een gemeenschappelijke broncode-basis. Er bestaat een groot aantal cross-platform benaderingen, die zeer verschillend zijn in hun aanpak, bijvoorbeeld Xamarin, RAD Studio, NativeScript en React Native. De belangrijkste eigenschappen zijn samengevat in **figuur 1**.

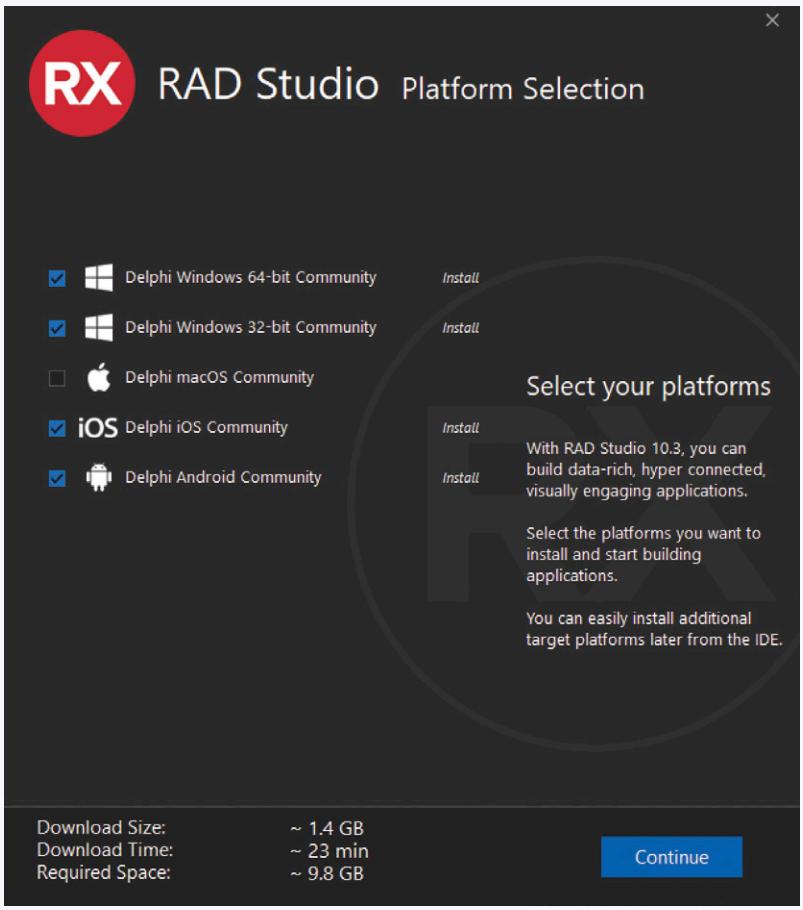


Figuur 1. Soorten apps en manieren om ze te ontwikkelen.

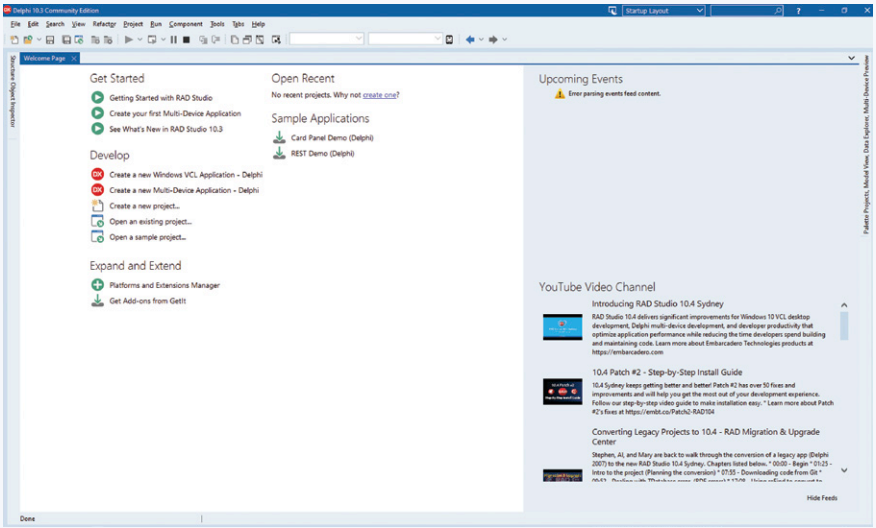
In dit artikel gaan we dieper in op de procedure met *RAD Studio (Delphi)*. Deze aanpak lijkt vooral geschikt te zijn voor elektronica- en IoT-projecten. De procedure is bij veel ontwikkelaars bekend. Met behulp van een grafisch ontwerptool kan de gebruikersinterface worden gemaakt, vergelijkbaar met het ontwikkelen voor de Windows-desktop. De logica wordt geschreven in de taal Delphi (Object Pascal), die intuïtief en niet bijzonder complex is. Sommige elektronica-ingenieurs

