

# Belegarbeit

## Internet-Technologien

App-Entwicklung mit Xamarin in Visual Studio

**Vorgelegt am:** 20.11.2023

**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

<b>Inhaltsverzeichnis .....</b>	<b>II</b>
<b>1 Zielführung .....</b>	<b>3</b>
<b>2 theoretische Grundlagen.....</b>	<b>3</b>
2.1 Cross-Plattform-App.....	3
2.2 Xamarin .....	4
<b>3 praktische Grundlagen .....</b>	<b>5</b>
3.1 Einrichten von Visual Studio.....	5
3.2 Verwendung von Xamarin .....	11
3.2.1 Anlegen einer neuen Klasse .....	11
3.2.2 Zugriff auf Gerätehardware und -software .....	11
3.3 Veröffentlichung von Cross-Plattform-Apps.....	16
<b>4 Aufgaben .....</b>	<b>17</b>
4.1 geführte Aufgabe .....	17
4.2 selbstständige Aufgabe .....	19
<b>Quellenverzeichnis.....</b>	<b>21</b>
<b>Anhangsverzeichnis.....</b>	<b>22</b>

# 1 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 einmal entwickelt und in nativen Paketen kompiliert werden. Ein geführtes Beispiel soll zu einem besseren Verständnis beitragen und das Lösen der eigenständigen Aufgabe ermöglichen.

## 2 theoretische Grundlagen

### 2.1 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, beispielsweise für den Zugriff auf Sensoren und Aktoren, individuell an das jeweilige Gerätesystem angepasst werden. Deshalb ist die Entwicklung einer Cross-Plattform-Applikation ungefähr 20% günstiger, zeitsparender und wartungsfreundlicher als die Entwicklung von unabhängigen Apps. Jedoch ist der Speicherbedarf durch die Kompilierung größer als der einer nativen Applikationen. 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 nutzt auch Web-Anwendungen.<sup>1</sup>

---

<sup>1</sup> vgl. itPortal24, 2023

## 2.2 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.<sup>2</sup>

---

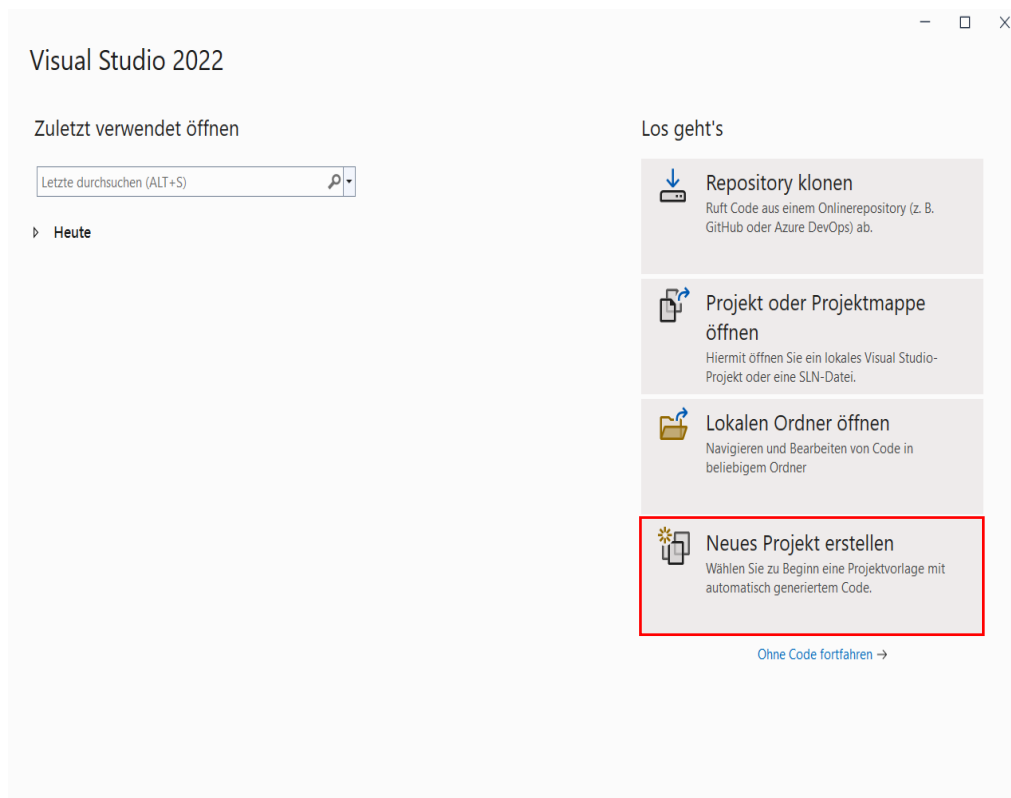
<sup>2</sup> vlg. Microsoft 4, 2023

## 3 praktische Grundlagen

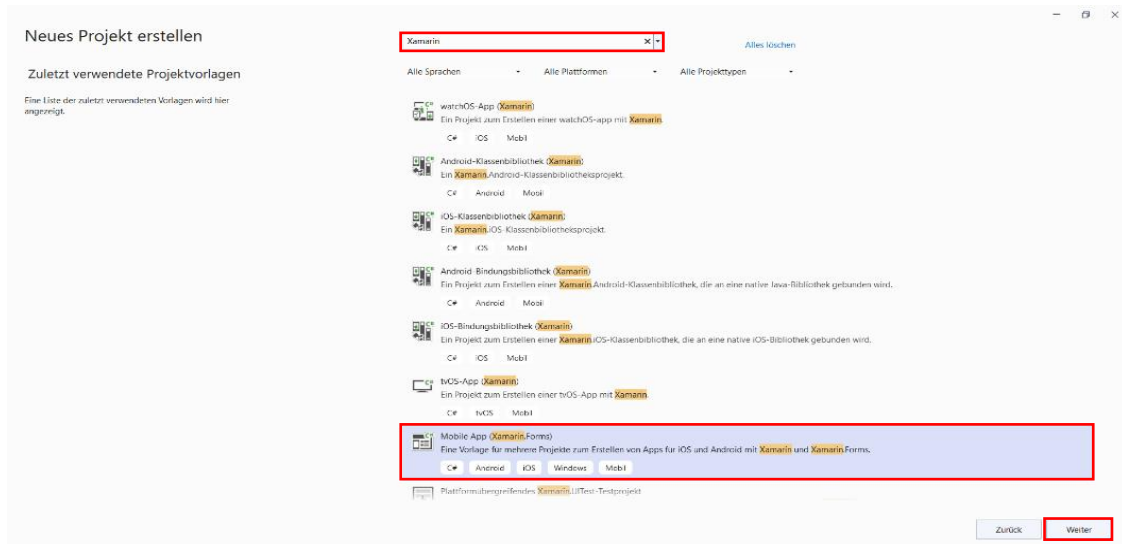
### 3.1 Einrichten von Visual Studio

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.

