**Belegarbeit**

**Internet-Technologien**

App-Entwicklung mit Xamarin in Visual Studio

**Vorgelegt am:** \_\_.\_\_.2023

**Von:** Elias Kunze, Pascal Köppel

**Matrikelnummern:** 4004399, 4004454

**Studiengang:** Technische Informatik

**Modul:** Internet-Technologien

**Modulcode:** 4TI-INT-40

**Seminargruppe:** TI21

**Gutachter:** Dr. Mathias Sporer (Staatl. Studienakademie Glauchau)

Inhaltsverzeichnis

[Inhaltsverzeichnis II](#_Toc140477078)

[1 Zielführung 3](#_Toc140477079)

[2 theoretische Grundlagen 3](#_Toc140477080)

[**2.1** **Cross-Plattform-App** 3](#_Toc140477081)

[**2.2** **Xamarin** 4](#_Toc140477082)

[3 praktische Grundlagen 5](#_Toc140477083)

[**3.1** **Einrichten von Visual Studio** 5](#_Toc140477084)

[**3.2** **Verwendung von Xamarin** 11](#_Toc140477085)

[**3.2.1** **Anlegen einer neuen Klasse** 11](#_Toc140477086)

[**3.2.2** **Zugriff auf Gerätehardware und -software** 11](#_Toc140477087)

[**3.3** **Veröffentlichung von Cross‑Plattform-Apps** 16](#_Toc140477088)

[4 Aufgaben 17](#_Toc140477089)

[**4.1** **geführte Aufgabe** 17](#_Toc140477090)

[**4.2** **selbstständige Aufgabe** 18](#_Toc140477091)

[Quellenverzeichnis 19](#_Toc140477092)

# **Zielführung**

Ziel der Belegarbeit ist eine Übungsanleitung zur Verwendung von Xamarin in Visual Studio. Diese Open-Source-Plattform ermöglicht das Entwickeln von Cross‑Plattform iOS-, Android- oder Windows-Applikationen. Die entsprechenden Anwendungen können betriebssystemunabhängig entwickelt werden und werden in nativen Paketen kompiliert. Ein geführtes Beispiel soll zu einem besseren Verständnis beitragen und das Lösen der eigenständigen Aufgabe ermöglichen.

# **theoretische Grundlagen**

## **Cross-Plattform-App**

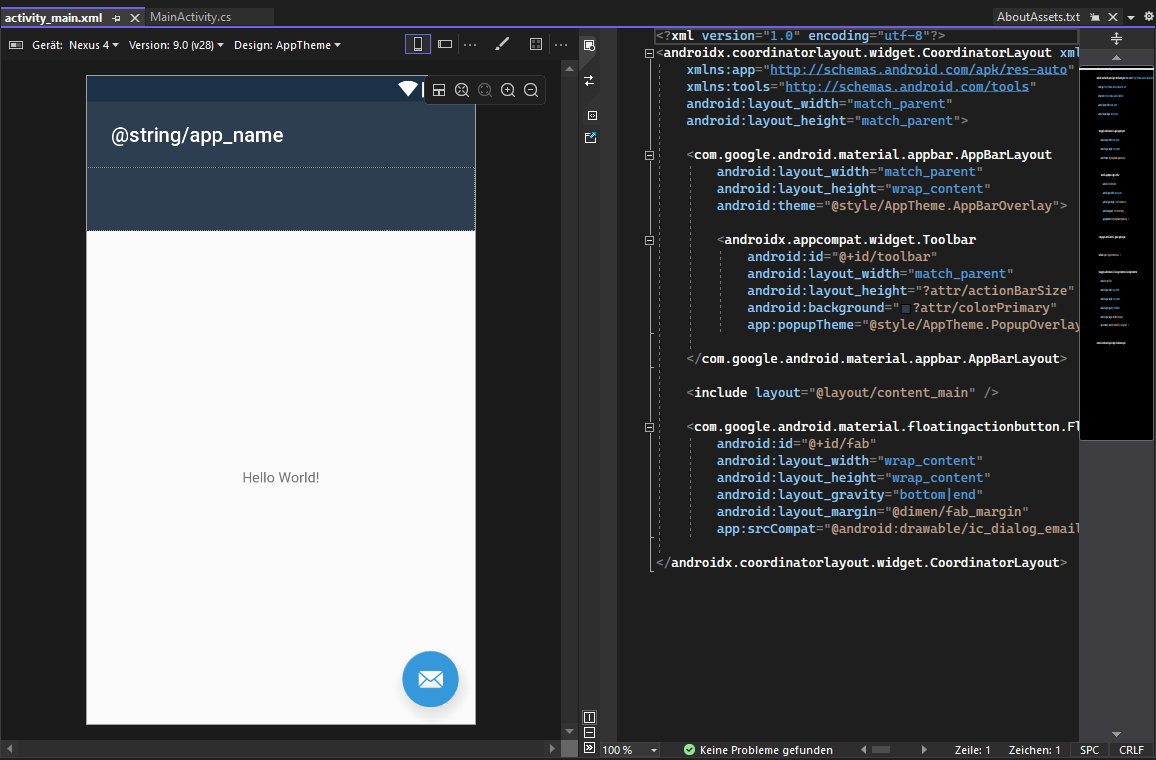
Da im Rahmen dieser Belegarbeit eine App entwickelt werden soll, welche für IOS und Android nutzbar ist, bietet sich eine Cross-Applikation (Cross-App) an. Es ist möglich aus einem einmalig geschriebenen Quelltextes mittels Framework in die entsprechenden Systemsprachen zu kompilieren. Hierdurch können bis zu 90% des Programmcodes unabhängig vom Betriebssystem entwickelt werden. Deshalb muss nur ein kleiner Teil individuell an das jeweilige Gerätesystem angepasst werden. Deshalb ist die Entwicklung einer Cross-Plattform-Applikation günstiger, zeitsparender und wartungsfreundlicher als die Entwicklung von unabhängigen Apps. Jedoch ist der Speicherbedarf durch die Kompilierung größer als der einer nativen Applikationen. Dies hat ebenfalls minimale Performance-Verluste zur Folge. Bei der Wahl dieser Entwicklungsvariante ist zu beachten, dass sich Betriebssystem-Updates auf die Funktionalität der Applikationen auswirken können. Dementsprechend werden auch regelmäßig Änderungen und Anpassungen des verwendeten Frameworks an das Betriebssystem vorgenommen.

Neben Cross-Plattform-Apps gibt es noch andere Methoden zur Entwicklung einer Applikation. Bei einer Web-Apps handelt es sich um eine App die über den Webserver geladen und im Webbrowser ausgeführt wird. Bei einer native App wird der Quellcode abhängig vom Betriebssystem entwickelt. Dies führt dazu, dass sie leistungsfähiger sind, weniger Speicherplatz benötigen sind und auch auf alle Features des Endgerätes zugreifen können. Das Mehrfache entwickeln für die einzelnen Plattformen erhöht die Komplexität und den Zeitaufwand des Projektes enorm. Eine Hybride-App ist eine Mischung aus Web-App und nativer App. Sie greift zum Teil auf das Betriebssystem des Endgerätes zu, aber nutz auch Web-Anwendungen.