RAD STUDIO, DELPHI EN C++ BUILDER

Delphi en C++ Builder zijn de geïntegreerde ontwikkelomgevingen van Embarcadero voor het bouwen van toepassingen voor verschillende besturingssystemen. Delphi gebruikt de gelijknamige Delphi-programmeertaal (een verdere ontwikkeling van Object Pascal), terwijl C++ Builder C++ gebruikt. In RAD Studio worden beide omgevingen gecombineerd. Er zijn verschillende edities beschikbaar. Voor professioneel gebruik zijn er de commerciële producten Professional, Enterprise en Architect. Voor een beperkt commercieel gebruik, voor testen, leren, voor privé- en open source-projecten is er de gratis Community Edition. Die gebruiken we ook hier. De huidige versie van RAD Studio is 10.4.



Figuur 2. Keuze van de platformen.



Figuur 3. Startscherm van de geïntegreerde ontwikkelomgeving van Delphi.

zullen ook vertrouwd zijn met Pascal, bijvoorbeeld van het programmeren van embedded software voor microcontrollers.

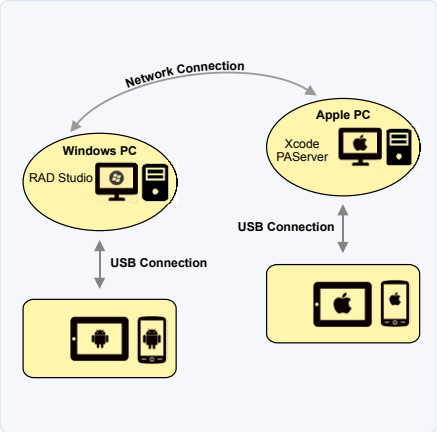
Installatie en systeemconfiguratie

We installeren Delphi op Microsoft Windows en richten het in voor mobiele ontwikkeling. Ga naar [1], registreer u, download de Community Editie en voer het installatieprogramma uit. Tijdens de installatie ziet u een dialoogvenster met opties om verschillende platformen te installeren (**figuur 2**).

Selecteer naast *Windows* ook *Android* en *iOS*. Vervolgens wordt u gevraagd om extra opties te selecteren. Dit moet u doen voor de *Android SDK/NDK*. Start na de installatie de ontwikkelomgeving (**figuur 3**).
Voor het ontwikkelen van mobiele apps is nog wel wat extra werk nodig. Als u de app op iOS wilt testen, hebt u toegang via het netwerk nodig tot een Apple PC met Xcode. Een simulator voor iOS kan alleen worden uitgevoerd onder macOS. Zelfs een fysieke iPhone is direct gekoppeld aan de Mac PC. De Mac-PC is dan met RAD Studio vanuit Windows toegankelijk via het netwerk (**figuur 4**).

Om dit te kunnen doen, moet op de Mac de software *PAServer* worden geïnstalleerd voor remote toegang. Als u op een Mac werkt, kunt u Windows in een virtuele machine draaien (Parallels, Virtual Box enzovoort) en RAD Studio in die virtuele machine installeren. Vanaf deze virtuele machine is de Mac PC dan toegankelijk. Alle componenten voor de ontwikkeling zijn dan op één computer geïnstalleerd. De noodzaak om gebruik te maken van een Mac om programma's voor iOS te bouwen is geen eigenaardigheid van RAD Studio, dat is altijd nodig om voor iOS te kunnen programmeren.