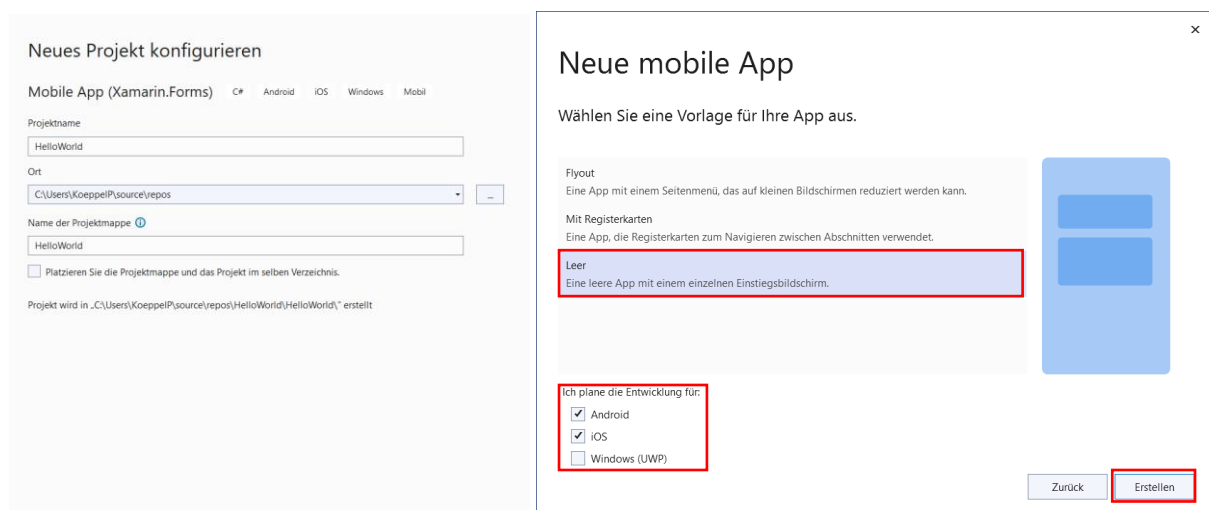
Nach Abschluss des Installationsvorgangs kann Visual Studio gestartet und ein erstes Startprojekt angelegt werden. Zur Erstellung ist *Neues Projekt erstellen* auszuwählen.



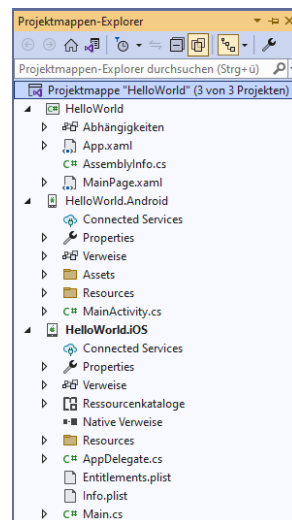
Um in die passende Vorlage zu finden, ist in die Suchleiste *Xamarin* einzugeben. Als Ergebnis werden verschiedene Entwürfe für Plattformen wie Android und iOS und unterschiedliche Geräte wie Handys, Uhren und TV-Geräte vorgeschlagen. Für die Entwicklung einer Cross-Plattform-App ist die Verwendung von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



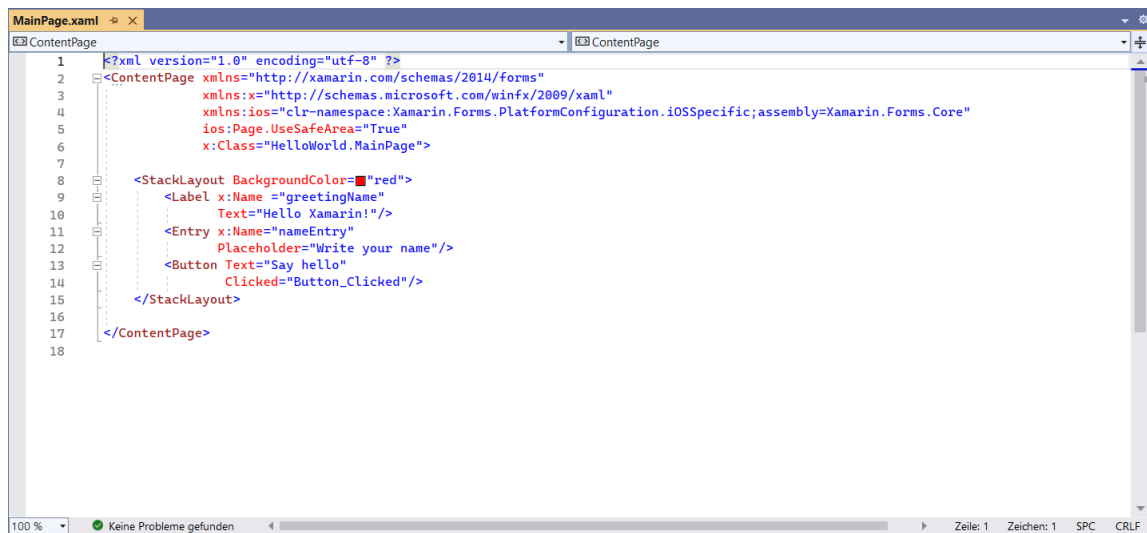
Als simpler Projektname für die Starter-App ist *HelloWorld* angebracht. Im nächsten Schritt ist eine Vorlage für die Anwendung auszuwählen. In diesem Testfall ist *Leer* zu wählen, um das Projekt so einfach wie möglich zu erstellen. Da eine Android und iOS-Applikation zu erstellen ist, sind diese beiden 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 Hauptprojekts 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 eingetragene 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:

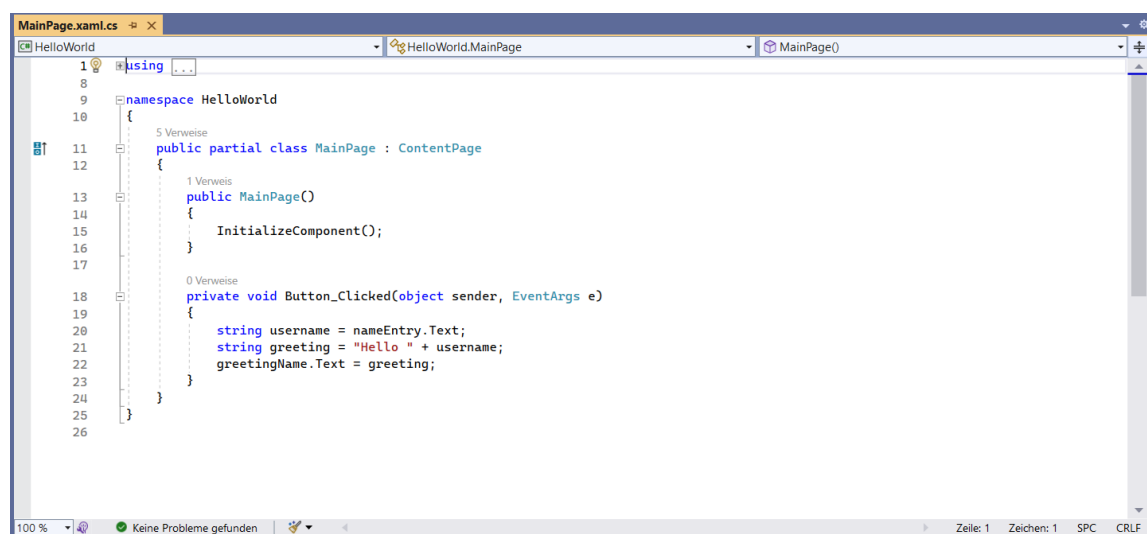


```

1  <?xml version="1.0" encoding="utf-8" ?>
2  <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3      xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4      xmlns:ios="clr-namespace:Xamarin.Forms.PlatformConfiguration.iOSSpecific;assembly=Xamarin.Forms.Core"
5      ios:Page.UseSafeArea="True"
6      x:Class="HelloWorld.MainPage">
7
8      <StackLayout BackgroundColor="red">
9          <Label x:Name="greetingName"
10             Text="Hello Xamarin!"/>
11          <Entry x:Name="nameEntry"
12             Placeholder="Write your name"/>
13          <Button Text="Say hello"
14             Clicked="Button_Clicked"/>
15      </StackLayout>
16
17  </ContentPage>
18

