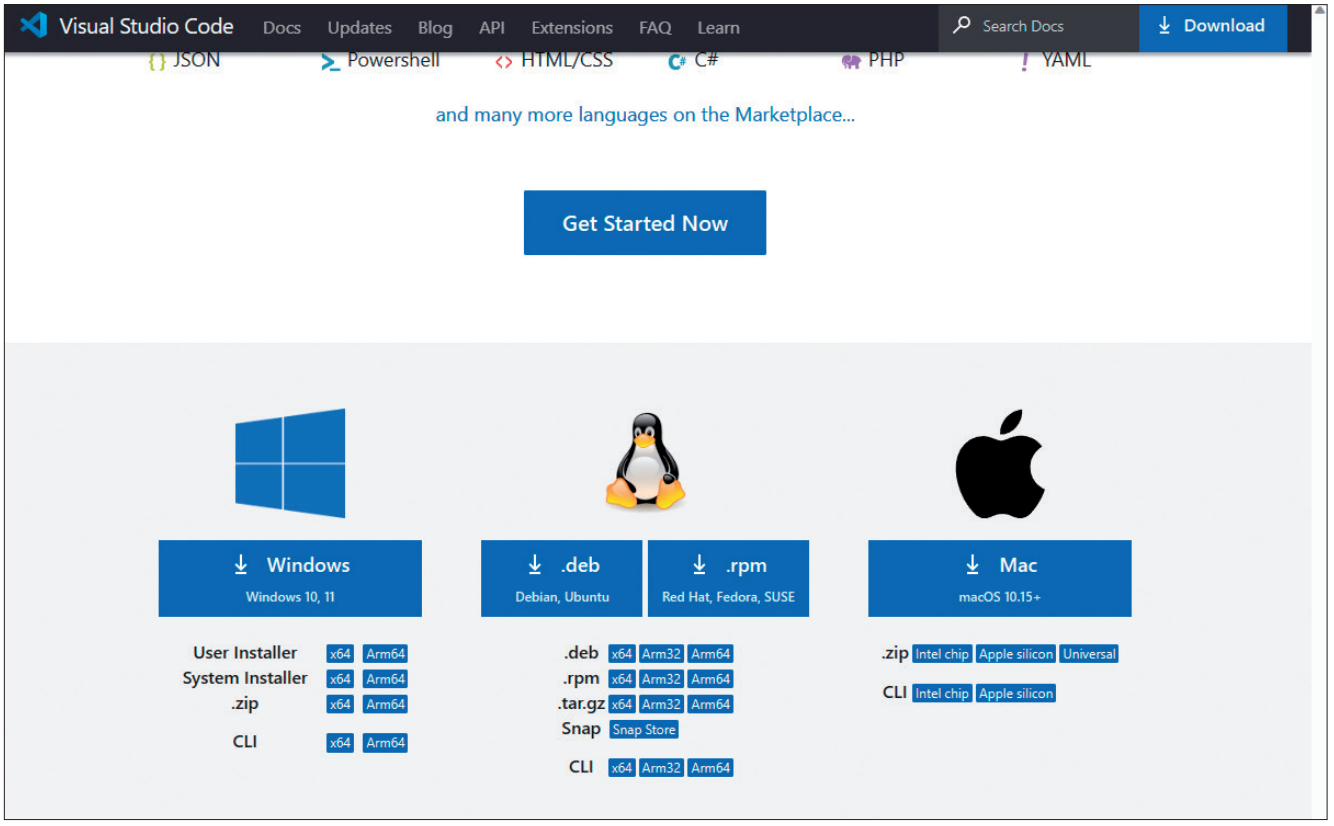
VS CODE AUF WINDOWS UND MAC

# Brauchbare Alternative

## Das C# Dev Kit for Visual Studio Code auf dem Prüfstand.

Bisher gibt es die Möglichkeit, .NET-Apps mit Visual Studio unter Windows oder macOS zu erstellen. Allerdings endet der Support für die macOS-Version von Visual Studio durch Microsoft bereits im August 2024. Das C# DevKit als Extension für VS Code könnte eine Alternative für Entwickler sein, die weiterhin macOS nutzen möchten. Weshalb wir die Praxistauglichkeit geprüft haben.