## **Xamarin**

Für die Programmierung einer Cross-Plattform-Applikation ist eine Entwicklungsumgebung wie Visual Studio und ein entsprechendes Framework notwendig, um die App für mehrere Betriebssysteme zugänglich zu machen. Ein Framework ist ein Entwicklungsrahmen mit dessen Hilfe die Architektur der Software bestimmt beziehungsweise vorgegeben wird. Es besteht aus einer Runtime-Umgebung, mehreren mitgelieferten Bibliotheken und weiteren Komponenten. Um eine Cross-Plattform-Applikation in Visual Studio zu entwickeln, steht das Framework Xamarin Verfügung. Es unterstützt die Sprache C# und bietet SDKs für Android, Windows-Phone und IOS. Mit Software Development Kits (SDKs), welche vom Hersteller der Plattform oder des Betriebssystems bereitgestellt werden, können Anwendungen für spezifische Plattformen erstellt werden. Sie beinhalten einen Compiler, Debugger und unterschiedliche APIs für verschiedene Anwendungsfälle. Zusätzlich kann ein SDK-Werkzeuge zur Dokumentationen, einen Editor, mehrere Bibliotheken und Netzwerkprotokolle enthalten.

Xamarin.Forms ist ein Teil von Xamarin und hilft dem Entwickler bei der Erstellung der Nutzeroberfläche mithilfe der Beschreibungssprache XAML. Dadurch ist es möglich die Oberfläche einmalig zu entwerfen und automatisch eine Oberfläche für das entsprechende Zielbetriebssystems zu generieren. Durch die simplen Aufbau der Sprache XAML, welcher analog zu XML aufgebaut ist, ist der Quellcode leicht verständlich. Durch Xamarin.Forms kann der Entwickler von Cross‑Plattform‑Applikationen die Entwicklungszeit minimieren. Insgesamt ist bei Frameworks, wie Xamarin zu beachten, dass regelmäßig die Funktionalitäten der Anwendungen getestet werden muss. Aufgrund von Aktualisierungen der Betriebssysteme oder des Frameworks, kann es notwendig sein, dass der Zugriff auf Gerätekomponenten geändert wurde und diese im Quellcode angepasst werden müssen.