```

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 eingegebenen Namen zu holen, zu modifizieren und erneut auszugeben.



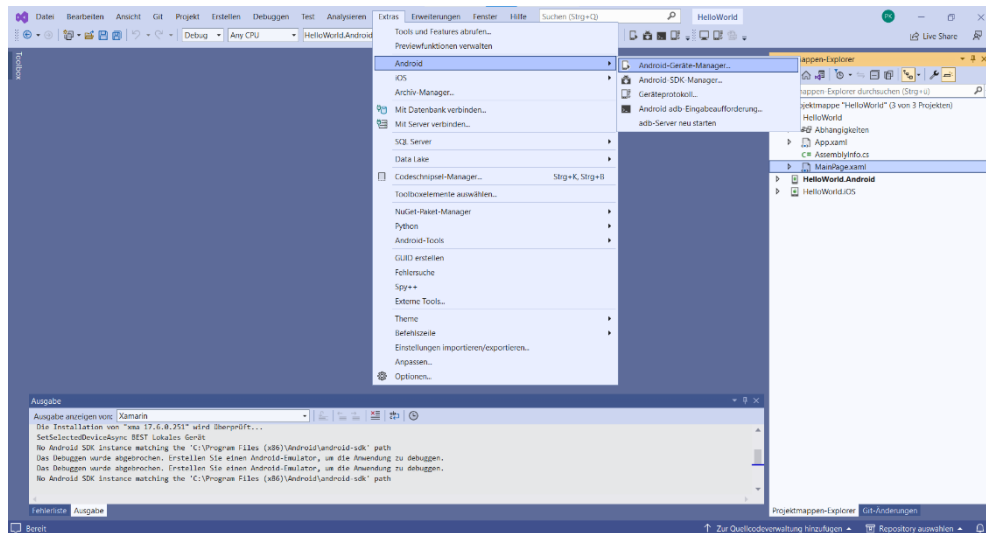
```

1  using System;
2
3  namespace HelloWorld
4  {
5      5 Verweise
6      public partial class MainPage : ContentPage
7      {
8          1 Verweis
9          public MainPage()
10         {
11             InitializeComponent();
12         }
13
14         0 Verweise
15         private void Button_Clicked(object sender, EventArgs e)
16         {
17             string username = nameEntry.Text;
18             string greeting = "Hello " + username;
19             greetingName.Text = greeting;
20         }
21     }
22 }
23
24
25
26

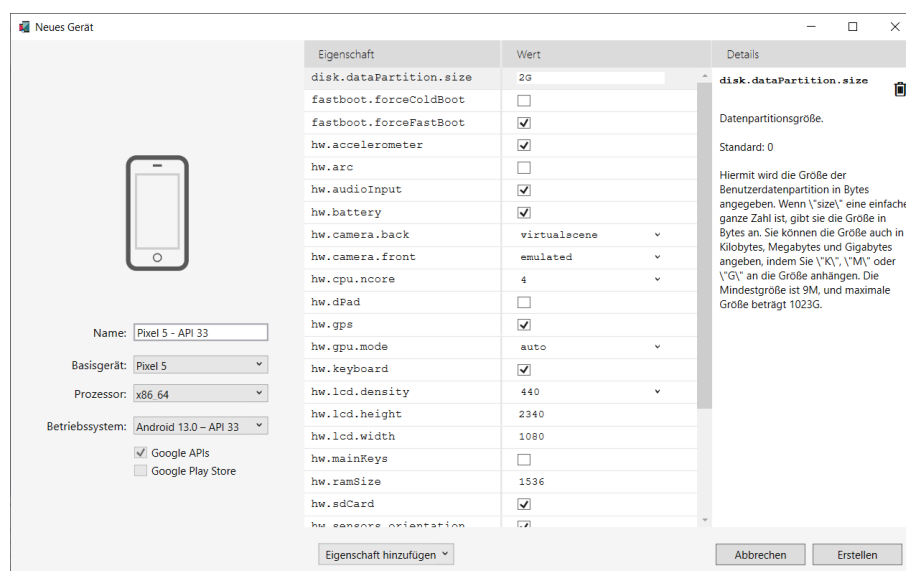
```