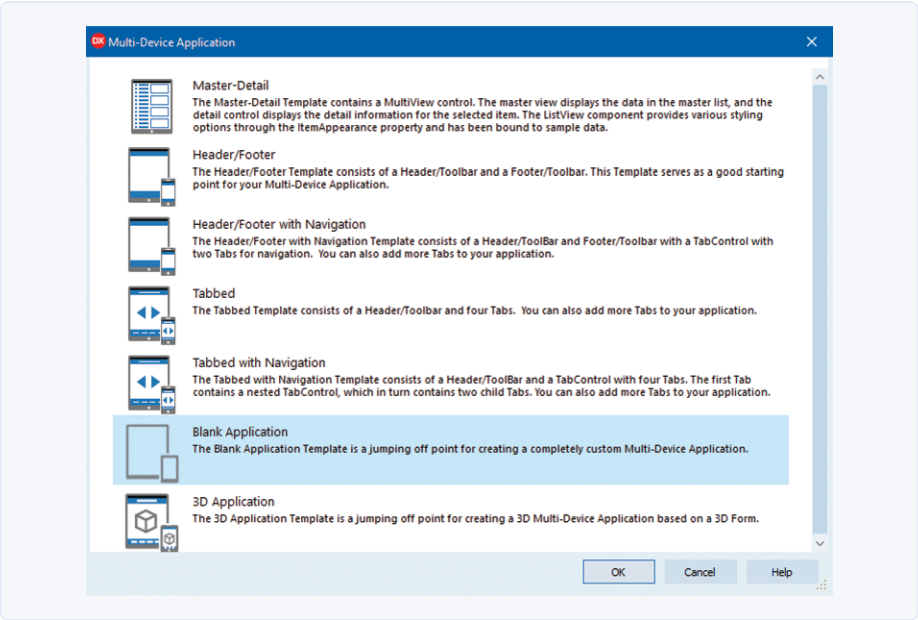
Voor Android kunt u apps direct vanuit Windows maken. Voor het testen kunt u een smartphone/tablet via USB op uw computer aansluiten of (als alternatief) een emulator installeren. Een 'echt' apparaat werkt veel sneller dan een emulator. Sluit uw smartphone aan op de computer met een USB-kabel en configureer deze op de juiste manier: schakel bij de instellingen op de smartphone ontwikkelaarsrechten in, inclusief debuggen via USB. Controleer nu in Apparaatbeheer of uw mobiele apparaat correct is herkend en ingesteld. De officiële documentatie voor het opzetten van Delphi voor mobiele ontwikke-



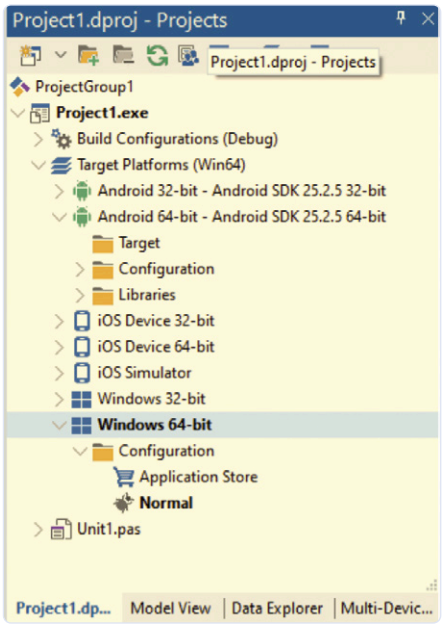
Figuur 4. Systeemconfiguratie voor het ontwikkelen van apps voor iOS.

GEEN iOS ZONDER macOS EN Xcode

iOS is een gesloten systeem; om de app-packages te maken en te distribueren naar de mobiele apparaten is Xcode (geïntegreerde ontwikkelomgeving) en dus een Mac met macOS nodig. Ook de iOS-simulator is alleen onder macOS te gebruiken. U kunt ook gebruik maken van een cloudservice, dat wil zeggen een gehoste Mac-PC met alle benodigde ontwikkeltools 'huren' en deze via een verbinding op afstand bedienen. De service MacInCloud [2] biedt deze mogelijkheid. Het scherm wordt dan overgebracht naar de eigen ontwikkelcomputer. Dit lost het probleem op, zodat u het app-package kunt maken en de iOS-simulator kunt gebruiken tijdens het programmeren. Een echt apparaat kan natuurlijk niet via de cloud worden gebruikt. De auteur heeft goede ervaringen met de genoemde dienst. De software PAServer voor de afstandsbediening van de Mac is al voorgeïnstalleerd, zodat u direct na registratie en aanmelding kunt beginnen. Om te beginnen is er het flexibele Pay-as-You-Go-plan, dat wil zeggen dat u alleen betaalt voor de tijd dat u de Mac in de cloud werkelijk hebt gebruikt.



Figuur 5. Aanmaken van een nieuwe cross-device applicatie.



