**Belegarbeit**

**Internet-Technologien**

Native 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](#_Toc137476854)

[1 Versuchsziel 3](#_Toc137476855)

[2 theoretische Grundlagen 3](#_Toc137476856)

[2.1 Was ist native App-Entwicklung 3](#_Toc137476857)

[2.2 Xamarin 4](#_Toc137476858)

[3 praktische Grundlagen 5](#_Toc137476859)

[3.1 Einrichten von Visual Studio 5](#_Toc137476860)

[3.2 Verwendung von Xamarin 10](#_Toc137476861)

[3.3 Veröffentlichung von nativen Apps 11](#_Toc137476862)

[4 Aufgaben 12](#_Toc137476863)

[4.1 geführte Aufgabe 12](#_Toc137476864)

[4.2 selbstständige Aufgabe 13](#_Toc137476865)

[Quellenverzeichnis 14](#_Toc137476866)

# Zielführung

Ziel der Belegarbeit ist eine Übungsanleitung zur Verwendung von Xamarin in Visual Studio. Diese Open-Source-Plattform ermöglicht das Entwickeln von nativen 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

## Was ist native App-Entwicklung

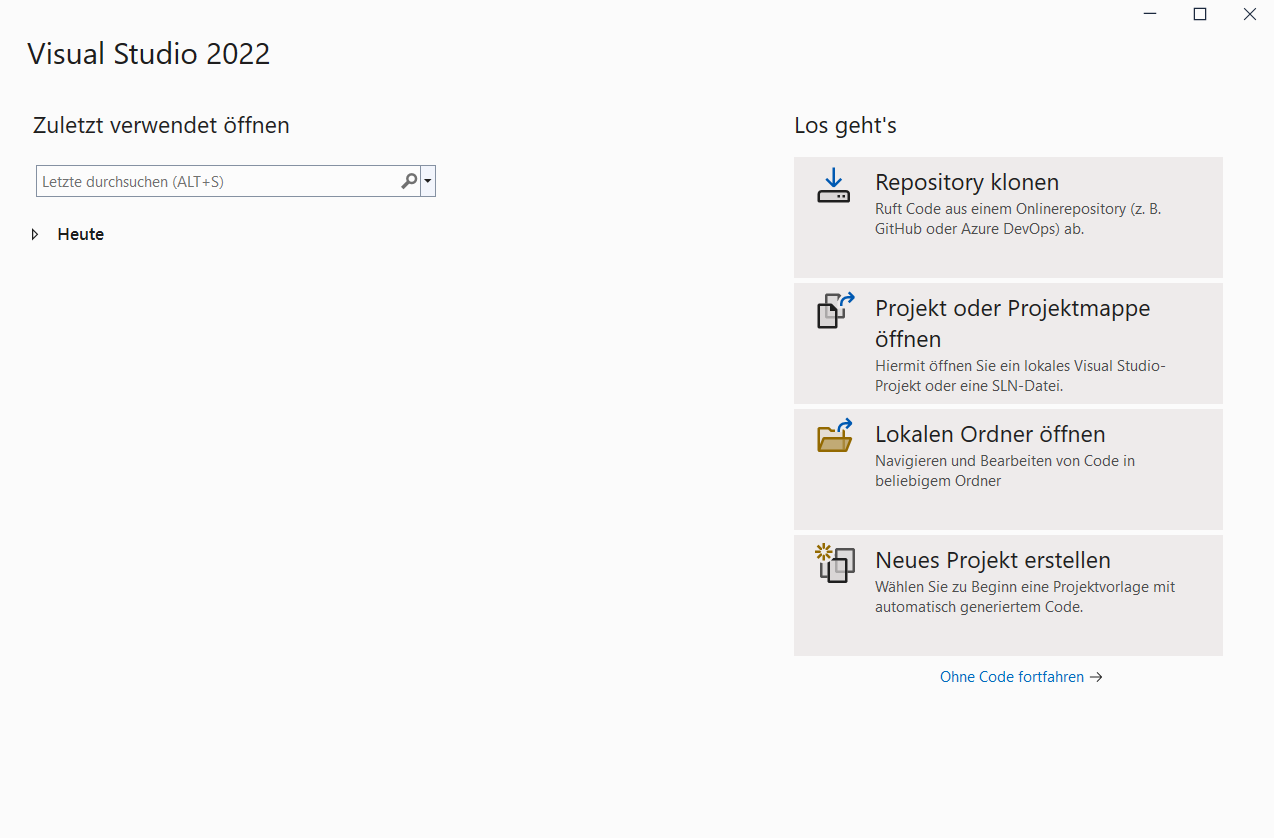
## Xamarin

# praktische Grundlagen

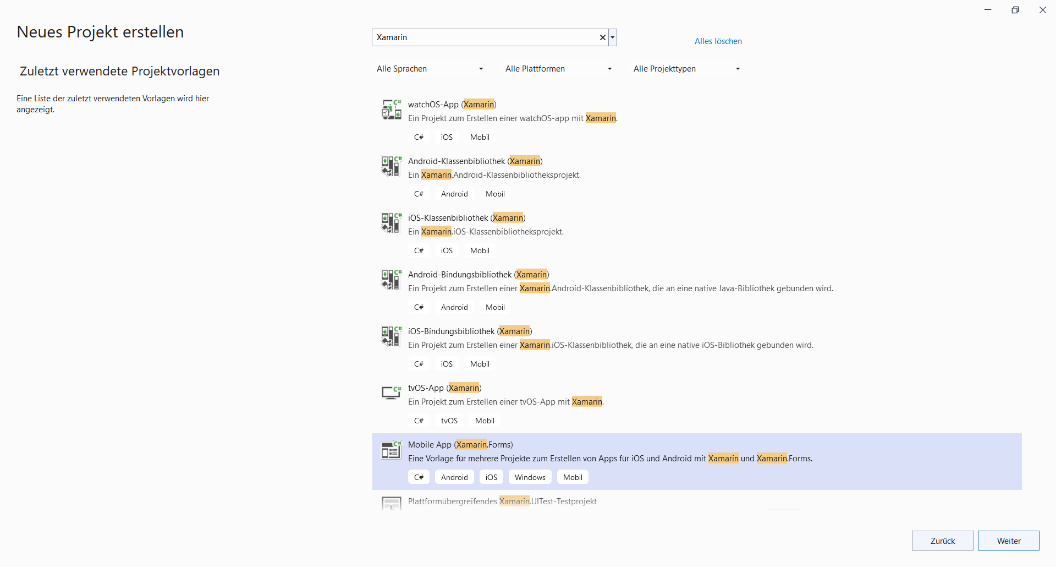
## Einrichten von Visual Studio

Um Xamarin für die Entwicklung einer nativen 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.

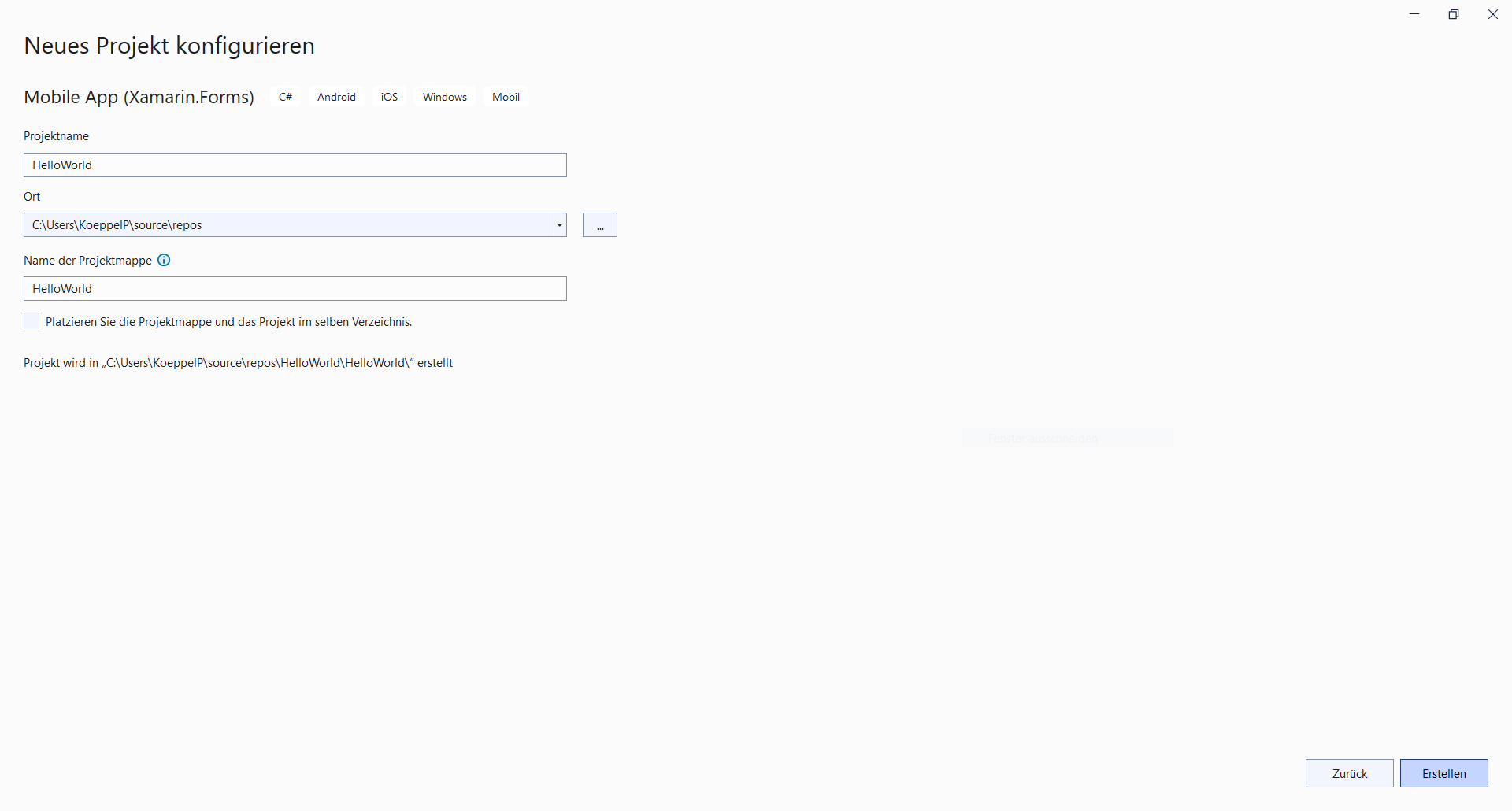
Nach Abschluss des Installationsvorgangs kann Visual Studio gestartet und ein erstes Startprojekt angelegt werden. Zur Erstellung ist *Neues Projekt erstellen* auszuwählen.