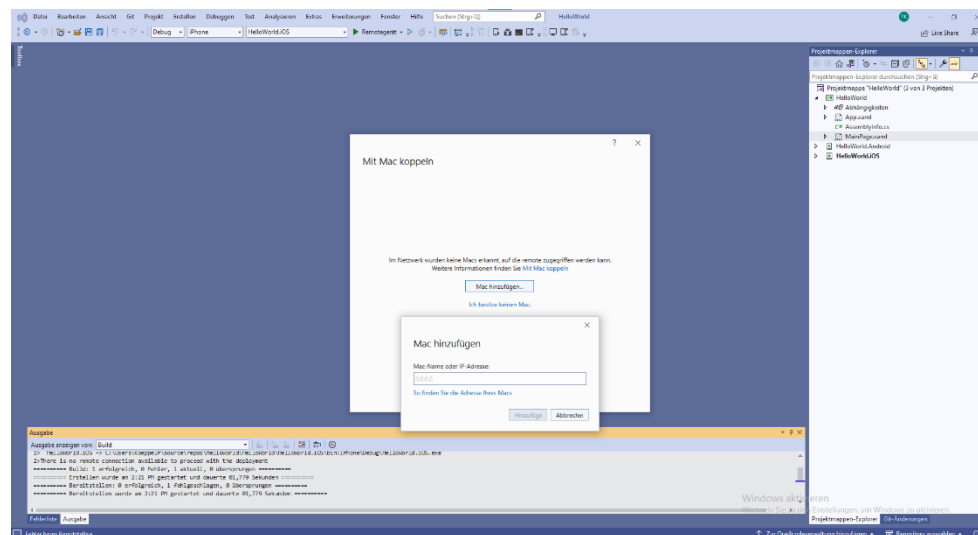
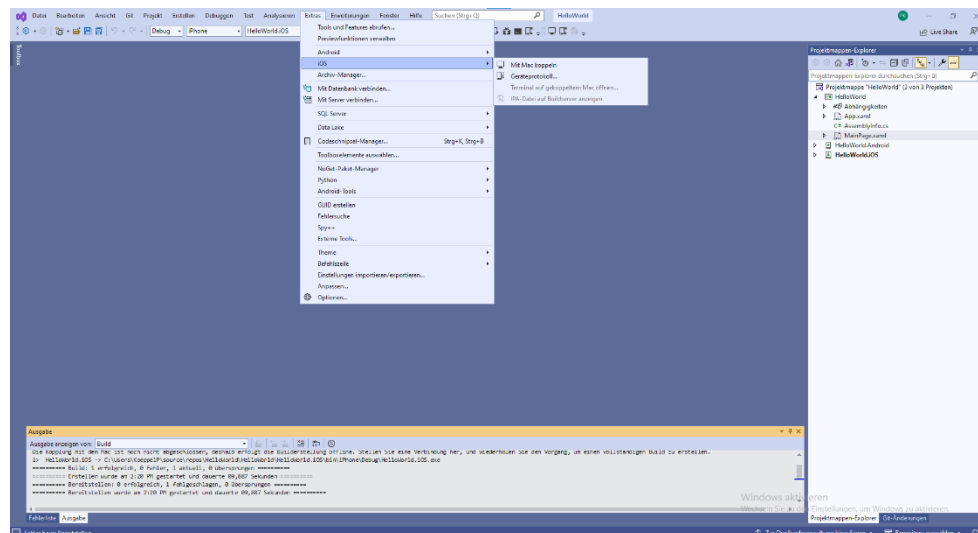
Bei der ersten Ausführung des Quelltextes müssen für iOS und Android einige Vorbereitungen getroffen werden. Um das Android-Projekt zu testen, muss im *Android Device Manager* ein neuer Emulator angelegt werden. Dieser kann über **Extras -> Android -> Android-Geräte-Manager** geöffnet werden.



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 Standardinstellungen 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.



Eine zweite mögliche Option 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.

## 3.2 Verwendung von Xamarin

### 3.2.1 Anlegen einer neuen Klasse

Xamarin.Forms besitzt bereits mehrere Vorlagen zur Erstellung von neuen Dateien. Es gilt 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.

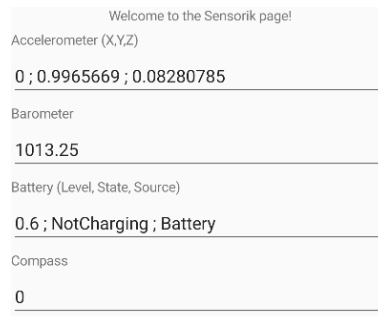
### 3.2.2 Zugriff auf Gerätehardware und -software

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())`

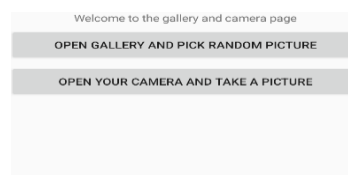
Die Oberfläche der neu erzeugten Klasse `SensorPage` beinhaltet ausschließlich lesbare Textfelder, welche die Werte der Sensoren ausgeben und Beschriftungen für diese Textfelder.



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.

```
<uses-permission android:name="android.permission.BATTERY_STATS" />
<uses-permission android:name="android.permission.VIBRATE" />
```

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.

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<queries>
  <intent>
    <action android:name="android.media.action.IMAGE_CAPTURE" />
  </intent>
</queries>
```

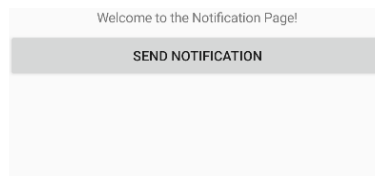
Ein weiteres Beispiel beschäftigt sich mit dem Umgang mit Karten in `Xamarin.Forms` in der neu erzeugten Klasse `MapsPage.xaml`. Die Seite besteht aus einem Willkommenslabel, zwei Textfeldern zur Ausgabe und Eingabe des gewählten Länge- und Breitengrades, einem Button, der die aktuelle Position ermittelt,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

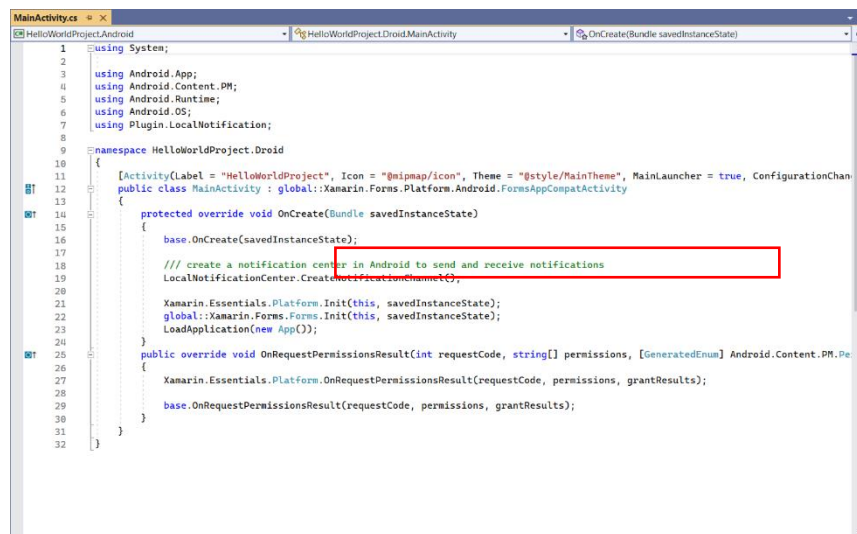
```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

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 Benachrichtigung. Hierfür kann zum Beispiel das `Plugin.LocalNotification` 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.

### 3.3 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.<sup>3</sup>

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

Sind die angegebenen 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 hinzugefügt. 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.<sup>4</sup>

---

<sup>3</sup> vgl. Gestaltenlernen, kein Datum

<sup>4</sup> vgl. Microsoft 5, 2023



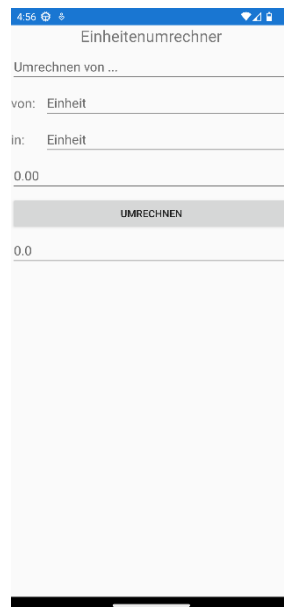
## 4 Aufgaben

### 4.1 geführte Aufgabe

Die Aufgabe ist es, eine eigene Applikation zu programmieren, um Einheiten umzurechnen. Die Anwendung soll die Möglichkeit bieten Währungen, Strecken, Gewichte, Geschwindigkeiten und Datengrößen von einer gewählten Einheit, in eine jeweils andere Einheit umzurechnen.