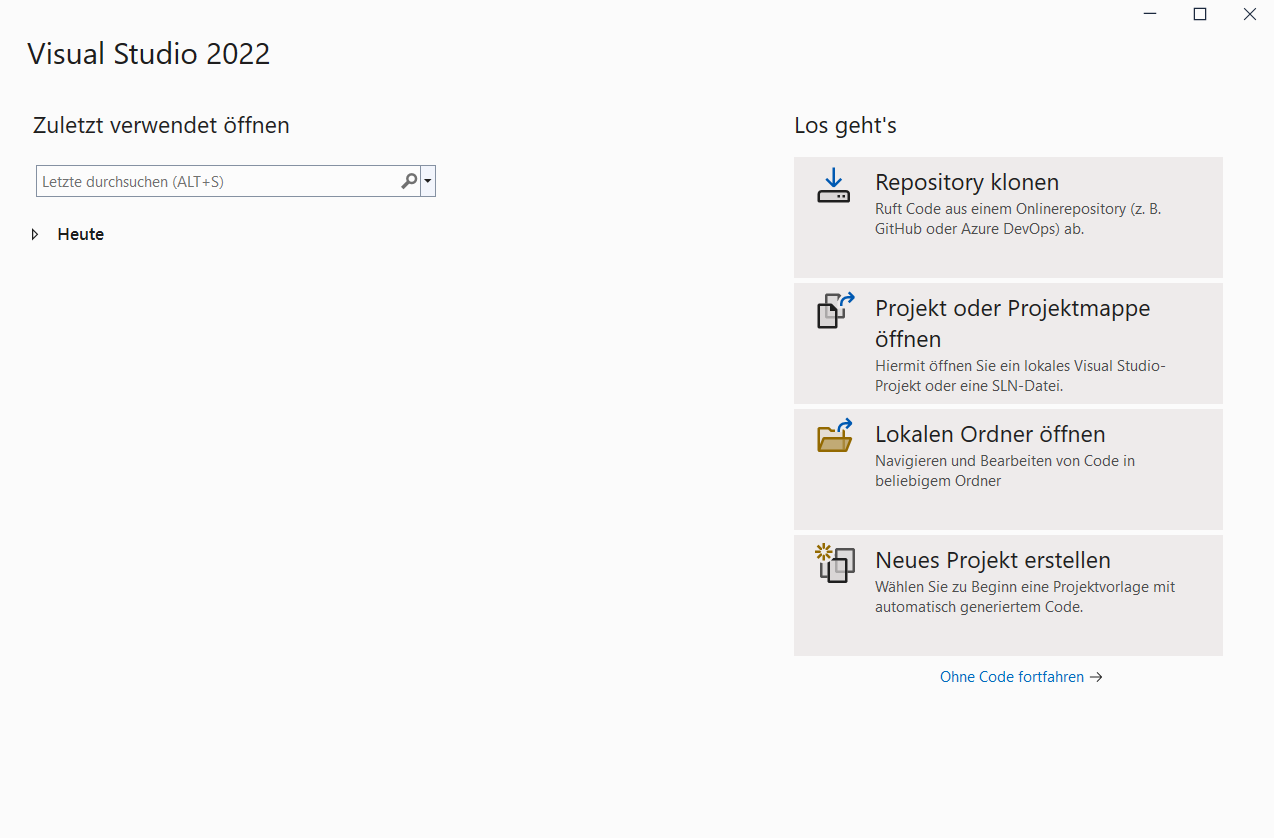


# **praktische Grundlagen**

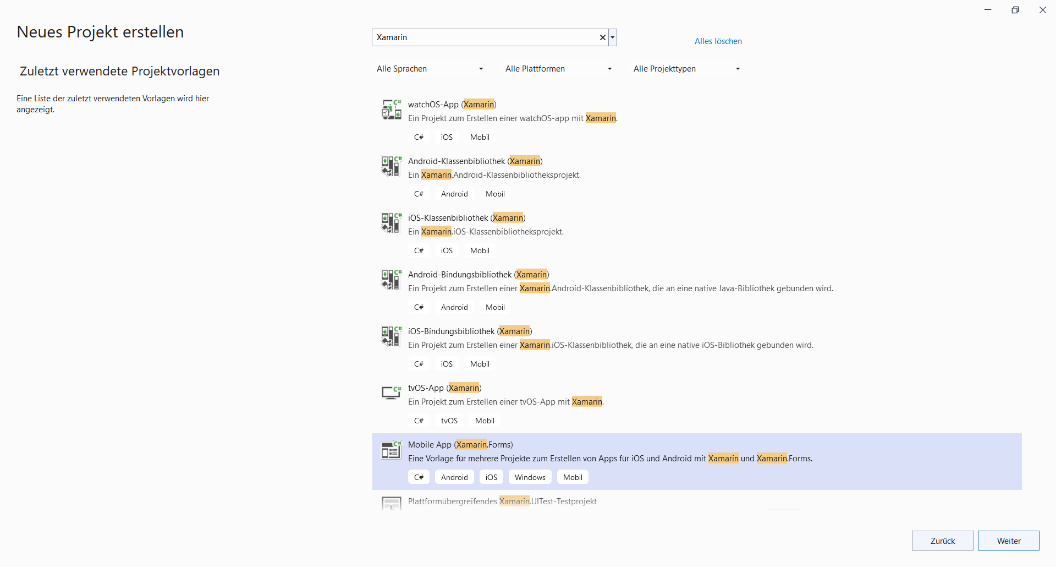
## **Einrichten von Visual Studio**

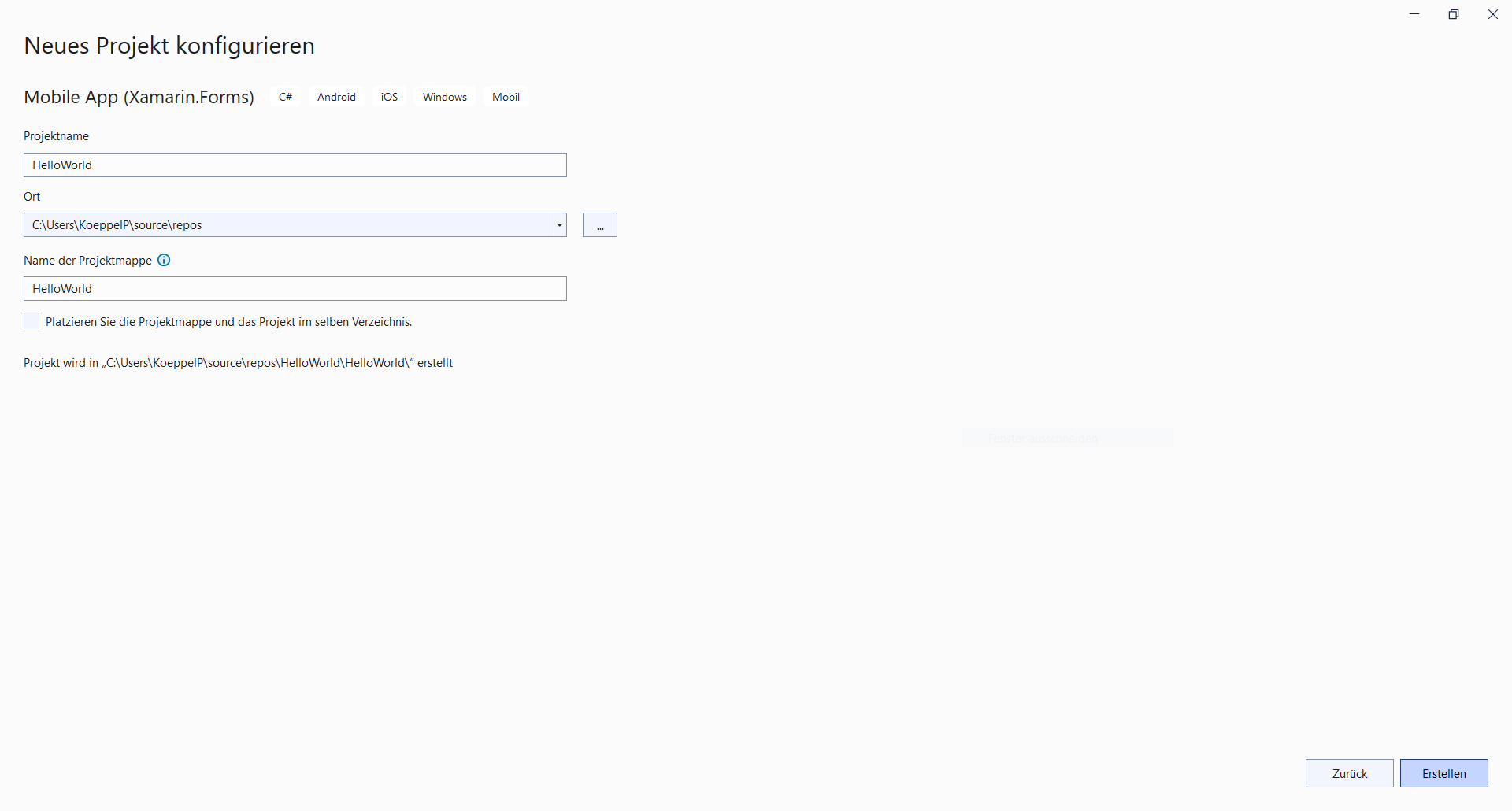
Um Xamarin für die Entwicklung einer Cross-Plattform‑App zu verwenden, ist die Installation des Workloads *.NET Multi‑Plattform App‑Benutzeroberflächenentwicklung* notwendig. Zusätzlich muss in den Installationsdetails als optionales Paket ausgewählt und heruntergeladen werden.

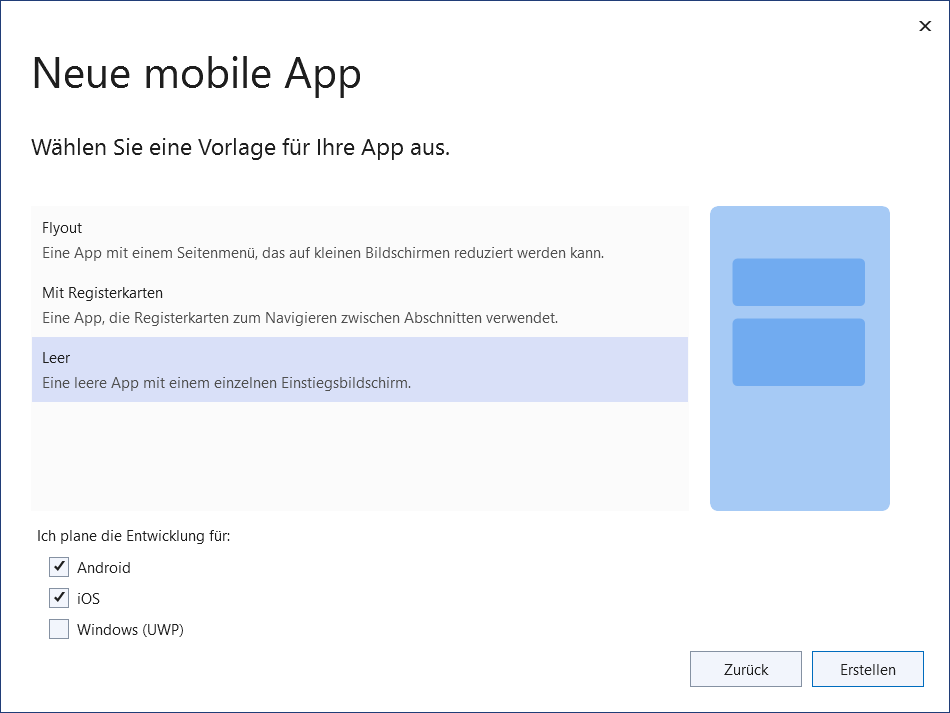
Nach Abschluss des Installationsvorgangs kann Visual Studio gestartet und ein erstes Startprojekt angelegt werden. Zur Erstellung ist *Neues Projekt erstellen* auszuwählen.