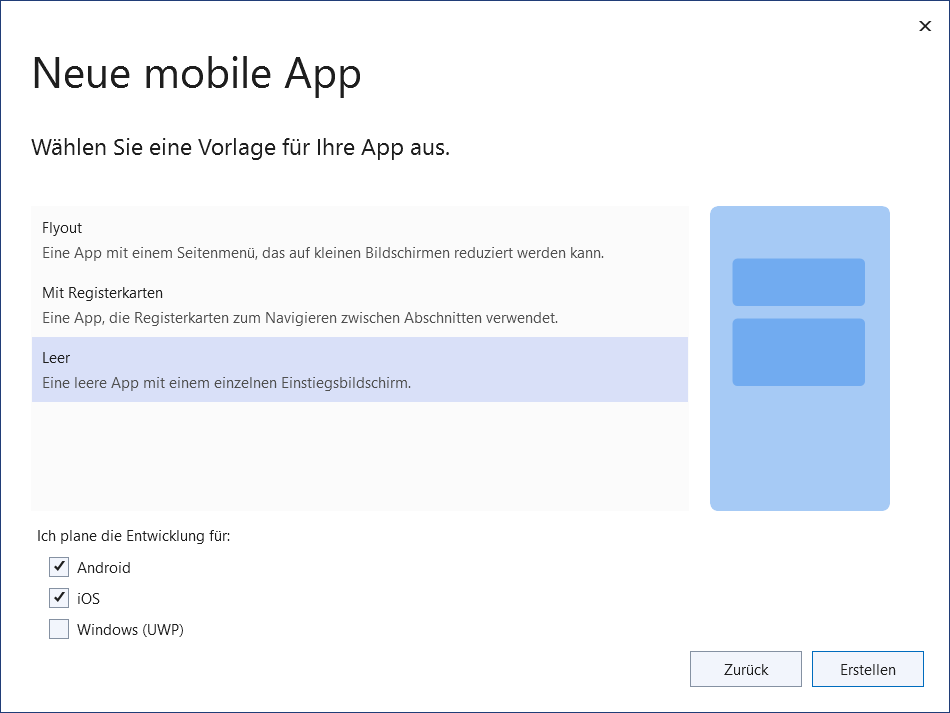


Um in die passende Vorlage zu finden, ist in die Suchleiste *Xamarin* einzugeben. Als Ergebnis werden verschiedene Entwürfe für Plattformen wie Android und iOS und unterschiedliche Geräte wie Handys, Uhren und TV-Geräte vorgeschlagen. Für die Entwicklung einer nativen App ist die Verwendung von *Mobile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