Die Auswahl was konvertiert wird und welche Größen relevant sind, wird mittels Pickern ausgewählt. Die Eingabe des umzurechnenden Wertes soll in einer Entry eingegeben werden. Um eine fehlerhafte Eingabe zu verhindern, wird die Tastatur auf numerisch beschränkt. Nach des Drücken des Buttons wird das Ergebnis in einer weiteren Entry, welche nur lesen erlaubt, ausgegeben.

Die Beispiel der Oberfläche sieht wie Folgt aus:



In der Logik des Programms wird bei der Initialisierung der erste Picker zur Wahl der Konvertierung mit den Möglichkeiten gefüllt. Die Picker zur Wahl der relevanten Größen wird bei jeder Änderung des ersten Pickers, neu mit den entsprechenden Werten gefüllt.

Wenn die Auswahl der gewünschten Werte erfolgt ist und der umzurechnende Wert eingegeben wurde, kann der Button zum Errechnen des Ergebnisses betätigt werden. Hierbei wird mittels try-catch geprüft, ob der eingegebene Wert in eine Gleitkommazahl umgerechnet werden kann. Ist dies der Fall wird der Wert zuerst in eine Standardeinheit umgerechnet. Diese sind der amerikanische Dollar für Währungen, der Meter bei Strecken, das Gramm bei Gewichten, Meter pro Sekunde bei Geschwindigkeiten und Byte bei den Datengrößen. Im Anschluss daran wird der Standardwert in die gewünschte Zielgröße umgerechnet.

Die Umrechnungszahlen werden zum großen Teil als konstante Werte in das Programm eingepflegt, da es sich um nicht änderbare Werte handelt. Die einzigen veränderlichen Werte sind die Wechselkurse der Währungen. Um diese auf einen aktuellen Stand zu halten, wird eine kostenlose API verwendet, welche monatlich 5000 kostenlose Anfragen genehmigt. Um die <https://freecurrencyapi.com/> zu verwenden ist lediglich eine Registrierung mit Mail-Adresse und einem Passwort notwendig. Um die API zu verwenden, muss das NuGet-Paket `freecurrencyapi` installiert werden. Ist eine Installation aufgrund einer veralteten DotNet-Version nicht möglich, muss unter Projekt->Eigenschaften das Zielframework angepasst werden. Um das Framework verwenden zu können, müssen einige Vorbereitungen getroffen werden. Im ersten Schritt muss eine neue Instanz mit dem Schlüssel als Parameter angelegt werden. Den API-Schlüssel findet man auf dem Dashboard der Webseite nach der Anmeldung. Mittels dieser Instanz ist es möglich den aktuellen Wechselkurs der Währungen abzufragen. Das Ergebnis der Abfrage ist ein String im JSON-Format. Um diesen auswerten zu können muss eine neue Klasse angelegt und das NuGet-Paket `Newtonsoft.Json` installiert werden. Die Klasse enthält eine Funktion zum setzen und holen der Werte in JObject-Format. Der String kann nun deserialisiert werden.

Im Anhang finden sich Klassendiagramme und Struktogramme zur erleichternden Entwicklung.

## 4.2 selbstständige Aufgabe

Es soll ein eigener Taschenrechner implementiert werden. Dieser Rechner besteht aus einem 6x4 großem Grid mit verschiedenen Buttons und einer Entry.

Zeile	Spalte	Element
1	1 - 3	Entry zum Anzeigen der Zahlen
1	4	Button zum Entfernen des letzten Zeichens
2	1	Button zum leeren der Entry
2	2	Button zum umkehren des Vorzeichens
2	3	Button zur Prozentrechnung
2	4	Button zur Division
3	1 - 3	Button 7 - 9
3	4	Button zur Multiplikation
4	1 - 3	Button 4 - 6
4	4	Button zur Subtraktion
5	1	Button 1 - 3
5	4	Button zur Addition
6	1	Button 0
6	2	Button zum setzen des Kommas
6	3	Button zum Umschalten der Felder
6	4	Button zum Anzeigen des Ergebnisses

Die Entry ist nur lesbar und kann nicht durch die Eingabe von einem Nutzer manipuliert werden. Wurde ein Operationszeichen gedrückt, soll der Wert gespeichert und das Feld gegebenenfalls aktualisiert werden. Die Entry bleibt so lange bestehen, bis ein neuer Zahlenwert eingegeben wurde.

Der Button zum Entfernen der zuletzt eingegebenen Zahl (1;4) löscht nur wenn zuvor eine Zahl eingegeben wurde. Steht nur die Zeichenkette „0.“ innerhalb der Entry, wird dies Kette komplett gelöscht.

Der Button zum Leeren des Eingabefeldes (2;1) wechselt die Beschriftungen zwischen „AC“, wenn kein Wert eingegeben wurde und die gesamte Operation abgebrochen werden soll, und „CC“, wenn nur der aktuelle Zahlenwert zurückgesetzt werden soll.

Der Button zum Wechseln des Vorzeichens (2;2) verändert sich in Abhängigkeit der eingegeben Zahl zwischen „-“ und „+“.

Der Button für die Prozentrechnung (2;3) unterscheidet, ob eine Operation ausgeführt wird oder nicht. Wenn zuvor kein Operationszeichen gesetzt wurde, wird die eingegebene Zahl auf 100 bezogen. Sonst wird der Prozentsatz auf den vorherigen Operanten bezogen und die Operation ausgeführt.

Der Button für die Division (2;4) verhindert das Teilen durch die Zahl „0“.

Der Button zum Ausgeben einer „0“ (6;1) gibt nur eine Null aus, wenn zuvor eine andere Zahl geschrieben wurde.

Der Button zum Setzen des Kommas (6;2) kann in einer Zahlenfolge nur einmal betätigt werden. Wenn das Komma an erster Stelle gesetzt wird, soll eine führende „0“ mit Ausgegeben werden.