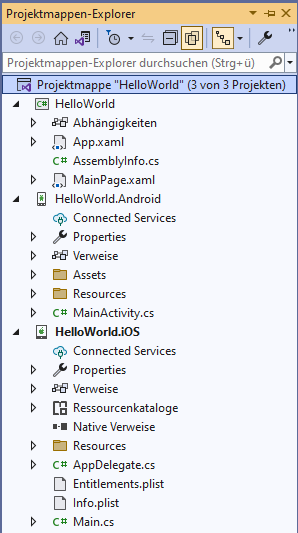


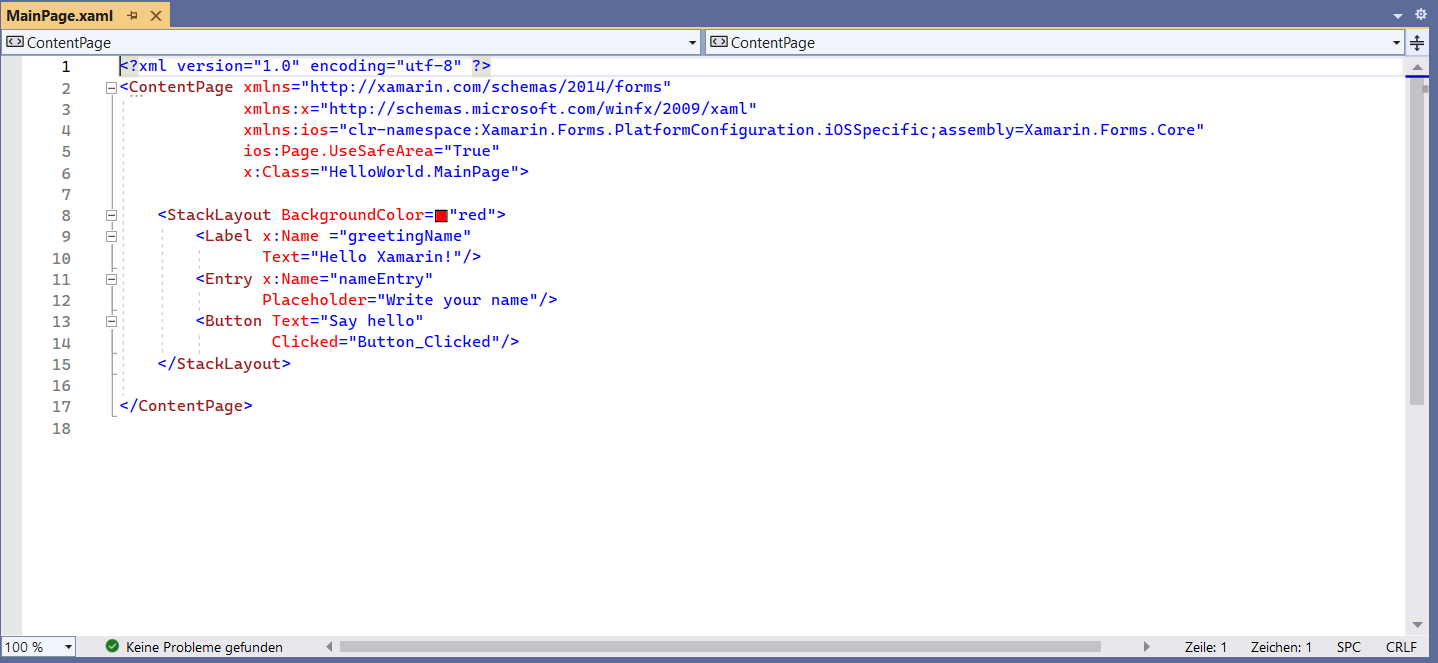
Um in die passende Vorlage zu finden, ist in die Suchleiste *Xamarin* einzugeben. Als Ergebnis werden verschiedene Entwürfe für Plattformen wie Android und iOS und unterschiedliche Geräte wie Handys, Uhren und TV-Geräte vorgeschlagen. Für die Entwicklung einer Cross‑Plattform‑App ist die Verwendung von *Mobile App (Xamarin.Forms)* passend.

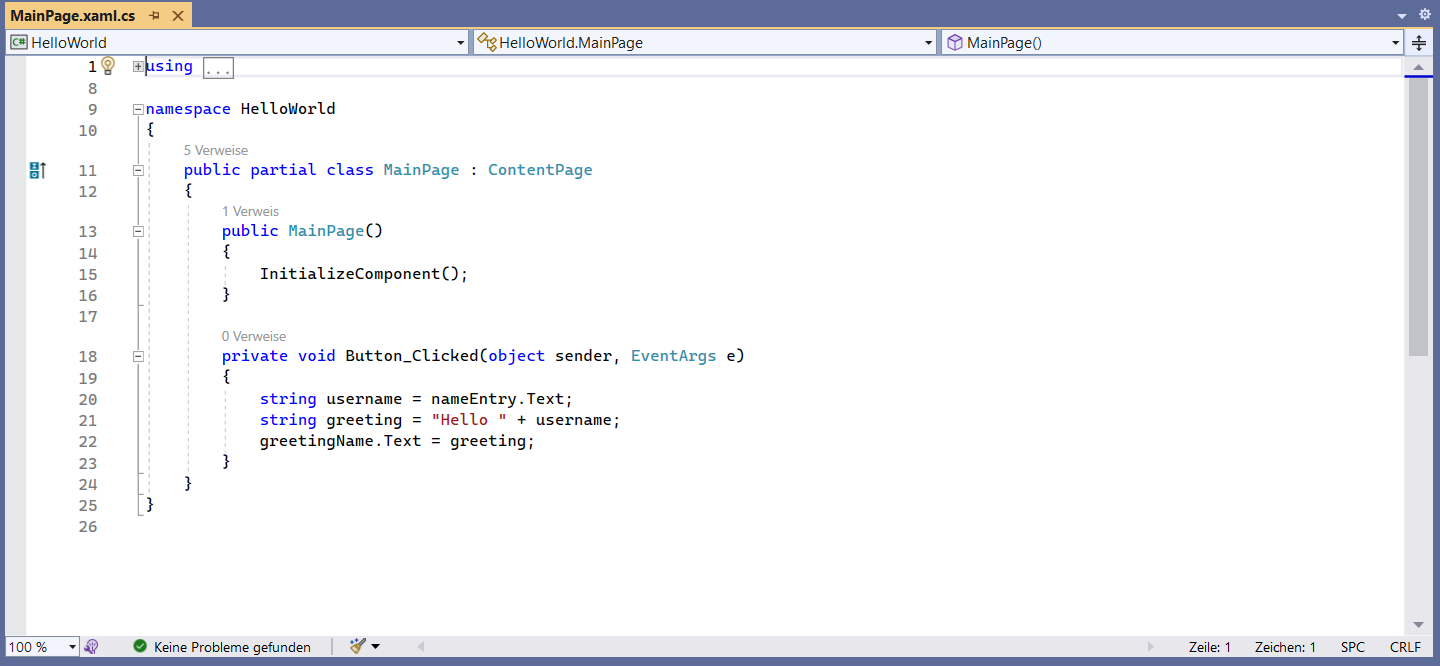


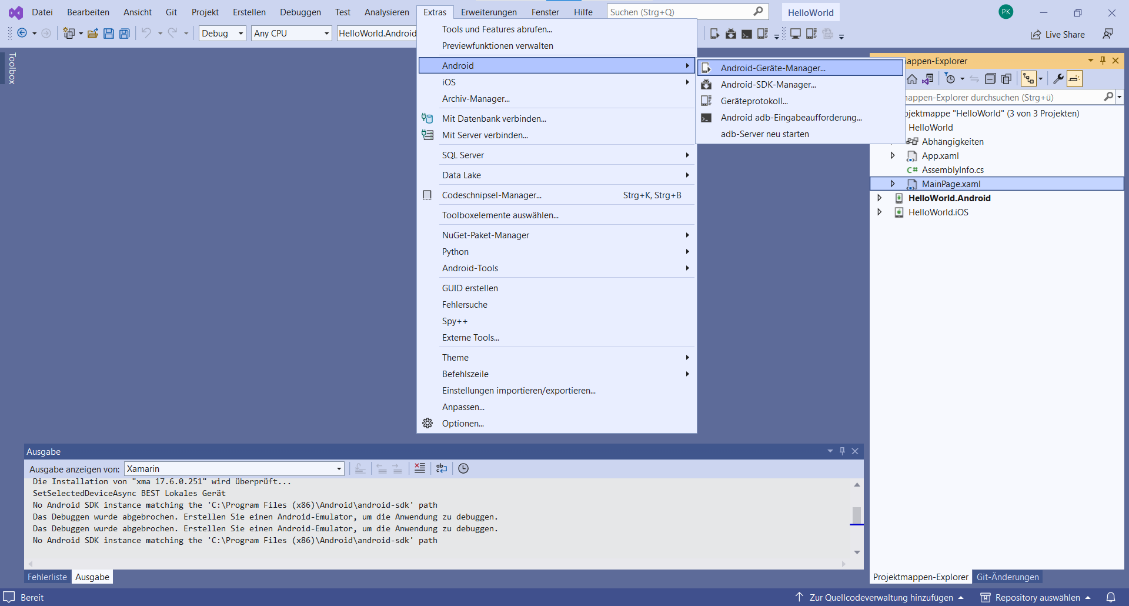
Als simpler Projektname für die Starter‑App ist *HelloWorld* angebracht. Im nächsten Schritt ist eine Vorlage für die Anwendung auszuwählen. In diesem Testfall ist *Leer* zu wählen, um das Projekt so einfach wie möglich zu erstellen. Da eine Android und iOS‑Applikation zu erstellen ist, sind diese beiden Fälle in den unteren Kästchen anzuhaken.