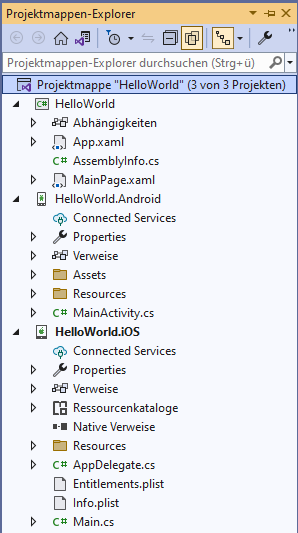


Als simpler Projektname für die Starter‑App ist *HelloWorld* angebracht. Im nächsten Schritt ist eine Vorlage für die Anwendung auszuwählen. In diesem Testfall ist *Leer* zu wählen, um das Projekt so einfach wie möglich zu erstellen. Da eine Android und iOS‑Applikation zu erstellen ist, sind diese beiden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

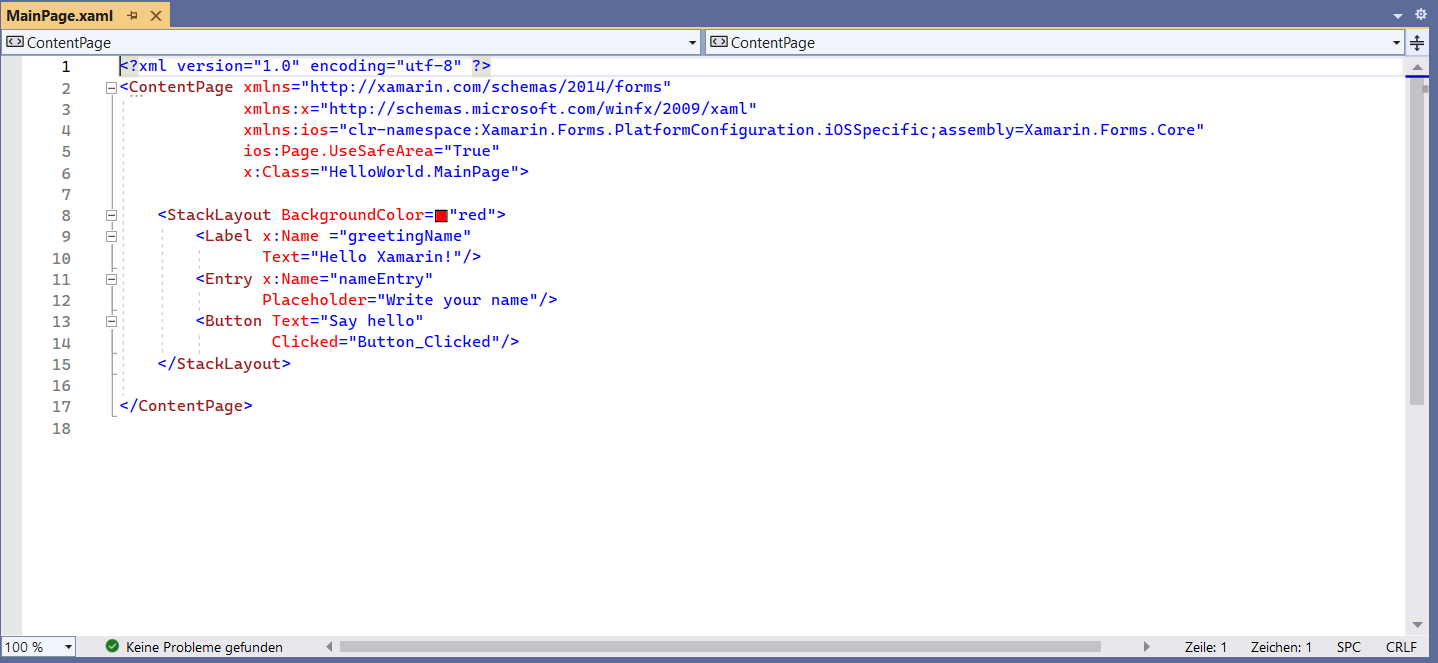




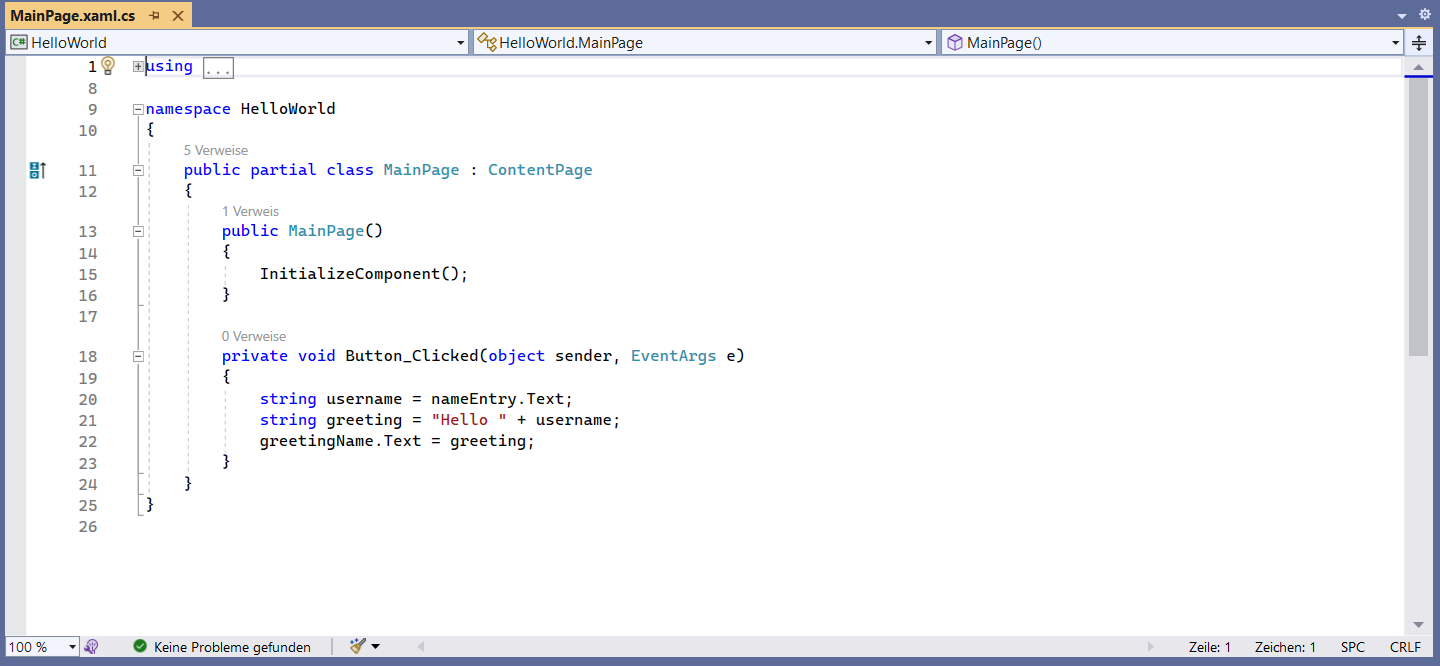
Die erstellte Projektmappe besteht aus drei einzelnen Projekten. In dem Projekt ohne iOS oder Android Erweiterung, werden alle Funktionalitäten programmiert und das Nutzerinterface erstellt. In diesem Projekt wird der Großteil der Anwendung implementiert. Der Aufbau Projekte ist identisch zu dem Aufbau 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.