Der Umschaltbutton (6;3) schaltet weitere Buttons frei. Diese sind zu Beginn nicht sichtbar und werden durch das betätigen sichtbar. Die Buttons, welche zuvor an dieser Stelle waren, werden unsichtbar. Wenn der Button ein zweites Mal betätigt oder eine der darin vorhandenen Elemente verwendet wurde. Ändert sich die Ansicht wieder. Folgende Buttons werden verändert:

Zeile	Spalte	Element
3	1	Button für den Logarithmus
3	2	Button für den Logarithmus zu Basis 10
3	3	Button für den Logarithmus Naturalis (Basis e)
3	4	Button für den Logarithmus Dualis (Basis 2)
4	1	Button zur Ausgabe der Eulerschen Zahl
4	2	Button zur Ausgabe von e hoch der eingegebenen Zahl
4	3	Button zur Ausgabe von 10 hoch der eingegebenen Zahl
4	4	Button zur Ausgabe der Zahl Pi
5	1	Button zur Ausgabe einer eingebenden Zahl x hoch einer Zahl y
5	2	Button zur Ausgabe einer eingebenden Zahl x ins Quadrat
5	3	Button zur Ausgabe der Quadratwurzel einer eingebenden Zahl x
5	4	Button zur Ausgabe der x-ten Wurzel
6	1	Button für die Modulo-Funktion
6	2	Button zur Berechnung der Fakultät

Ein beispielhafter Aufbau der Oberfläche der Applikation ist folgender:



## Quellenverzeichnis

Bluesource (kein Datum): *Web App vs. native App - Was ist die bessere Wahl?*.  
<https://www.bluesource.at/blog/detail/web-apps-vs-native-app>  
[Zugriff am 17.10.2023].

Gestaltenlernen (kein Datum): *Eine App im App Store veröffentlichen*.  
<https://gestaltenlernen.ch/app-design/eine-app-im-app-store-veroeffentlichen>  
[Zugriff am 02.10.2023].

IntCoder (17.12.2021): *Using GitHub in Visual Studio 2022*.  
<https://www.youtube.com/watch?v=yDIncg-5c8Y> [Zugriff am 11.01.2023].

itPortal24 (14.09.2023): *Cross-Platform App - Plattformübergreifende Entwicklung mit Flutter, React Native & Co.* <https://www.itportal24.de/ratgeber/cross-platform-app>  
[Zugriff am 15.09.2023].

Microsoft 1 (30.11.2022): *Tutorial: Öffnen eines Projekts von einem Repository aus*.  
<https://learn.microsoft.com/de-de/visualstudio/get-started/tutorial-open-project-from-repo?view=vs-2022> [Zugriff am 11.01.2023].

Microsoft 2 (kein Datum): *Visual Studio: IDE und Code-Editor für Softwareentwickler und -teams*. <https://visualstudio.microsoft.com/de/> [Zugriff am 11.01.2023].

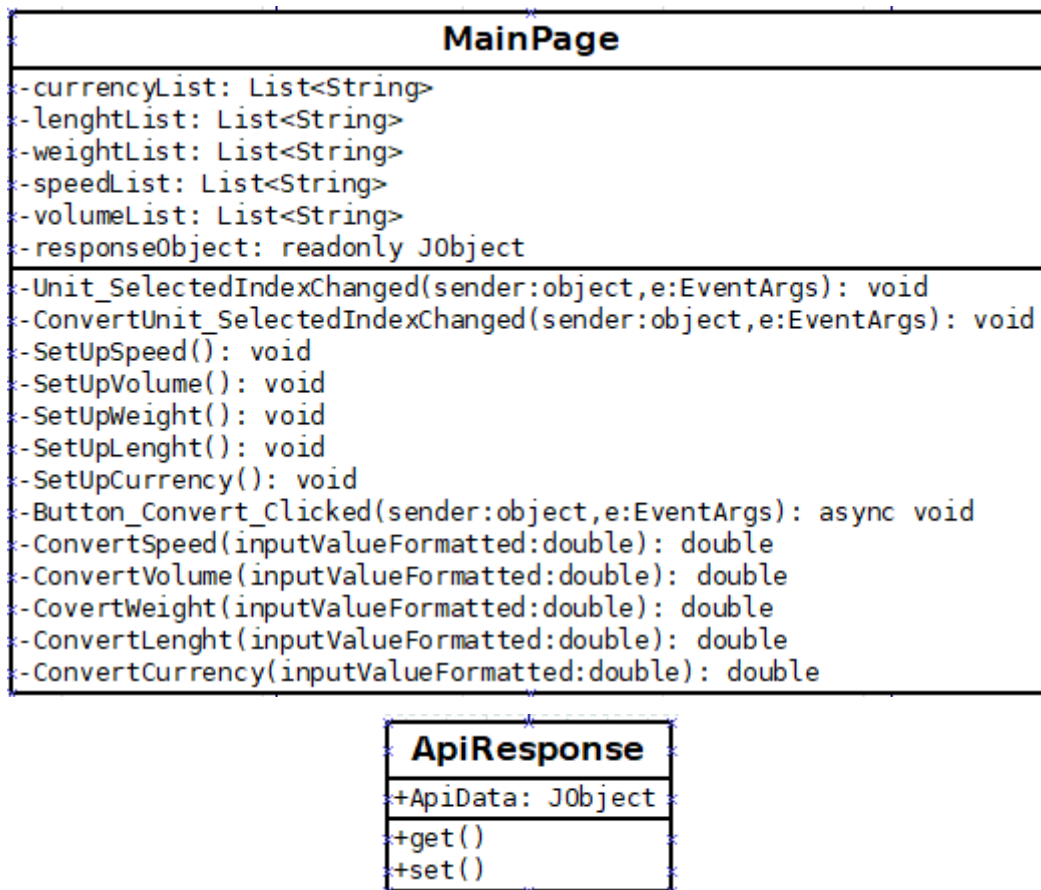
Microsoft 3 (kein Datum): *Visual Studio und GitHub*.  
<https://visualstudio.microsoft.com/de/vs/github/> [Zugriff am 11.01.2023].

Microsoft 4 (14.07.2023): *Was ist Xamarin?*.  
<https://learn.microsoft.com/de-de/xamarin/get-started/what-is-xamarin>  
[Zugriff am 15.09.2023].

Microsoft 5 (19.07.2023): *Veröffentlichen im Google Play Store*.  
<https://learn.microsoft.com/de-de/appcenter/distribution/stores/googleplay>  
[Zugriff am 02.10.2023].

## **Anhangsverzeichnis**

Anhang 1	Klassendiagramme für den Einheitenumrechner
Anhang 2	Struktogramme für den Einheitenumrechner
Anhang 3	Quellcode der Projekte



**Unit\_SelectedIndexChanged**

input\_value.Text = ""

output\_value.Text = ""

**ConvertUnit\_SelectedIndexChanged**

input\_unit.Items.Clear()

ouput\_unit.Items.Clear()

input\_value.Text = ""

output\_value.Text = ""

convert_unit.SelectedIndex					
0	1	2	3	4	default
SetUpCurrency	SetUpLength	SetUpWeight	SetUpVolume	SetUpSpeed	Ø
input_unit.SelectedIndex = 0					
ouput_unit.SelectedIndex = 1					



**SetUpSpeed()**

speedList = new List<String>()
speedList.Add("µm/s")
speedList.Add("nm/s")
speedList.Add("mm/s")
speedList.Add("m/s")
speedList.Add("km/s")
speedList.Add("km/h")
speedList.Add("mph")
foreach String speed in speedList
input_unit.Items.Add(speed)
output_unit.Items.Add(speed)