- **Windows:** Aktuell für die Versionen Windows 10 und Windows 11.
- **macOS:** Unterstützung der jüngsten Versionen; im Moment ist zudem nach dem Prozessortyp zu differenzieren (Intel oder ARM M1, M2, M3).
- **Linux:** Hier kann man gegebenenfalls unterschiedliche Distributionen adressieren.




**VS Code steht als Download** für viele Systeme zur Verfügung (Bild 1)

Entwickler sind heute gefordert, Software für unterschiedliche Betriebssysteme und Geräteklassen zu erstellen, was gerne als „plattform- und geräteübergreifende Entwicklung" bezeichnet wird. Auf der anderen Seite möchte man bei der Softwareentwicklung möglichst flexibel bleiben, also nicht nur auf der Zielplattform, sondern auf den unterschiedlichsten Systemumgebungen entwickeln können.

Ein Beispiel: Bei der plattformübergreifenden Programmierung ist es das Ziel, native Apps für die üblichen Client-Systeme (Betriebssysteme) zu erstellen. Konkret sind das üblicherweise:

- **iOS:** Apps für die aktuellen iOS-Systeme, welche von Apple unterstützt werden.
- **Android:** Große Vielfalt an Betriebssystemversionen, unter anderem auch ältere Versionen auf Geräten, welche keine Updates mehr erhalten.

Für die Entwicklung von Apps für unterschiedliche Zielsysteme gibt es weitere Einschränkungen zu beachten, die sich wie folgt darstellen: Die Entwicklung von Windows-Applikationen kann ausschließlich auf Windows-Systemen durchgeführt werden. Aktuell können dafür sowohl Windows 10 als

auch Windows 11 genutzt werden. Zudem besteht die Möglichkeit, Windows auf einer virtuellen Maschine zu installieren, beispielsweise auf Linux-Rechnern oder unter macOS. Grundsätzlich stehen mehrere Ansätze, Frameworks und Grafikbibliotheken zur Auswahl.

Die Entwicklung von Apps für macOS ist deutlich eingeschränkter, diese kann nur unter macOS stattfinden. Um native Apps zu erstellen, ist die Verwendung der integrierten Entwicklungsumgebung (IDE) Xcode erforderlich. Nur mit ihr lässt sich das finale App-Package für das Betriebssystem erstellen. Eine weitere Einschränkung ist, dass eine Virtualisierung von macOS auf einem anderen Rechnersystem als einem Apple-PC aus lizenzrechtlichen Gründen nicht möglich ist. Kurz: Für das Entwickeln von macOS-Apps wird ein Apple-PC benötigt.

Bei Android ist die größte Flexibilität gegeben: Die Entwicklung ist auf allen Desktop-Betriebssystemen möglich, sowohl unter Windows als auch unter macOS und Linux kann das erforderliche Android-SDK installiert werden. Zudem ist man frei bei der Wahl der Entwicklungsumgebung.

Als Standard gilt die Programmierung mithilfe von Android Studio und den Sprachen Java beziehungsweise Kotlin. Android-Smartphones oder -Tablets können per Kabel an die USB-Buchse des Entwicklungsrechners angeschlossen werden. Alternativ lässt sich eine Verbindung über das lokale Netzwerk (WLAN) herstellen.

Die Entwicklung von Apps für iOS ist ähnlich restriktiv wie die Entwicklung für macOS-Anwendungen: Um das finale App-Paket zu erstellen, ist zwingend ein Apple-PC mit Xcode erforderlich.

Als .NET-Entwickler hat man mit dem Framework .NET MAUI die Möglichkeit, mit dem vertrauten Technology-Stack C#, XAML und .NET Apps für diese Systeme (aktuell mit Ausnahme von Linux) zu entwickeln. Es stellt sich die Frage der IDE. Ein ähnliches Szenario zeigt sich bei der Entwicklung von Web-Apps mit Blazor, da diese ebenfalls auf .NET und C# als Basistechnologien basieren.