Figuur 6. Doelplatformen worden rechtstreeks in Delphi geselecteerd

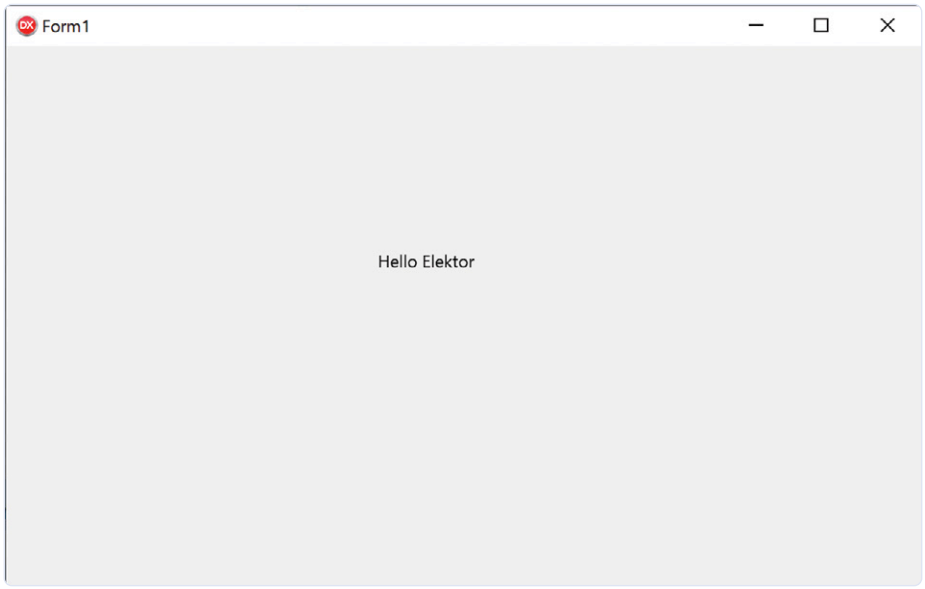
ling is te vinden onder [3] en [4]. Het configuratiewerk is nu klaar. Een eerste test ("*Hello World*") illustreert de procedure voor het ontwikkelen van een mobiele app.

Hello Elektor

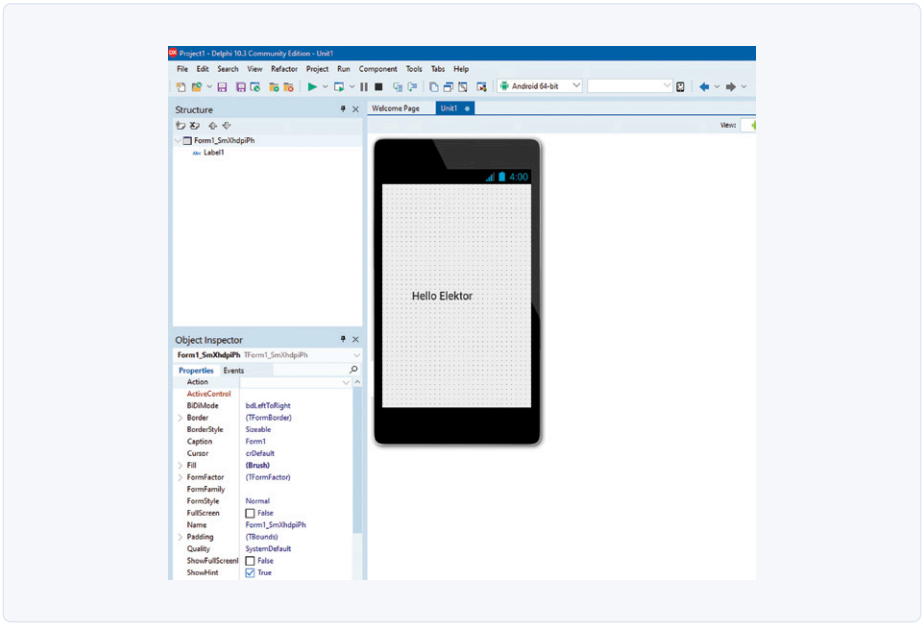
We gaan nu een eerste project maken. Kies in de Delphi-ontwikkelomgeving *File* | *New* | *Multi-Device-Application – Delphi* en klik in het venster op *Blank Application* (**Figuur 5**) en daarna op *OK*.

Sla het project op. Met behulp van het *Tool*-palet kunnen we de gebruikersinterface samenstellen met behulp van visuele besturingselementen. Sleep een *TLabel*-component naar het dialoogvenster voor een test. U kunt de preview omschakelen tussen de verschillende systemen (Windows, Android, iOS).