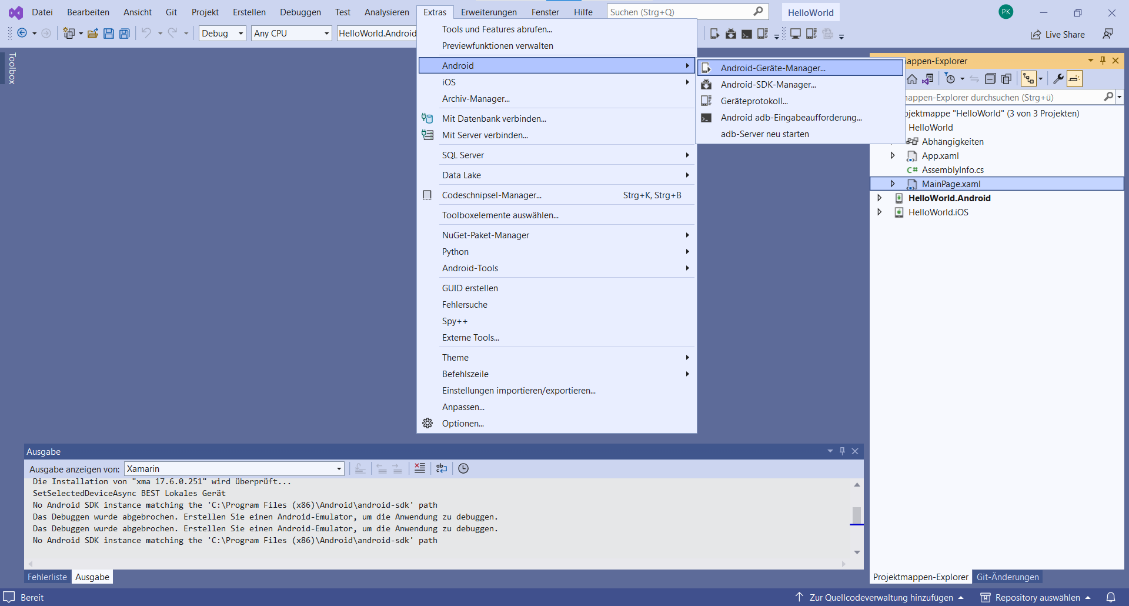
Für ein einfaches *HelloWorld*-Programm wird der Quelltext der *MainPage*-Klasse verändert. Es muss der Bereich innerhalb des *<StackLayout></StackLayout>* geleert werden. Ziel ist es durch das Betätigen eines Buttons den in ein Textfeld eingegebene Namen auf einem Label auszugeben. Um die benötigten 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:



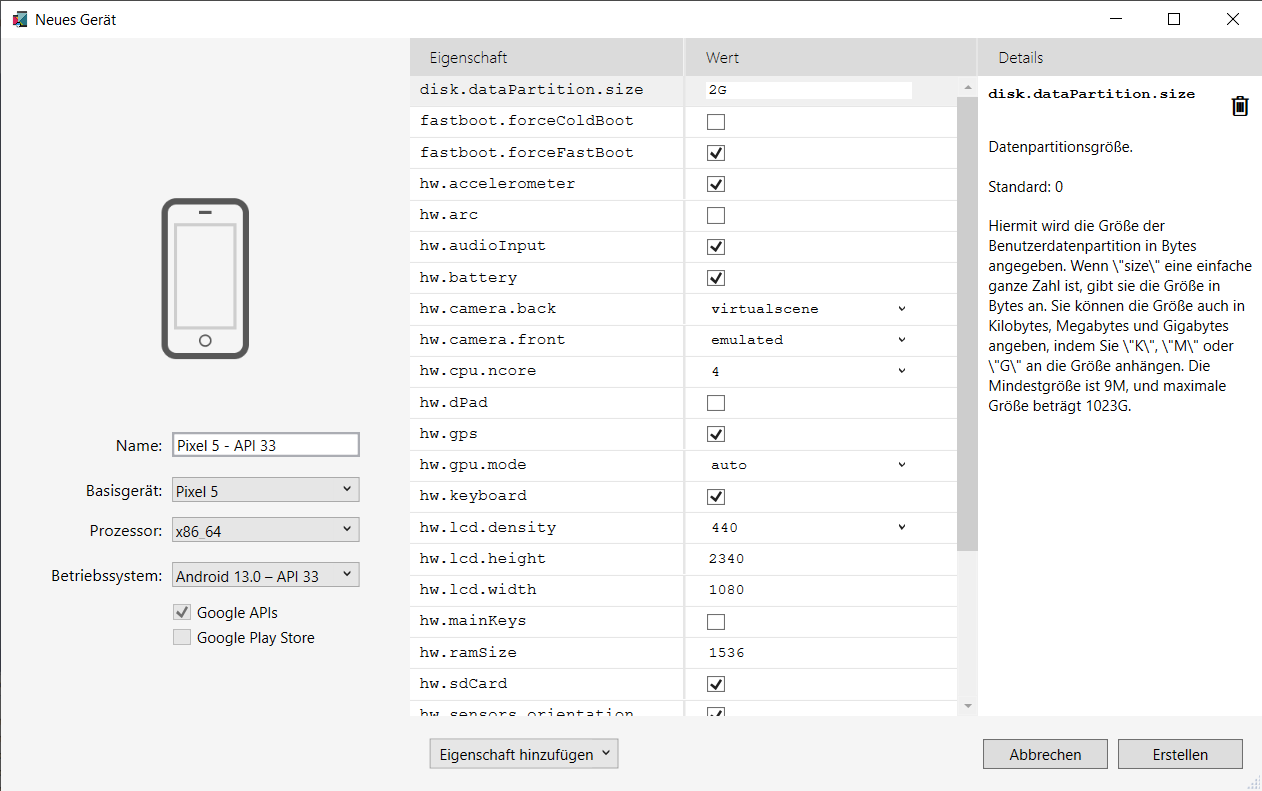
Durch das Einfügen der *Clicked*-Eigenschaft wird innerhalb der *MainPage.xaml.cs* Datei eine Methode angelegt, welche beim Aufruf des Klickereignisses durchlaufen wird. Innerhalb dieser Klasse wird der folgende Quelltext verwendet, um den eigetragenen Namen zu holen, zu modifizieren und erneut auszugeben.