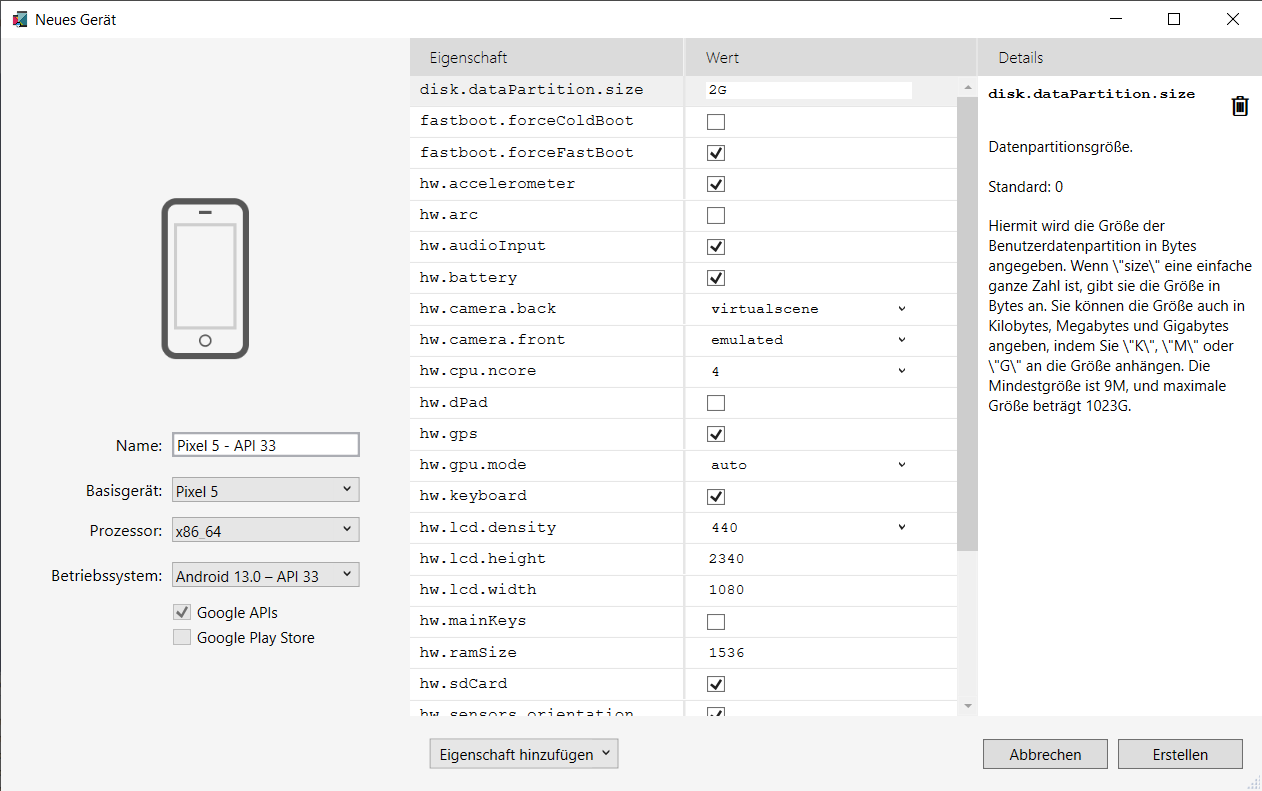


Die erstellte Projektmappe besteht aus drei einzelnen Projekten. In dem Projekt ohne iOS oder Android Erweiterung, werden alle Funktionalitäten programmiert und das Nutzerinterface erstellt. In diesem Projekt wird der Großteil der Anwendung implementiert. Der Aufbau Projekte ist identisch zu dem Aufbau bei der expliziten Appentwicklung. Der einzige Unterschied ist, dass ausschließlich C# Dateien zum Einsatz kommen. Innerhalb des Hauptprojekt ist die Klasse *App* zu finden, welche die Dateien *App.xaml* und *App.xaml.cs* enthält. *App.xaml.cs* enthält die gesamte Funktionalität des Projektes. In ihrem Konstruktor wird das Programm erstellt. Dieser Konstruktor wird in den Hauptklassen der jeweiligen Plattformprojekte aufgerufen. Der Aufruf ist möglich, da das Hauptprojekt in den Verweisen referenziert wurde. Der Programmcode des Konstruktors erzeugt ein neues Objekt der Klasse *MainPage*, welche ebenfalls in die Dateien *MainPage.xaml* für die Nutzeroberfläche und *MainPage.xaml.cs* für die Logik unterteilt ist.