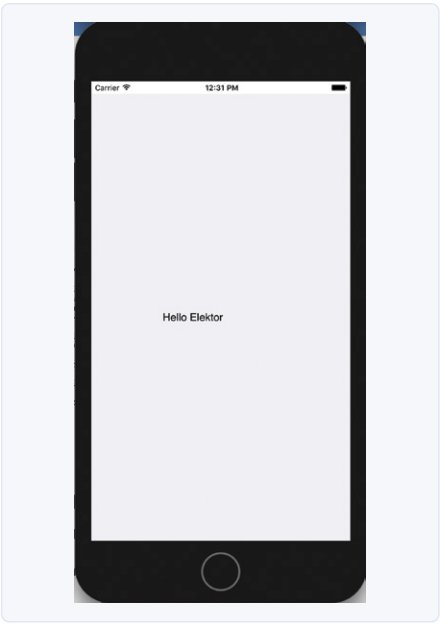
In de *Object Inspector* kunnen de eigenschappen van de besturingselementen



Figuur 7. De testapplicatie als een Windows-toepassing.



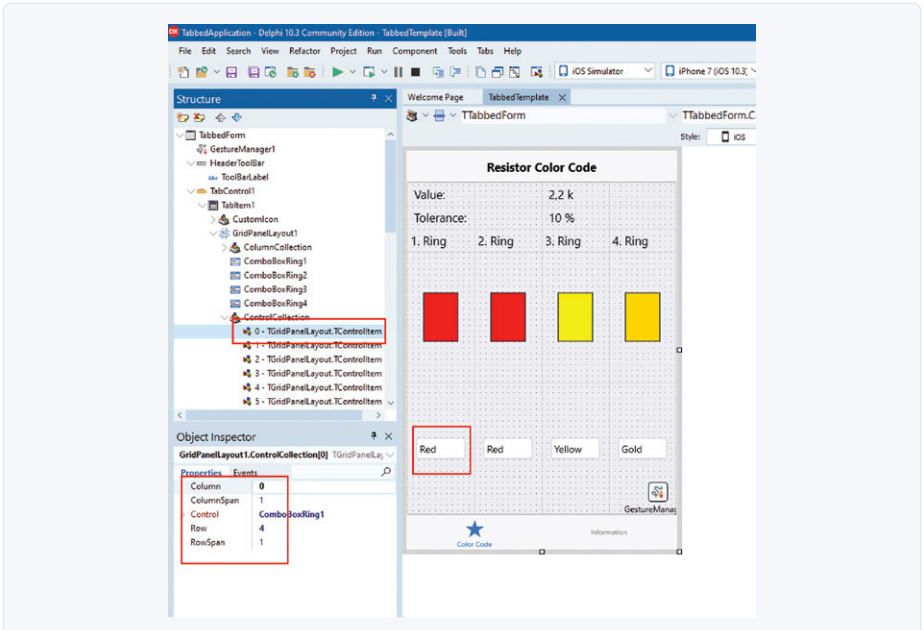
Figuur 8. Preview van de app voor Android in RAD Studio Designer.



Figuur 9. "Hello Elektor" op de iPhone-simulator.

worden geconfigureerd. We stellen de eigenschap *Tekst* in op de waarde "*Hello Elektor*". Sla het project en de bijbehorende bestanden op via het menu *File*. Nu kunnen we al gaan testen op de verschillende platformen. Open in de *Projectmanager* het knooppunt *Target Platforms* en selecteer het platform *32-bits Windows* door erop te dubbelklikken (**figuur 6**). Start de toepassing (groene pijl). Hij wordt uitgevoerd als een 'normale' Windows-toepassing (**figuur 7**).

Sluit nu de applicatie af en ga terug naar de ontwikkelomgeving. Selecteer nu *Android 64-bit* als doelplatform. Als de smartphone/tablet correct is geconfigureerd, verschijnt hier nu het aangesloten apparaat. Activeer het en start de app opnieuw. De app wordt geïnstalleerd en uitgevoerd op het mobiele apparaat. U kunt een voorbeeld van de app voor een Android-toestel ook bekijken in de *Designer* van de ontwikkelomgeving (**figuur 8**).
Als u het systeem ook voor iOS hebt geconfigureerd, kunt u verbinding maken met de Mac PC met PAServer en Xcode (zie boven). Dan kunt u het programma uitvoeren in de simulator en op een iPhone (**figuur 9**).
Na deze minimale introductie gaan we nu een uitgebreidere app bouwen om de werkwijze beter te leren kennen.