**SetUpVolume()**

volumeList = new List<String>()
volumeList.Add("bit")
volumeList.Add("B")
volumeList.Add("kB")
volumeList.Add("MB")
volumeList.Add("GB")
volumeList.Add("TB")
volumeList.Add("PB")
volumeList.Add("EB")
volumeList.Add("ZB")
volumeList.Add("YB")
volumeList.Add("KiB")
volumeList.Add("MiB")
volumeList.Add("GiB")
volumeList.Add("TiB")
volumeList.Add("PiB")
volumeList.Add("EiB")
volumeList.Add("ZiB")
volumeList.Add("YiB")
foreach String volume in volumeList
input_unit.Items.Add(volume)
output_unit.Items.Add(volume)

**SetUpWeight()**

```
weightList = new List<String>()
```

```
weightList.Add("u")
```

```
weightList.Add("ng")
```

```
weightList.Add("mg")
```

```
weightList.Add("g")
```

```
weightList.Add("dag")
```

```
weightList.Add("hg")
```

```
weightList.Add("kg")
```

```
weightList.Add("kN")
```

```
weightList.Add("t")
```

```
weightList.Add("lb")
```

```
foreach String weight in weightList
```

```
    input_unit.Items.Add(weight)
```

```
    output_unit.Items.Add(weight)
```

**SetUpLength()**

```
lengthList = new List<String>()
```

```
lengthList.Add("A")
```

```
lengthList.Add("nm")
```

```
lengthList.Add("µm")
```

```
lengthList.Add("mm")
```

```
lengthList.Add("cm")
```

```
lengthList.Add("dm")
```

```
lengthList.Add("m")
```

```
lengthList.Add("km")
```

```
lengthList.Add("in")
```

```
lengthList.Add("ft")
```

```
lengthList.Add("yd")
```

```
lengthList.Add("mil")
```

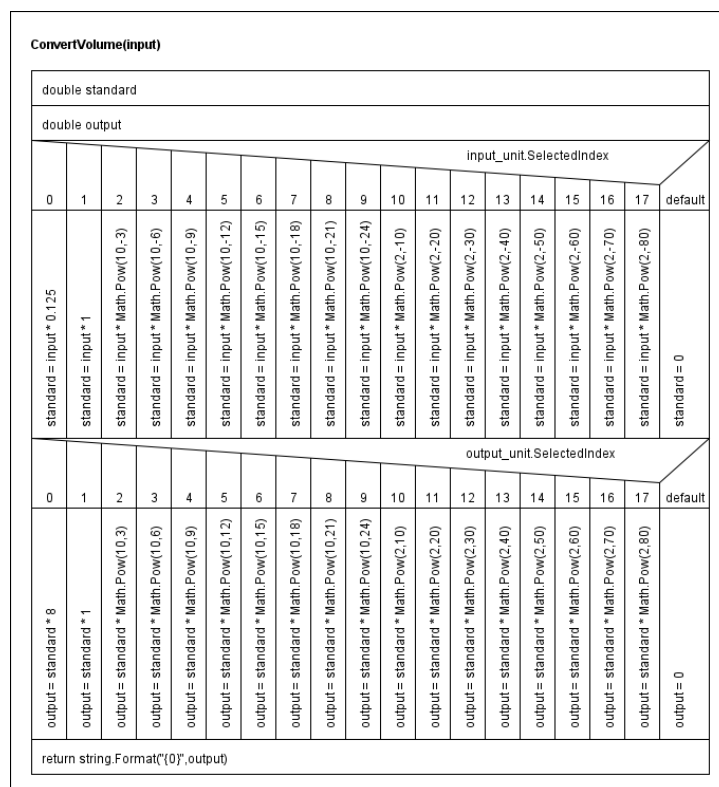
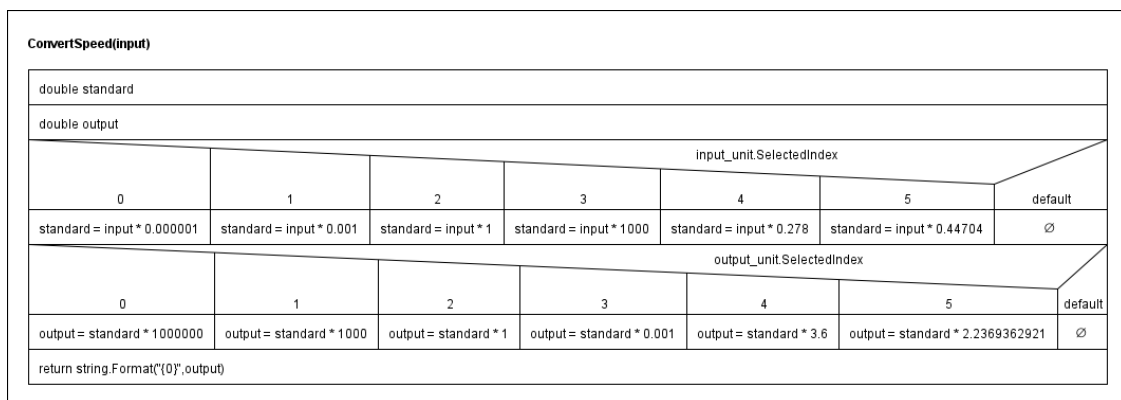
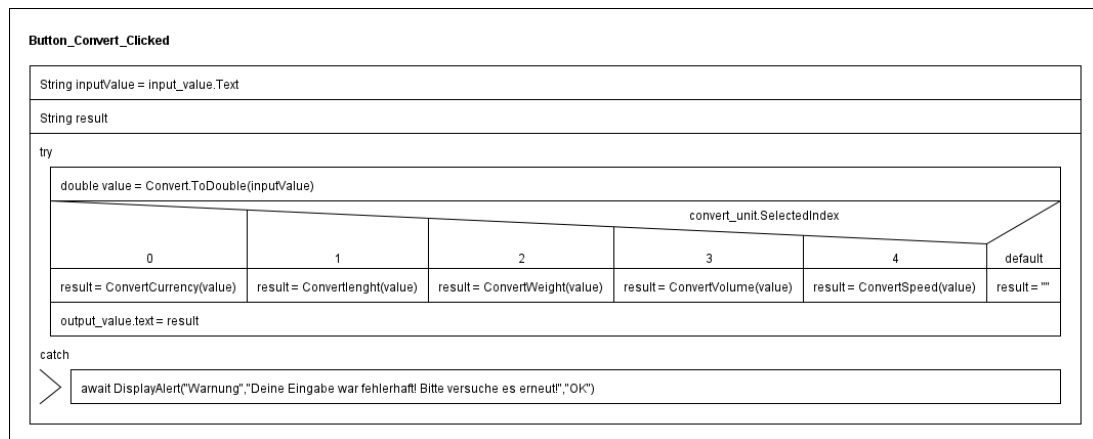
```
foreach String length in lengthList
```

```
    input_unit.Items.Add(length)
```

```
    output_unit.Items.Add(length)
```