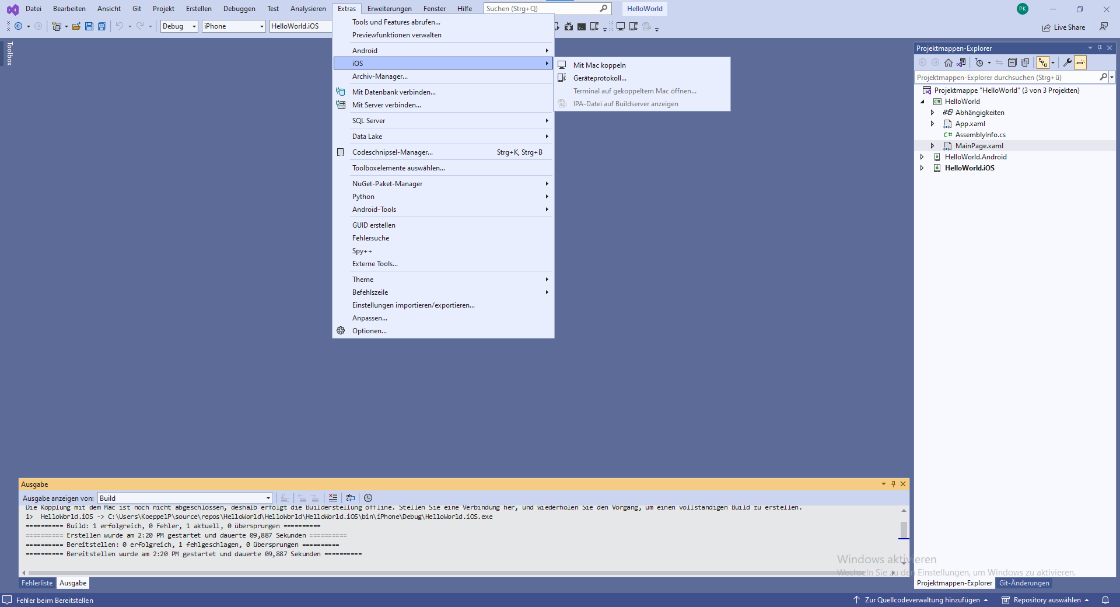
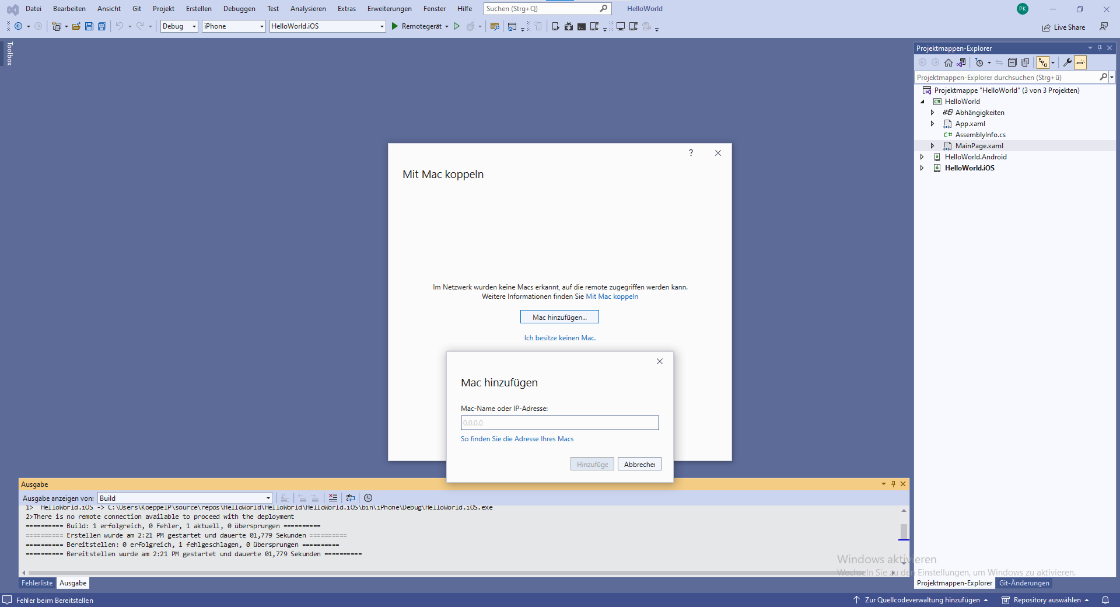
Für ein einfaches *HelloWorld*-Programm wird der Quelltext der *MainPage*-Klasse verändert. Es muss der Bereich innerhalb des *<StackLayout></StackLayout>* geleert werden. Ziel ist es durch das Betätigen eines Buttons den in ein Textfeld eingegebene Namen auf einem Label auszugeben. Um die benötigten Elemente einzufügen kann unter Ansicht die Toolbox verwendet werden. Des Weiteren muss eine SafeArea für iOS angelegt werden. Um die Größe der Applikation anzupassen. Zu beachten ist, dass die Bestandteile die folgenden Eigenschaften enthalten:

Durch das Einfügen der *Clicked*-Eigenschaft wird innerhalb der *MainPage.xaml.cs* Datei eine Methode angelegt, welche beim Aufruf des Klickereignisses durchlaufen wird. Innerhalb dieser Klasse wird der folgende Quelltext verwendet, um den eigetragenen Namen zu holen, zu modifizieren und erneut auszugeben.

Bei der ersten Ausführung des Quelltextes müssen für iOS und Android einige Vorbereitungen getroffen werden. Um das Android-Projekt zu testen, muss im *Android Device Manager* ein neuer Emulator angelegt werden. Dieser kann über **Extras -> Android -> Android-Geräte-Manager** geöffnet werden.

In dem geöffneten Fenster kann nun ein neues Gerät erstellt werden. Der Manager ermöglicht das Arbeiten mit verschiedenen Geräten mit unterschiedlichen Prozessoren und Betriebssystemen. Der Nutzer hat die Möglichkeit das Smartphone nach seinen eigenen Vorstellungen anzupassen. Zu testzwecken sind die Standarteinstellungen des Pixel 5 mit einem x86\_64 Prozessor und Android 13.0 ausreichend. Nach dem Erstellen dieses Emulators ist es möglich die App darauf zu starten.

Das Testen des iOS-Projektes ist über zwei Wege möglich. Für die erste Option mit einem Remotegerät ist ein Mac notwendig. Dieser wird mittels IP-Adresse im eigenen Netzwerk mit Visual Studio verbunden. Dies ist über **Extras -> Optionen… -> Xamarin -> iOS-Einstellungen** möglich. Optional kann der Nutzer unter *Simulator remote auf Windows* auswählen, ob das Testfenster auf dem Mac oder in Visual Studio zu sehen ist. Dann kann beim Start des Tests eines der vorgegebenen Geräte ausgewählt werden.



Option zwei ist das Testen über ein lokales Gerät. Voraussetzungen für diese Variante sind ein iPhone, iTunes auf dem PC und einen Apple Developer Account, welcher im Jahr 99 € kostet. Nach der Anmeldung mit der Apple ID, muss das Smartphone über USB mit dem PC verbunden werden und kann dann als Testgerät verwendet werden.

## **Verwendung von Xamarin**

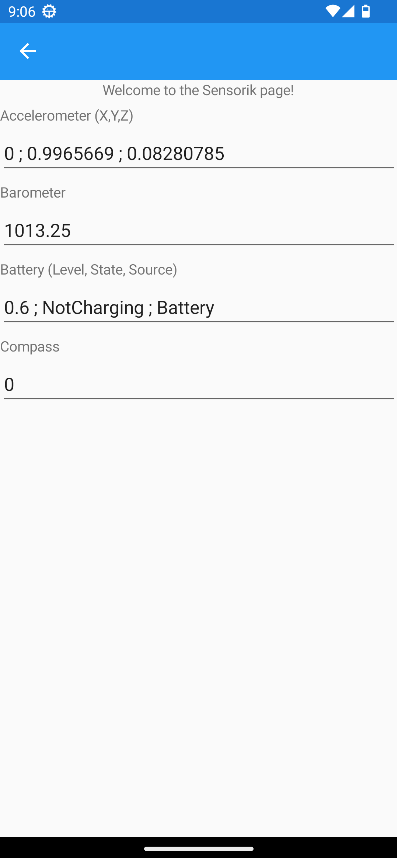
### **Anlegen einer neuen Klasse**

Xamarin.Forms besitzt bereits mehrere Vorlagen zur Erstellung von neuen Dateien. Es gilt die dabei die Unterscheidung zwischen zwei verschiedenen Vorlagetypen. Bei den Elementen ohne (C#) werden die XAML-Datei für das Format und die C#-Datei für die Logik erstellt. Bei den anderen Vorlagen wird ausschließlich die C#-Datei erstellt. Des Weiteren Unterscheidet Xamarin zwischen Inhaltsseite, Listenansichtsseite und einer Registerkartenseite. Die Inhaltsseite ist eine Standardseite ohne eine besondere Formatierung, in welcher der Inhalt simpel wieder gegeben wird. Bei einer Listenansichtsseite handelt es sich um eine Ansicht, welche eine ListView zum Anzeigen von verschiedenen Daten beinhaltet. Die Registerkartenseiten dienen zur Navigation zwischen verschiedenen Seiten. Sie besitzen einen Navigationskopf und enthalten untergeordnete Seiten, durch welche navigiert werden kann.