Das „große" Visual Studio läuft nur auf einem Windows-Betriebssystem. Zwar steht aktuell mit Visual Studio für Mac noch eine Alternative für das Betriebssystem aus dem Hause Apple zur Verfügung. Bekanntermaßen wird diese speziell angepasste Version der IDE zum 31. August 2024 von Microsoft eingestellt [1]. Funktionsupdates und Erweiterungen sind daher für diese Variante der Entwicklungsumgebung nicht mehr zu erwarten. Für neue Projekte sollte man sich daher einen anderen Werkzeugkasten zusammenstellen. In der Dokumentation heißt es dazu:

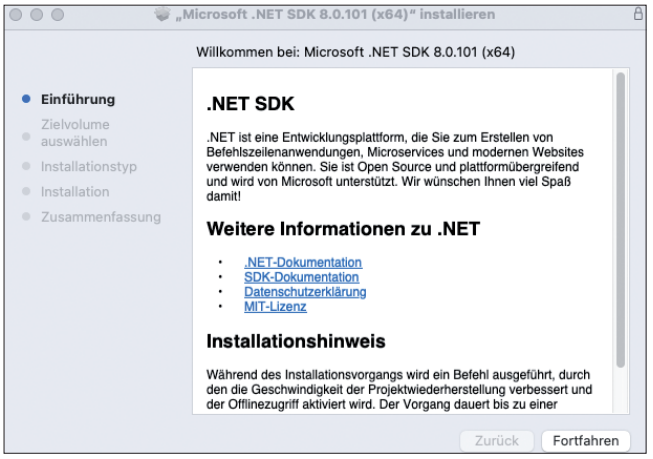
*„Microsoft plant keine Unterstützung für .NET 8 oder C# 12 in Visual Studio für Mac. Wir werden auch vor der Einstellung nicht auf andere Workloads übergehen."*

Als Alternativen für Entwickler, die weiterhin mit .NET auf einem Mac-PC coden möchten, nennt Microsoft folgende Alternativen:

**Visual Studio Code:** Es wird empfohlen, den plattformübergreifend einsetzbaren Editor Visual Studio Code (VS Code) zu nutzen. Für die Anpassung an C#, .NET und .NET MAUI gibt es Erweiterungen. Auf der Webseite dazu heißt es:

*„Diese Erweiterungen funktionieren nativ auf allen unterstützten Plattformen, einschließlich macOS, und die Nutzung dieser Erweiterungen wird auf ihrem Weg von der Vorschauversion zur allgemeinen Verfügbarkeit und darüber hinaus kontinuierlich verbessert. Wir investieren weiterhin in die Entwicklung von C#, .NET MAUI und Unity für Visual Studio Code. Sie sind herzlich eingeladen, für diese Erweiterungen im GitHub-Projekt-Repository von Visual Studio Code Vorschläge zu übermitteln und Probleme zu melden."*

**Visual Studio unter Windows auf einem virtuellen Computer (VM):** Diesen Ansatz haben viele Entwickler bereits genutzt. Sie können einen VM-Host wie Parallels verwenden, um




**Installation des .NET SDK** auf macOS (Bild 2)

Windows einzurichten und mit Visual Studio (Windows) zu arbeiten.

**Visual Studio in der Cloud:** Eine in der Cloud gehostete Virtuelle Maschine von Microsoft Dev Box bietet über einen Webclient oder einen nativen RDP-Client von einem Mac aus Zugriff auf Visual Studio, ohne dass dafür auf dem lokalen Computer ein virtueller Computer eingerichtet und ausgeführt werden muss.

In diesem Artikel testen wir Variante eins, das heißt das individuelle Anpassen von VS Code mit ausgewählten Erweiterungen (Extensions).

## Visual Studio Code

Nahezu alle .NET-Entwickler haben Visual Studio Code vermutlich schon mehrfach im Einsatz gehabt; nichtsdestotrotz lohnt sich ein Blick auf die Möglichkeiten und die Technik dieses Editors.

VS Code ist ein vielseitiger und leistungsfähiger Code-Editor, der von Microsoft entwickelt wurde und sich großer Beliebtheit erfreut. Als Open-Source-Editor ist er kostenlos verfügbar und unterstützt eine Vielzahl von Programmiersprachen wie beispielsweise JavaScript, Python, C# und viele andere mehr.

Visual Studio Code basiert auf dem Electron-Framework, welches die Nutzung von Webtechnologien wie JavaScript, HTML und CSS für Desktop-Anwendungen ermöglicht. ▶

Diese Basis macht Visual Studio Code plattformübergreifend kompatibel mit Betriebssystemen wie etwa Windows, macOS und Linux.