Figuur 10. De positionering van de besturingselementen gebeurt hier in een tabelraster.

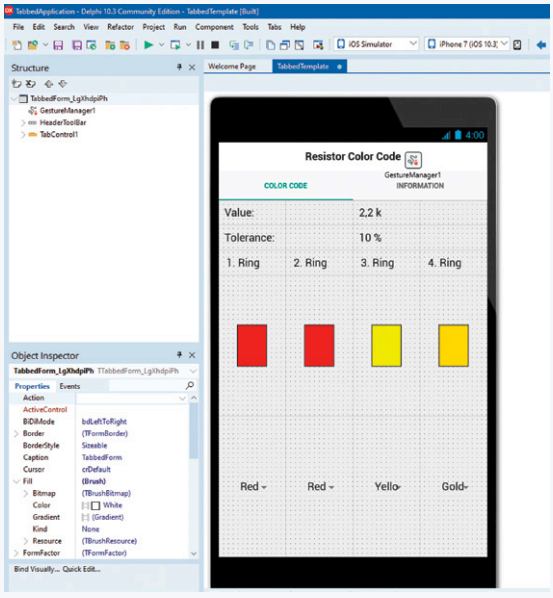
Een app voor elektronici

We beschrijven nu de ontwikkeling van een app voor het bepalen van de waarde van een weerstand aan de hand van de kleurringen. Eerst ontwerpen we de interface in de grafische designer. Kies opnieuw *File* | *New* | *Multi-Device-Application – Delphi* en kies dan *Tabbed*. We krijgen dan een weergave met tabbladen (in Android bovenaan, in iOS onderaan). We moeten nu de inhoud van deze tabbladen invullen voor onze applicatie. Het sjabloon bevat vier tabbladen (type *TTabItem*). We hebben er maar twee nodig, dus we verwijderen twee tabbladen met behulp van de dialoog *Structure*. In tabblad één selecteren we de elementen voor de kleurcode en in tabblad twee plaatsen we algemene informatie over de app. Het is een goed idee om eerst een schets te maken voordat u de gebruikersinterface gaat implementeren, zodat u weet waar de besturingselementen moeten worden geplaatst (potlood en papier zijn voldoende). De rangschikking van de besturingselementen gebeurt meestal niet met absolute positie-informatie, omdat we er rekening mee moeten houden dat de app gebruikt wordt op mobiele apparaten met verschillende schermafmetin-

Kleur	Ring 1 Positie 1	Index in TComboBox	Ring 2 Positie 2	Index in TComboBox	Ring 3 Factor	Index in TComboBox	Ring 4 Tolerantie	Index in TComboBox
zwart	0	0	0	0	1	0	-	-
bruin	1	1	1	1	10	1	1%	0
rood	2	2	2	2	100	2	2%	1
orange	3	3	3	3	1.000	3	-	-
geel	4	4	4	4	10.000	4	-	-
groen	5	5	5	5	100.000	5	0,5%	2
blauw	6	6	6	6	1.000.0000	6	0,25%	3
violet	7	7	7	7	10.000.000	7	0,1%	4
grijs	8	8	8	8	-	-	0,05%	5
wit	9	9	9	9	-	-	-	-
goud					0,1	8	5%	6
zilver					0,01	9	10%	7

Tabel 1. Betekenis van de kleurringen en de positie (index) in elk keuzevak (TComboBox).