### **Zugriff auf Gerätehardware und -software**

Um auf dem Endgerät mit anderen Applikationen und Sensoren zu kommunizieren, existieren eine Vielzahl von verschiedenen NuGet-Paketen. Diese Bibliotheken enthalten Klassen, welche ein bestimmtes Problem lösen und dem Entwickler die Arbeit erleichtern. Sie sind in den meisten Fällen kostenlos und können einfach über **Projekt -> NuGet-Pakete verwalten…** im jeweiligen Projekt installiert werden. Eine wichtige Bibliothek für den Zugriff auf externe Daten ist Xamarin.Essentials. Dieses Paket von Microsoft enthält mehrere essenzielle APIs für die Applikation und wird regelmäßig angepasst und aktualisiert. Das Paket wird bereits bei der Erstellung des Programms installiert. Es sollte aber geprüft werden, ob bereits die neuste Version mitgeliefert wurde oder gegebenenfalls eine aktuellere Variante nachinstalliert werden muss. Den genauen Einsatz dieses Paketes soll an einigen Beispielen demonstriert werden. Für einige dieser Beispiele müssen separat Zugriffsrechte erteilt werden. Für Android werden diese unter **HelloWorldProject.Android ‑> Properties ‑> AndroidManifest.xml** festgelegt. Für das iOS-Projekt muss die Datei **Info.plist** überarbeitet werden. Der Programmcode für die nachfolgenden Beispiele ist im Anhang … aufgelistet.

Im ersten Teil soll der Zugriff auf einige Sensoren und Aktoren des Gerätes beschrieben werden. Hierfür wird ein neues Inhalts-Element mit der Bezeichnung *SensorPage.xaml* innerhalb des HelloWorldProject hinzugefügt. Um einen Zugriff auf diese Klasse zu haben, wird in der Klasse *MainPage.xaml* zu dieser per Button-Druck navigiert. Um diese Navigierung zur ermöglichen muss in der Hauptdatei *App.xaml.cs* der Zugriff auf die MainPage-Klasse abgeändert werden. Anstatt mit *MainPage = new MainPage()* diese direkt zu erzeugen, soll sie nun über eine NavigationPage welche als Wurzel das die neu erzeugte MainPage enthält, erzeugt werden. Im Quellcode findet dieser Zugriff wie folgt statt: *MainPage = new NavigationPage(new MainPage())*

Die Oberfläche der neu erzeugten Klasse SensorPage beinhaltet ausschließlich lesbare Textfelder, welche die Werte der Sensoren ausgeben und Beschriftungen für diese Textfelder.

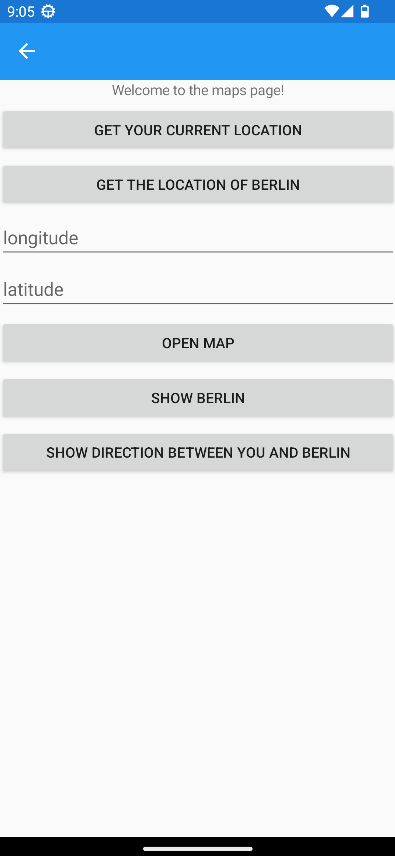
In der C#-Datei erfolgt der Zugriff auf die einzelnen Sensoren und Aktoren. In diesem Beispiel wird auf den Beschleunigungsmesser, das Barometer, die Batterie, den Kompass und den Erschütterungssensor zugegriffen. Als Aktor wird eine Vibration ausgeführt. Der Programmablauf für die einzelnen Sensoren ist analog. Es wird abgefragt, ob sich die Werte aktualisiert haben und dann die Inhalte der Textfelder aktualisiert. Die Vibration findet zwei Sekunden lang statt, wenn das Gerät geschüttelt wird. Für alle diese Zugriffe müssen die folgenden Berechtigungen für iOS und Android vergeben werden.

Ein weiteres Beispiel ist die Verwendung der Kamera. Auch hierfür wird eine neue Klasse mit dem Namen *CameraPage.xaml* erstellt. Diese lässt sich erneut von einem Button auf der MainPage aufrufen. Die Seite besteht aus vier Elementen. Einem Label mit einer kurzen Willkommensnachricht, einem Button zum Öffnen der Galerie, um ein Bild auszuwählen, ein Button zum Öffnen der Kamera, um ein Bild aufzunehmen und einem Image, um das jeweilige Bild anzuzeigen.

Die Zugriffe auf Kamera und Galerie erfolgen über die MediaPicker Klasse von Xamarin.Forms in einer asynchronen Methode. Dies ist notwendig, um bei Aufruf der Klasse mit await den aktuellen Thread nicht zu blockieren und die Task weiter auszuführen. Auch für dieses Beispiel müssen die folgenden Berechtigungen vergeben werden.



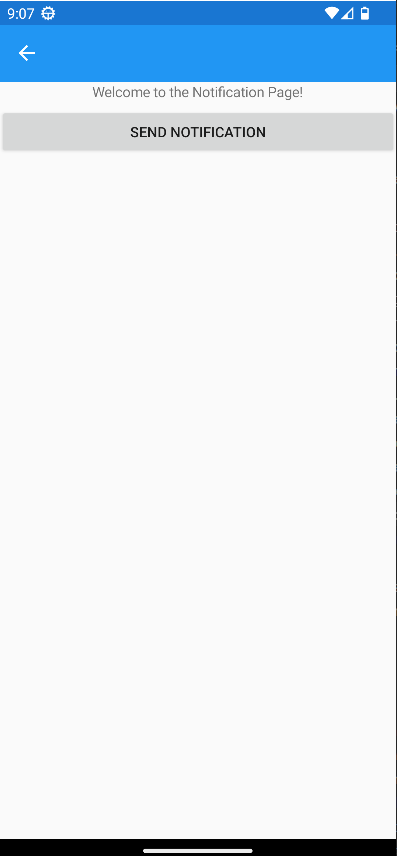
Ein weiteres Beispiel beschäftigt sich mit dem Umgang mit Karten in Xamarin.Forms in der neu erzeugten Klasse *MapsPage.xaml*. Die Seite besteht aus einem Willkommenslabel, zwei Textfeldern zur Ausgabe und Eingabe des gewählten Länge‑ und Breitengrades, einem Button, der die aktuelle Position ermittelt, einem Button, der die Koordinaten von Berlin anzeigt und mehreren Buttons, welche die ausgewählte Position auf der Karte anzeigen, Berlin auf der Karte anzeigen oder die Route zwischen dem aktuellen Standort und Berlin ausgibt.