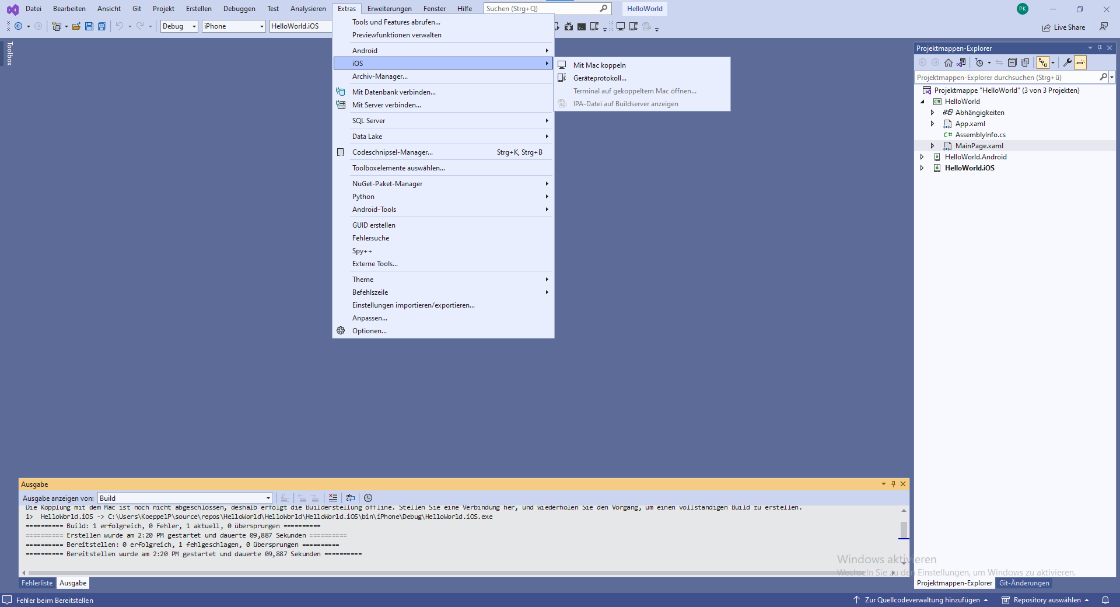
Bei der ersten Ausführung des Quelltextes müssen für iOS und Android einige Vorbereitungen getroffen werden. Um das Android-Projekt zu testen, muss im *Android Device Manager* ein neuer Emulator angelegt werden. Dieser kann über **Extras -> Android -> Android-Geräte-Manager** geöffnet werden.