gen en resoluties. Een tabelstructuur kan een goed uitgangspunt zijn voor het verdelen van de ruimte. De afzonderlijke elementen worden dan gerangschikt in rijen en kolommen. Voor elke rij en kolom kunnen we de afmetingen aangeven (absoluut of – beter – relatief). We kiezen voor onze app een raster met vijf rijen en vier kolommen. Gebruik een element van het type *TGridPanelLayout*. Selecteer dit uit het palet en sleep het naar het eerste tabblad. Om alle ruimte te benutten, stelt u de eigenschap *Align* in op de waarde *Client*. Voeg met behulp van de dialoog *Structure* in totaal vijf rijen en vier kolommen toe aan deze *TGridPanelLayout*. U kunt de grootte van de rijen bijvoorbeeld procentueel als volgt verdelen: 7%, 7%, 7%, 40% en 39% (totaal = 100%). Verdeel de vier kolommen gelijk, dat wil zeggen, elk met een grootte van 25%. Nu moeten de besturingselementen in deze tabelstructuur worden geplaatst. Selecteer een gewenst element uit het palet en sleep het naar het tabblad. Voor de kleurselectie gebruiken we besturingselementen van het type *TComboBox*. In de *Structure View* kunnen we dan de exacte positie (rij en kolom) voor elk element instellen onder *TabItem1 | Control Collection*. In **figuur 10** is dit te zien voor de kleurkeuze van de eerste ring.
We kiezen als positie *Column: 0* en *Row: 4*, want daar moet het keuzevakje komen te staan. Voor teksten, bijvoorbeeld voor labels of aanwijzingen, gebruiken we besturingselementen van het type *TLabel*. We gebruiken *TLabel* ook voor waarden die we later willen weergeven, bijvoorbeeld voor de berekende weerstandswaarde. We simuleren de gekleurde ringen met gekleurde rechthoeken, dus besturingselementen van het type *TRectangle*. We willen hun kleur later aanpassen vanuit de broncode. In de keuzevelden voor de kleurringen (*TComboBox*) voegen we de beschikbare kleuren toe volgens het patroon uit **tabel 1**.
De layout van de applicatie in de grafische designer is te zien in **figuur 11**. Na het starten van de app (in de simulator of op een fysiek apparaat) zien we de ontworpen layout van het tabblad (**figuur 12**).
We hebben het tabblad *Information* in eerste instantie alleen gevuld met een logo (type *TImage*) en een tekst (zie **figuur 13**).
Als u zover bent gekomen, is de volgende stap het schrijven van de code voor het berekenen van de weerstandswaarde. Telkens wanneer we een nieuwe kleurwaarde selecteren, moet er een herberekening plaatvinden en moet



Figuur 11. De layout van de Delphi-app in Android-weergave.

Listing 1. Aanpassen van de kleurweergave aan de selectie.

```
case digit1 of
    0:
    begin
        RectangleRing1.Fill.Color := TAlphaColors.Black;
    end;
    1:
    begin
        RectangleRing1.Fill.Color := TAlphaColors.Brown;
    end;
    2:
    begin
        RectangleRing1.Fill.Color := TAlphaColors.Red;
    end;
    …
    …
    9:
    begin
        RectangleRing1.Fill.Color := TAlphaColors.White;
    end;
end;
```

Listing 2. Berekening van de weerstandswaarde en de weergave ervan.

```
if digit3 < 8 then
    value := Power10((digit1 * 10 + digit2), digit3)
  else if digit3 = 8 then
    value := (digit1 * 10 + digit2) * 0.1
  else if digit3 = 9 then
    value := (digit1 * 10 + digit2) * 0.01;

result := FloatToStr(value) + ' Ohm';

if value > 1000 then
    result := FloatToStr((value / 1000)) + ' k Ohm';

if value > 1000000 then
    result := FloatToStr((value / 1000000)) + ' M Ohm';

LabelValue.Text := result;
```

de weergave worden geactualiseerd. Om het makkelijker te maken de besturingselementen van de interface in de broncode te identificeren, geven we de elementen unieke namen. Hiervoor passen we de eigenschap *Name* van elk element aan, bijvoorbeeld *LabelValue*, *RectangleRing1*, *ComboBoxRing1* enzovoort. We selecteren alle selectievakjes voor de kleuren van de ringen tegelijk en kiezen het event *OnChange*. Met een dubbelklik in de Object Inspector maken we een procedure voor dit event aan in de ontwikkelomgeving. Deze broncode wordt dus telkens aangeroepen wanneer we een andere kleur kiezen.

Eerst passen we de kleur van de rechthoeken aan aan de hand van de selectie van de ring. Dit gebeurt bijvoorbeeld met:

```
RectangleRing1.Fill.Color := TAlphaColors.Red
```

Dit zal de rechthoek in het rood tekenen. Voor het eerste selectievakje ziet dat eruit als in **listing 1**.
Voor de andere kleurringen wordt de selectie aangepast aan de eerder genoemde index (zie **tabel 1**). Dan wordt de weerstandswaarde berekend, op deze manier:

(waarde van positie 1 * 10 + waarde van positie 2) * 10^vermenigvuldigfactor

De berekening wordt gedaan in de eenheid ohm. Voor waarden groter dan 1.000 wordt een omrekening gemaakt naar kΩ en voor waarden > 1.000.000 naar MΩ. Voor de ringkleuren goud en zilver wordt vermenigvuldigd met 0,1 resp. 0,01. De waarde voor de tolerantie hoeft alleen uit de eerdere tabel (**listing 2**) te worden gelezen.