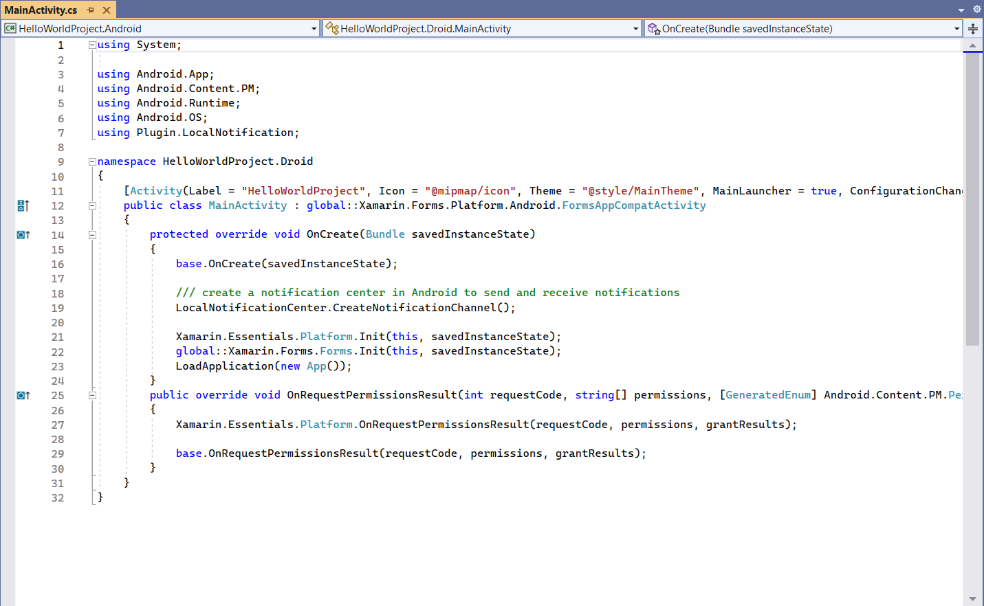
Die Karte wird in allen Fällen über die Funktion *OpenMapsAsync()* geöffnet. Der Quellcode zum Erzeugen und Ausgeben der Positionen ist im Anhang angeführt und genauer beschrieben. Auch für diese Funktionen müssen die folgenden Berechtigungen vergeben werden.



Xamarin Essentials deckt einen großen Teil der Funktionalitäten ab. Es ist beispielsweise mit diesem Paket auch möglich, eine SMS zu versenden, Text in Sprache auszugeben, Zugriff auf Kontakte und Zwischenablage zu haben und weitere Funktionen. Mit diesem Paket, ist allerdings nicht alles möglich. Beispielsweise kann mit Xamarin Essentials keine Datenbank erstellt und auch nicht darauf zugegriffen werden. Hierfür müssen in NuGet andere Pakete wie sqlite-net-pcl nachinstalliert und verwendet werden. Weiterhin gibt es ebenfalls keine Funktion zur Ausgabe einer lokalen Benachteiligung. Hierfür kann zum Beispiel das Plugin.LocalNotfication installiert werden.

Mit Hilfe dieses Paketes soll in der Klasse *NotificationPage.xaml* eine Benachrichtigung erstellt und ausgegeben werden. Bevor aber das Paket installiert und eingebunden werden kann, muss die Android‑Mindestversion auf Android 10.0 oder höher, gestellt werden. Dazu wird in die Eigenschaften des *HalloWorldProject.Android* Projekts navigiert und unter Android-Manifest die Einstellungen verändert. Um eine Benachrichtigung zu erhalten, wird ein Button angelegt, welcher dieses Ereignis ausführen soll.

Die Methode zum Senden der Benachrichtigung enthält eine Variable *notification*. Diese Variable ist vom Typ *NotificationRequest* und enthält Parameter wie den Titel, eine Beschreibung und eine Identifikation und wird mit *LocalNotificationCenter.Current.Show aufgerufen*. Zusätzlich können Events angelegt werden. Ein Event ist das Anzeigen eines Alerts, wenn eine Benachrichtigung eintrifft. Diese wird auf dem Main Thread der Applikation ausgeführt. Um diese Seite unter Android nutzen zu können, muss die Hauptdatei *MainActivity.cs* modifiziert werden. Es muss auch hier das Paket *LocalNotification* eingebunden und ein neuer Channel angelegt werden.



Die vollständige Beschreibung von Xamarin.Essentials ist zu finden unter https://learn.microsoft.com/de-de/xamarin/essentials/. Sollten andere Funktionen benötigt werden, die noch nicht im Projekt eingebunden sind, empfiehlt es sich die Suchfunktion von NuGet zu nutzen. An dieser Stelle findet man als Entwickler auch zahlreiche Informationen zu den Versionen, den Voraussetzungen, dem Paket- Entwicklern und eine Beschreibung, die in den meisten Fällen einen Link zu einer ausführlichen Dokumentation enthält.

## **Veröffentlichung von Cross‑Plattform-Apps**

Um eine Cross-Plattform‑Applikation einem Nutzer zur Verfügung zu stellen, muss sie in den entsprechenden Stores, wie der Apple AppStore und der Android Google Play Store, veröffentlicht werden. Bevor eine App herausgegeben werden kann, müssen mehrere Schritte beachtet und evaluiert werden.

Ein wichtiger Aspekt sind die Kosten, welche berücksichtigt werden müssen. Bevor eine Applikation veröffentlicht wird, müssen mehrere Kostenquellen beachtet werden. Für das Speichern und für den Zugriff auf Daten wird meist eine Datenbank benötigt. Die Kosten für ein Datenbanksystem sind abhängig von der Anzahl der Nutzer, die viel Speicherplatzbedarf besteht oder der Anzahl von Abfragen in einem bestimmten Zeitraum. Ebenfalls müssen eventuelle Kosten von verwendeten APIs beachtet werden. Eine Applikation läuft meist auf einem Serversystem. Deshalb müssen auch die Miet- beziehungsweise Beschaffungskosten für einen oder mehrere Server einkalkuliert werden. Die sind variable Kosten und sind bei jeder Anwendung unterschiedlich.

Wenn alle verfügbaren Angebote verglichen wurden und die Applikation fertig entwickelt ist, müssen die Kosten für das Hochladen eingerechnet werden. Diese sind von der Plattform abhängig. Um eine iOS-Applikation hochzuladen, ist eine Mitgliedschaft im Apple Developer Programm notwendig. Die Kosten für eine Lizenz belaufen sich auf 99€ pro Jahr. Zusätzlich muss der Entwickler im ersten Jahr 30% und im Anschluss 15% der App-Einnahmen an Apple abführen. Für das Veröffentlichen einer Applikation im PlayStore ist ein Google Entwicklerkonto notwendig. Neben diesen Kosten von 25€ im Jahr, muss eine Provision von 30% der Einnahmen im ersten Jahr und 15% in den Folgejahren gezahlt werden.

Plattformunabhängigen ist es verpflichtend für den Entwickler sich an bestehende Gemeinschaftsrichtlinien, Sicherheitsrichtlinien und der Regelungen der DSGVO zu halten.

Sind die angegeben Punkte beachtet und eingehalten, kann mit der hochladen begonnen werden. Bei Apple ist dies mit App-Store Connect möglich. Bevor Apple die App nach interner Prüfung freigibt, werden noch Bildschirmfotos, ein Titel und eine Beschreibung hinzufügt. Bei Google wird die Applikation per APK-Datei hochgeladen. Es ist möglich Bildschirmfotos, Titel, eine Beschreibung, eine Vorschau und eine Kategorie einzufügen. Nach der Prüfung durch Google, wird die App im Anschluss im Play Store freigegeben.

# **Aufgaben**

## **geführte Aufgabe**

## **selbstständige Aufgabe**

# **Quellenverzeichnis**