Im Herzen von Visual Studio Code arbeitet der Monaco-Editor, der schnelles Arbeiten erlaubt und umfangreiche Code-Bearbeitungsfunktionen bietet. Der Großteil seines Quellcodes ist in TypeScript geschrieben, der von Microsoft entwickelten Sprache, die JavaScript erweitert und statische Typisierung bietet.

Die Integration von Node.js ermöglicht es VS Code, auf das Dateisystem zuzugreifen und verschiedene Betriebssystemfunktionen zu nutzen. Darüber hinaus profitiert der Editor von der JavaScript Engine V8, die auch in Googles Chrome-Browser zum Einsatz kommt und das effiziente Ausführen von JavaScript-Code gewährleistet. Die Hauptmerkmale des Editors sind folgende:

- **Erweiterbarkeit:** Eines der herausragendsten Merkmale von VS Code ist seine Erweiterbarkeit durch Plug-ins. Nutzer können aus einer Vielzahl von Erweiterungen wählen, die diverse Funktionen mitbringen, Unterstützung für zusätzliche Sprachen bieten, Themes ändern und vieles mehr.
- **Integrierte Git-Unterstützung:** VS Code hat eine eingebaute Unterstützung für Git, was es Benutzern erlaubt, Versionskontrollaufgaben direkt im Editor auszuführen.
- **Intelligenter Code-Editor:** Der Editor bietet Funktionen wie Syntaxhervorhebung, Code-Vervollständigung, Code-Navigation, Refactoring-Tools und vieles mehr.
- **Debugging:** VS Code enthält integrierte Debugging-Tools, die das Setzen von Haltepunkten, Überprüfen von Variablen und Durchführen von Schritt-für-Schritt-Analysen ermöglichen.
- **Anpassbare Benutzeroberfläche:** Die Benutzeroberfläche von VS Code ist hochgradig anpassbar. Benutzer können Layout, Icons, Themes und Tastenkombinationen nach ihren Bedürfnissen anpassen.
- **Integriertes Terminal:** VS Code hat ein integriertes Terminal, das es ermöglicht, Shell-Befehle innerhalb der Entwicklungsumgebung auszuführen.
- **Remote-Entwicklung:** Entwickler können direkt auf Code zugreifen, der in Containern, auf Remote-Maschinen oder in der Cloud gehostet wird, und ihn bearbeiten.

Insgesamt ist VS Code für seine Schnelligkeit und Leichtgewichtigkeit bekannt und wird von zahlreichen Entwicklern gern eingesetzt.

## VS Code für die .NET-Entwicklung

Für die Einrichtung zum Entwickeln .NET-basierter Anwendungen sind die folgenden Tools erforderlich:

- VS Code
- C#-Dev-Kit-Extension
- .NET SDK

Für die Installation von VS Code wählen Sie die passende Installationsdatei für Ihr System (Bild 1). Die Download-Optionen finden Sie unter [2]. Ist VS Code auf Ihrem System bereits eingerichtet, dann veranlassen Sie zunächst ein Update. Danach geht es an die Installation der C#-Dev-Kit-Extension. Diese Erweiterung bietet die folgenden Features:

- Projektverwaltung und Solution Explorer
- Unterstützung für die Programmiersprache C#
- Paketmanagement
- Debugging-Funktionen
- Test-Unterstützung