Nu is onze app functioneel: u kunt een willekeurige kleurencombinatie van de ringen kiezen en de juiste waarde van de weerstand wordt weergegeven. U kunt nu de app-packages voor Android en/of iOS aanmaken en de app uitrollen. Het is ook mogelijk om de app vanuit Delphi in Google Play of de Apple Store aan te bieden. U kunt de broncode voor het programmavoorbeeld downloaden van de projectpagina bij dit artikel [5].

Verdere mogelijkheden

In het voorbeeld hebben we al enkele mogelijkheden van app-ontwikkeling met Delphi laten zien. Maar er is nog veel meer mogelijk. Enkele voorbeelden:

- *Visuele componenten*: om aantrekkelijke interfaces te creëren, zijn er verschillende componenten (knoppen, tekstvelden, selectievakjes, tabbladen enzovoort) beschikbaar. U kunt deze besturingselementen plaatsen met behulp van de grafische designer en configureren via de eigenschappen. Het systeem creëert automatisch de juiste elementen voor het doelplatform (Android, iOS, Windows) wanneer u de applicatie compileert.
- *Niet-visuele componenten*: deze leveren belangrijke bouwstenen voor de programmering die steeds weer nodig zijn. Voorbeelden zijn een *Timer*-component en componenten om de toegang tot databases te vergemakkelijken.
- *Specifieke functionaliteiten van mobiele apps*: er zijn visuele en niet-visuele componenten beschikbaar waarmee typische taken bij het programmeren van mobiele apps sneller kunnen worden opgelost, bijvoorbeeld toegang tot de locatiegegevens, de camera van de smartphone of de afhandeling van de gegevensuitwisseling met servers (backends) in de cloud.

Deze ontwikkelmogelijkheden kunt u gebruiken als u een nuttige mobiele app wilt maken voor productief gebruik. De ontwikkelaar wordt vanuit de ontwikkelomgeving ook ondersteund bij het rechtstreeks genereren van de app-packages die nodig zijn voor de distributie van de apps via de Google of Apple Store.

Conclusie en vooruitzichten

Hoewel apps slechts toepassingen zijn voor die 'kleine' apparaatjes die u overal meeneemt, is het programmeren ervan bepaald geen 'kleine' klus. Met een geschikt tool kunt u werk besparen door tegelijkertijd te ontwikkelen voor Android en iOS vanuit een gemeenschappelijke broncode. Delphi biedt een intuïtieve aanpak, die veel elektronici misschien bekend voorkomt (uit Visual Basic, Windows Forms enzovoort). De gebruikersinterface kan rechtstreeks en vrij snel worden ontworpen in de grafische editor. Deze aanpak is ook nuttig als u een applicatie voor Windows of Linux moet schrijven. Op deze manier kunt u de broncode meerdere malen gebruiken. Met één klik kunt u omschakelen tussen diverse platformen. De ontwikkelomgeving genereert automatisch de benodigde app-packages. In het ideale geval hoeft u verder niets meer aan te passen.

200265-03

Vragen of opmerkingen?
Hebt u vragen of opmerkingen naar aanleiding van dit artikel? Stuur een e-mail naar de auteur of naar de redactie van Elektor, via redactie@elektor.com.

GERELATEERDE PRODUCTEN

- **Engelstalig boek "Android App Development for Electronics Designers"**
  www.elektor.nl/18687
- **Engelstalig boek "C# Programming for Windows and Android"**
  www.elektor.nl/17342

Figuur 12. De gestarte app (nog zonder functionaliteit) in de simulator (iOS)

Figuur 13. het tabblad Information.

Een bijdrage van
Auteur: **dr. Veikko Krypczyk**
Redactie: **Jens Nickel**
Vertaling: **Evelien Snel**
Layout: **Giel Dols**

WEBLINKS

[1] https://www.embarcadero.com/products/delphi/starter/free-download
[2] http://www.macincloud.com/
[3] http://docwiki.embarcadero.com/RADStudio/Rio/en/Android_Mobile_Application_Development
[4] http://docwiki.embarcadero.com/RADStudio/Rio/en/IOS_Mobile_Application_Development
[5] **Projectpagina bij dit artikel**: http://www.elektormagazine.nl/200265-03