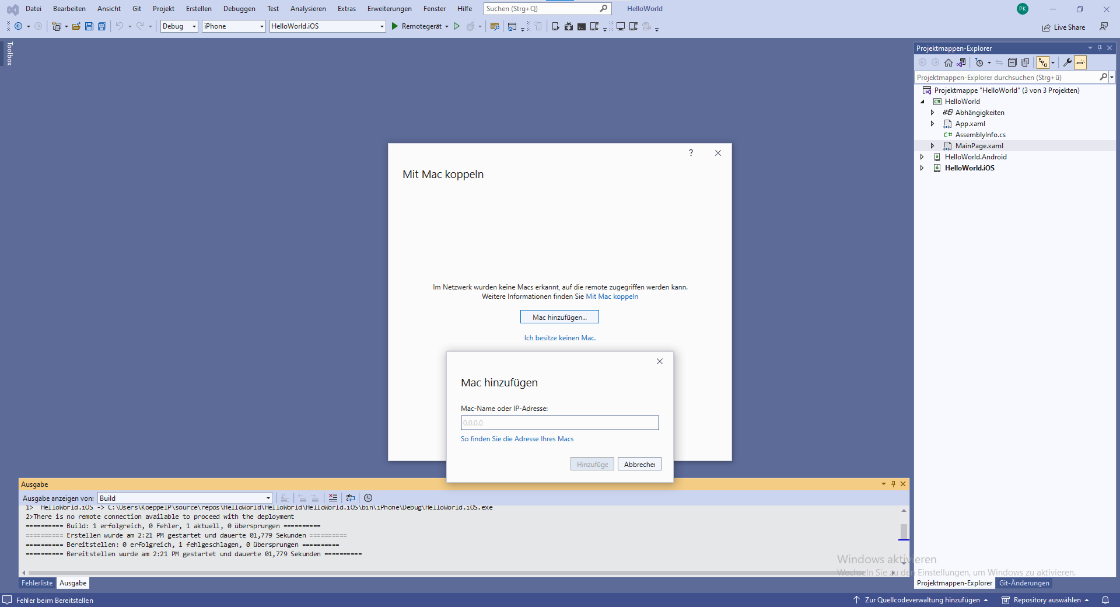


In dem geöffneten Fenster kann nun ein neues Gerät erstellt werden. Der Manager ermöglicht das Arbeiten mit verschiedenen Geräten mit unterschiedlichen Prozessoren und Betriebssystemen. Der Nutzer hat die Möglichkeit das Smartphone nach seinen eigenen Vorstellungen anzupassen. Zu testzwecken sind die 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

## Veröffentlichung von nativen Apps

Um eine Native App zu veröffentlichen, muss sie auf einer entsprechenden Plattform hochgeladen werden. Die gängigsten Plattformen sind hierbei der AppStore von Apple und der Play Store von Google. Jedoch benötigt es einige Schritte, um die App final hochzuladen, sodass andere Nutzer sie benutzen können. Bevor die App veröffentlicht werden kann, gibt es einige Kosten, die berücksichtigt werden müssen. Falls die App Daten speichert oder auf sie zugreifen soll, wird eine Datenbank benötigt, die Kosten orientieren sich hierbei daran, wie viele User auf die Datenbank zugreifen und wie groß diese sein muss. Neben einer Datenbank tauchen auch Kosten für eine API auf, welche die Datenbank verwaltet, dies sind aber nur fällig, falls man eine API von Drittanbietern nutzt. Ein weiterer Punkt für die Kosten wäre der Server, welcher benötigt wird, um die Anwender der App zu hosten, hier richtet sich der Preis nach der Anzahl der aktiven Nutzer. Die letzten Kostenpunkte sind Wartung und Service, bei allen angebrachten Kostenpunkten handelt es sich um laufende Kosten. Das bedeutet, dass diese Kosten jährlich bzw. monatlich anfallen und bei der Preiskalkulierung beaufsichtigt, werden müssen. Neben den laufenden Kosten verlangen die Inhaber der Plattform, auf welcher die App hochgeladen wird, auch Geld. Bevor die App im App-Store veröffentlicht werden kann, wird als allererstes eine Mitgliedschaft im Apple Developer Programm benötigt, welche 99 € pro Jahr kostet. Weiterhin bekommt Apple anfangs eine Provision von 30 % und nach einem Jahr nur noch 15 % von den Einnahmen der App. Um die App zu veröffentlichen, benötigt man außerdem einen ITunes Connect Account und eine DUNS -Nummer. Des Weiteren muss die App den Gemeinschaftsrichtlinien entsprechen und Security und DSGVO Vorgaben erfüllen. Wenn alles abgeschlossen ist, kann die App über App-Store Connect hochgeladen werden. Danach kann man Sceenshots, Titel und Beschreibung hinzufügen und nachdem sie von Apple überprüft wurde, wird die App freigeschaltet. Wenn die App im Play-Store hochgeladen werden soll, benötigt man ein Entwicklerkonto, welches 25 € pro Jahr kostet. Zudem wird auch eine Provision von anfangs 30 % und nach einem Jahr 15 % verlangt. Um die App final zu veröffentlichen, müssen auch hier die Gemeinschaftsrichtlinien, Security- und DSGVO Vorgaben beachtet werden. Ist das alles ordnungsgemäß, kann die APP per APK-Datei hochgeladen werden und zum Schluss noch Screenshots, Titel, Kurzbeschreibung, Vorschaugrafik und Kategorie festgelegt werden. Freigeschaltet wird die App aber erst, nach dem sie von Google geprüft wurde.

# Aufgaben

## geführte Aufgabe

## selbstständige Aufgabe

# Quellenverzeichnis