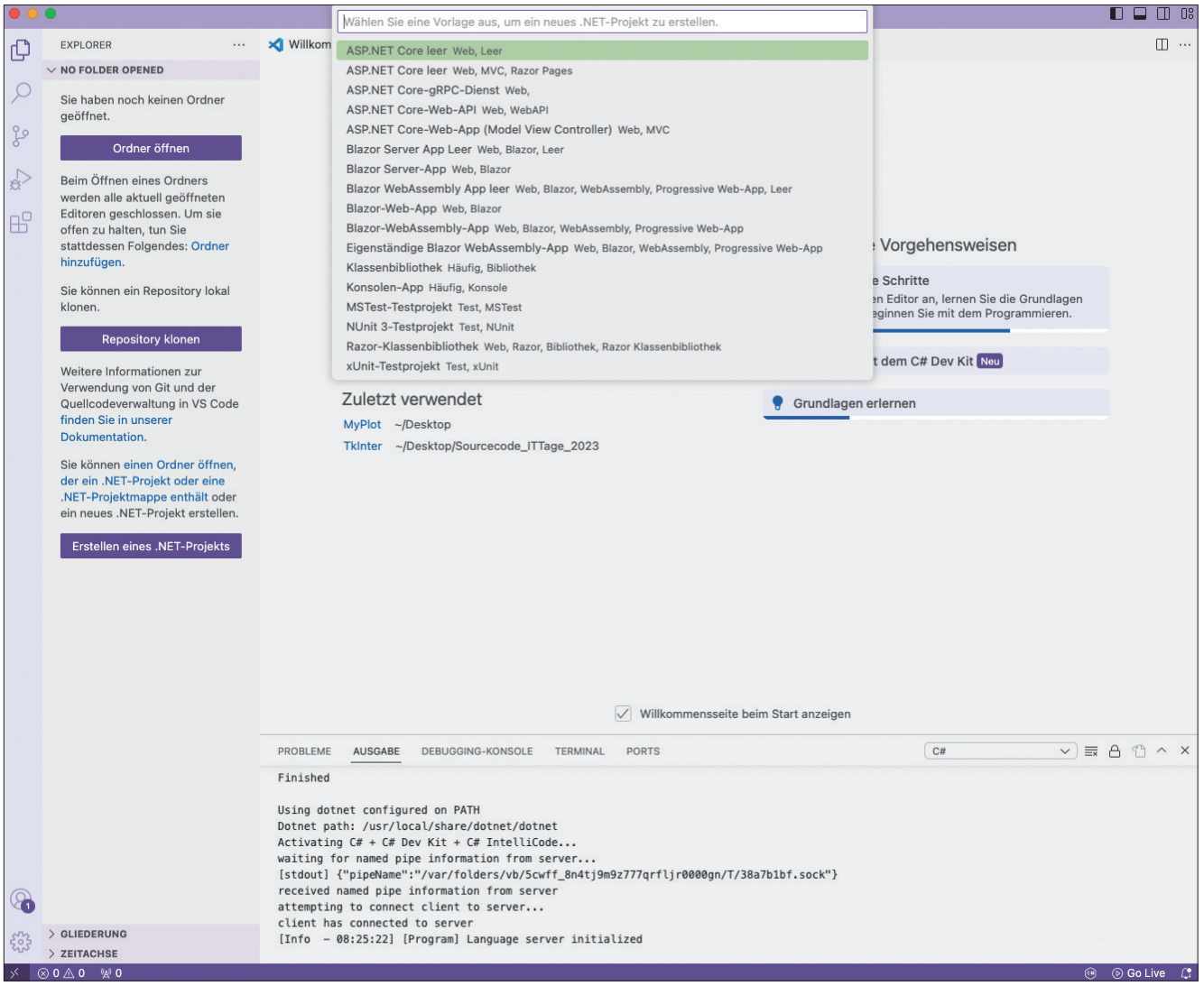
Welche Helferlein das Dev Kit außerdem noch mitbringt, lesen Sie im nebenstehenden Kasten Das Bundle C# Dev Kit.

Damit eine Entwicklung von .NET-Anwendungen möglich ist, ist das .NET SDK auf dem Zielsystem einzurichten [4]. Dieses steht auch für die unterschiedlichen Betriebssysteme und Versionen zur Verfügung, beispielsweise für macOS Intel und macOS ARM64. Nach dem Download starten Sie die Installation (Bild 2). ▶