**SetUpCurrency()**

currencyList = new List<String>()
currencyList.Add("Euro")
currencyList.Add("US Dollar")
currencyList.Add("Yen")
currencyList.Add("Tschechische Krone")
currencyList.Add("Dänische Krone")
currencyList.Add("Pfund")
currencyList.Add("Zloty")
currencyList.Add("Schwedische Krone")
currencyList.Add("Franken")
currencyList.Add("Rubel")
currencyList.Add("Lira")
currencyList.Add("Real")
currencyList.Add("Yuan")
currencyList.Add("Rupee")
currencyList.Add("Won")
currencyList.Add("Peso")
currencyList.Add("Bitcoin")
currencyList.Add("Mark")
foreach String currency in currencyList
input_unit.Items.Add(currency)
output_unit.Items.Add(currency)



ConvertWeight(input)										
double standard										
double output										
input_unit.SelectedIndex										
0	1	2	3	4	5	6	7	8	9	default
standard = input * 1.6605654724 * Math.Pow(10,-15)	standard = input * 0.0000000001	standard = input * 0.001	standard = input * 1	standard = input * 10	standard = input * 100	standard = input * 1000	standard = input * 1.019716005 * Math.Pow(10,5)	standard = input * 1000000	standard = input * 453.59237	standard = 0
output_unit.SelectedIndex										
0	1	2	3	4	5	6	7	8	9	default
output = standard * 6.022045 * Math.Pow(10,23)	output = standard * 10000000000	output = standard * 1000	output = standard * 1	output = standard * 0.1	output = standard * 0.01	output = standard * 0.001	output = standard * 0.0000098067	output = standard * 0.000001	output = standard * 0.0022046226	output = 0
return string.Format("{0}",output)										

ConvertLength(input)														
double standard														
double output														
input_unit.SelectedIndex														default
0	1	2	3	4	5	6	7	8	9	10	11			
standard = input * 0.000000000001	standard = input * 0.0000000001	standard = input * 0.000001	standard = input * 0.001	standard = input * 0.01	standard = input * 0.1	standard = input * 1	standard = input * 1000	standard = input * 0.0254	standard = input * 0.3048	standard = input * 0.9144	standard = input * 1609.34	standard = 0		
output_unit.SelectedIndex														default
0	1	2	3	4	5	6	7	8	9	10	11			
output = standard * 1000000000000	output = standard * 10000000000	output = standard * 1000000	output = standard * 1000	output = standard * 100	output = standard * 10	output = standard * 1	output = standard * 0.001	output = standard * 39.3701	output = standard * 3.28084	output = standard * 1.09361	output = standard * 0.000621371	output = 0		
return string.Format("{0}",output)														

ConvertCurrency(input)																			
double standard																			
double output																			
																		input_unit Selected index	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	default	
standard = input * 1 / (double) responseObject.GetValue("EUR")	standard = input * 1 / (double) responseObject.GetValue("USD")	standard = input * 1 / (double) responseObject.GetValue("JPY")	standard = input * 1 / (double) responseObject.GetValue("CZK")	standard = input * 1 / (double) responseObject.GetValue("DKK")	standard = input * 1 / (double) responseObject.GetValue("GBP")	standard = input * 1 / (double) responseObject.GetValue("PLN")	standard = input * 1 / (double) responseObject.GetValue("SEK")	standard = input * 1 / (double) responseObject.GetValue("CHF")	standard = input * 1 / (double) responseObject.GetValue("RUB")	standard = input * 1 / (double) responseObject.GetValue("TRY")	standard = input * 1 / (double) responseObject.GetValue("BRL")	standard = input * 1 / (double) responseObject.GetValue("CNY")	standard = input * 1 / (double) responseObject.GetValue("INR")	standard = input * 1 / (double) responseObject.GetValue("KRW")	standard = input * 1 / (double) responseObject.GetValue("MXN")	standard = input * 1 / (double) responseObject.GetValue("MNT")	standard = input * 1 / 3.4215 * Math.Pow(10,-5)	standard = input * 1 / 1.78	Standard = 0
																		output_unit Selected index	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	default	
output = standard * (double) responseObject.GetValue("EUR")	output = standard * (double) responseObject.GetValue("USD")	output = standard * (double) responseObject.GetValue("JPY")	output = standard * (double) responseObject.GetValue("CZK")	output = standard * (double) responseObject.GetValue("DKK")	output = standard * (double) responseObject.GetValue("GBP")	output = standard * (double) responseObject.GetValue("PLN")	output = standard * (double) responseObject.GetValue("SEK")	output = standard * (double) responseObject.GetValue("CHF")	output = standard * (double) responseObject.GetValue("RUB")	output = standard * (double) responseObject.GetValue("TRY")	output = standard * (double) responseObject.GetValue("BRL")	output = standard * (double) responseObject.GetValue("CNY")	output = standard * (double) responseObject.GetValue("INR")	output = standard * (double) responseObject.GetValue("KRW")	output = standard * (double) responseObject.GetValue("MXN")	output = standard * (double) responseObject.GetValue("MNT")	output = standard * 3.4215 * Math.Pow(10,-5)	output = standard * 1.78	output = 0
return string.Format("{0}",output)																			

## Ehrenwörtliche Erklärung

"Ich erkläre hiermit ehrenwörtlich",

1. dass wir unsere ...Belegarbeit Internet-Technologien..... mit dem Thema  
App-Entwicklung mit Xamarin in Visual Studio  
.....  
.....

ohne fremde Hilfe angefertigt haben,

2. dass wir die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet haben und
3. dass wir unsere ...Belegarbeit..... bei keiner anderen Prüfung vorgelegt haben

Wir sind uns bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Glauchau, 17.11.2023

Ort, Datum



Unterschrift



## Erklärung zur Prüfung wissenschaftlicher Arbeiten

Die Bewertung wissenschaftlicher Arbeiten erfordert die Prüfung auf Plagiate. Die hierzu von der Staatlichen Studienakademie Glauchau eingesetzte Prüfungskommission nutzt sowohl eigene Software als auch diesbezügliche Leistungen von Drittanbietern. Dies erfolgt gemäß § 7 des Gesetzes zum Schutz der informationellen Selbstbestimmung im Freistaat Sachsen (Sächsisches Datenschutzgesetz - SächsDSG) vom 25. August 2003 (Rechtsbereinigt mit Stand vom 31. Juli 2011) im Sinne einer Datenverarbeitung im Auftrag.

Der Studierende bevollmächtigt die Mitglieder der Prüfungskommission hiermit zur Inanspruchnahme o. g. Dienste. In begründeten Ausnahmefällen kann der Datenschutzbeauftragte der Berufsakademie Sachsen sowohl vom Verfasser der wissenschaftlichen Arbeit als auch von der Prüfungskommission in den Entscheidungsprozess einbezogen werden.

Name:	Köppel	Kunze
Vorname:	Pascal	Elias
Matrikelnummer:	4004454	4004399
Studiengang:	Technische Informatik	
Titel der Arbeit:	App-Entwicklung mit Xamarin in Visual Studio	
Datum:	17.11.2023	
Unterschrift:		

---