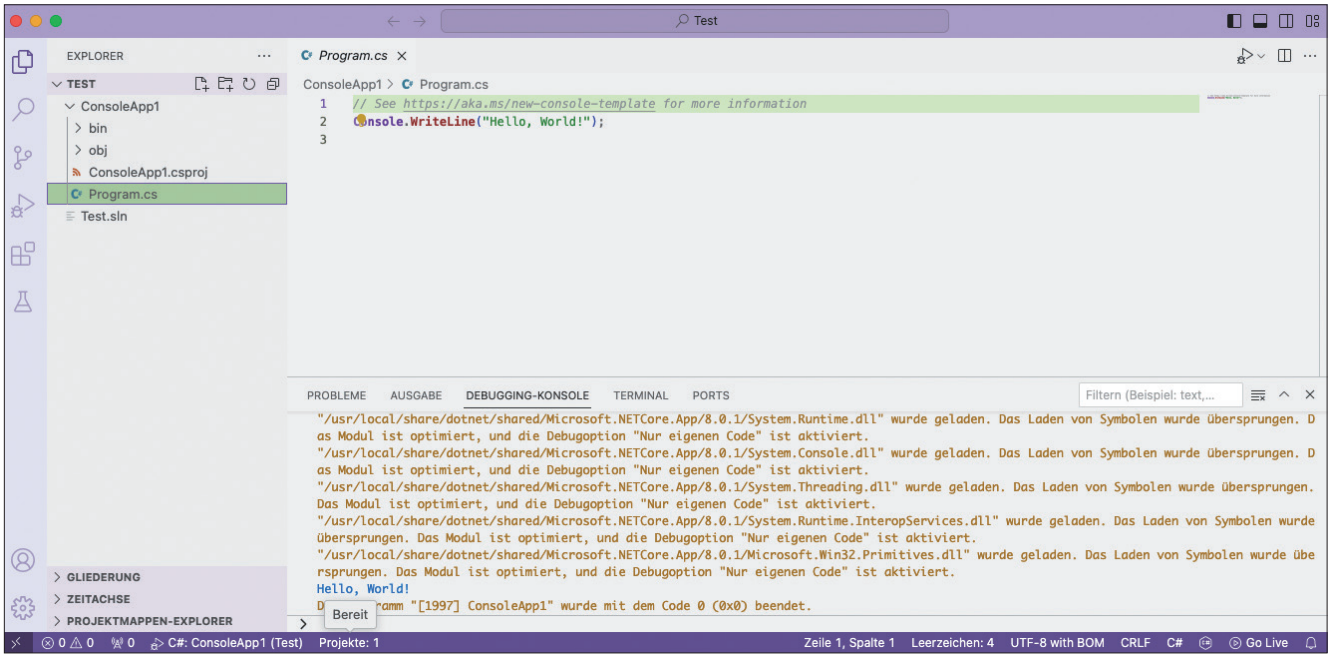
### Das Bundle C# Dev Kit

Diese Erweiterung installiert eine Reihe von weiteren Extensions in VS Code. Das sind:

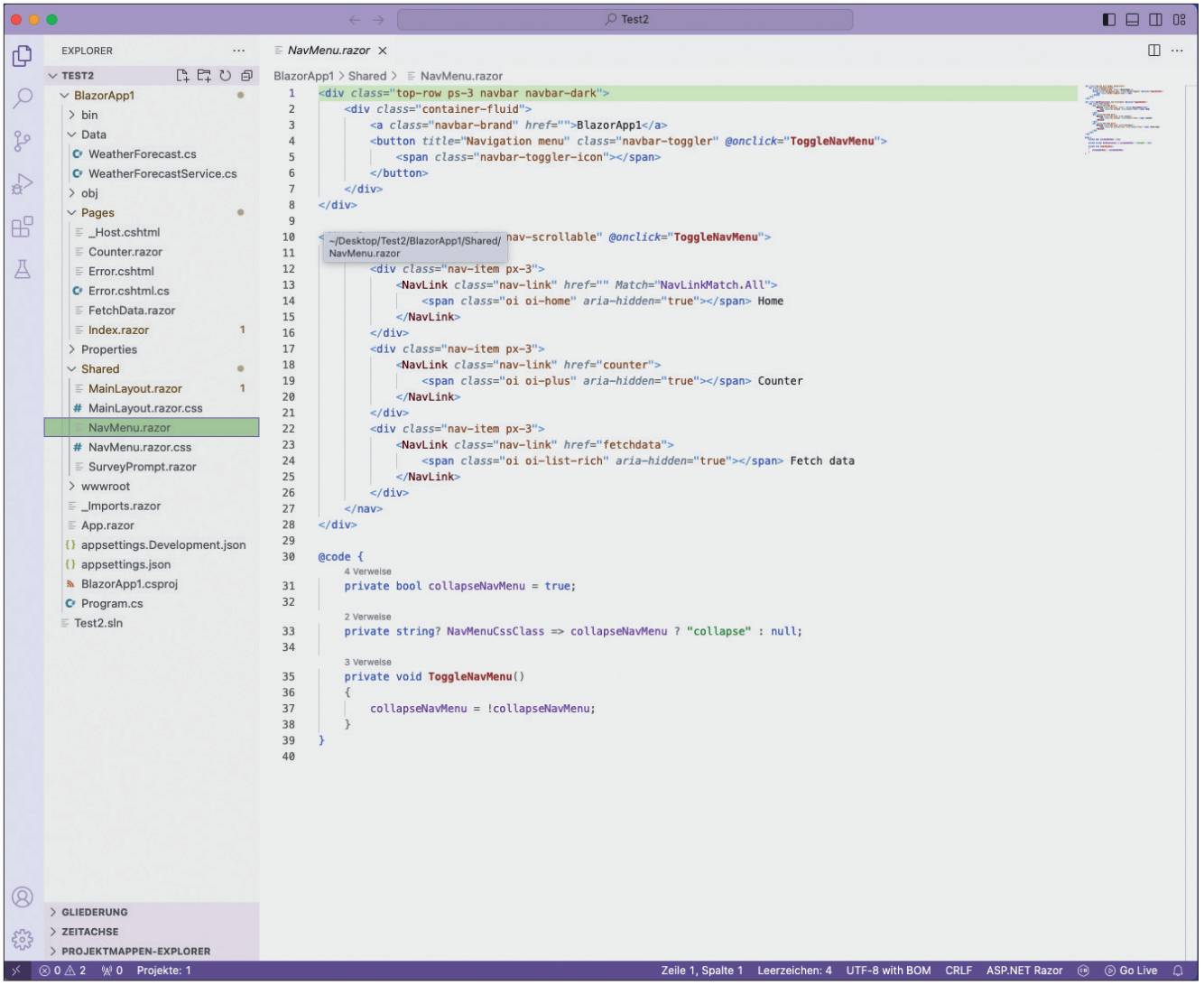
- **.NET Install Tool:** Teilt dem .NET-Installationstool mit, wenn ein .NET-SDK auf dem Computer installiert werden soll, und bietet dessen Einrichtung an.
- **C#:** Bietet eine umfangreiche Sprachunterstützung für C#. Die Erweiterung basiert auf einem LSP-Server (Language Server Protocol) und lässt sich in Open-Source-Komponenten wie Roslyn und Razor integrieren, um umfassende Typinformationen zu C# bereitzustellen.
- **IntelliCode for C# Dev Kit:** Bietet KI-gestützte Entwicklungsfunktionen für Python-, TypeScript/JavaScript-, C#- und Java-Entwickler in VS Code mit Erkenntnissen, die auf dem Verständnis ihres Code-Kontexts in Kombination mit maschinellem Lernen basieren. Wird ein grauer Text angezeigt wird, drücken Sie einfach die Tabulatortaste, um die Vorhersage zu akzeptieren, oder Sie können die Tastenkombination [Strg] + [Pfeil nach rechts] nutzen, um das erste Token/Wort der Vorhersage zu akzeptieren. Dazu ist der folgende Hinweis wichtig: Der gesamte Quellcode bleibt lokal, das heißt, das KI-Modell läuft direkt auf Ihrem Computer, sodass Sie für das individuelle Modelltraining keinen Code an einen Remote-Server übertragen müssen. Das führt zu einem reduzierten Speicherbedarf und einer verbesserten Inferenzgeschwindigkeit.
- **Weitere Funktionen:** Der Einsatz der Test-Frameworks xUnit, NUnit, MSTest und bUnit wird erkannt und organisiert, um eine schnelle Ausführung und Ergebnisnavigation zu ermöglichen. Die Erweiterung macht außerdem die Testbefehle der Befehlspalette von VS Code zum Debuggen und Ausführen von Tests verfügbar. Auch die Codevervollständigung wird durch eine KI-unterstützte Vervollständigung ganzer Zeilen und markierter Vorschläge basierend auf ihrer persönlichen Codebasis ergänzt und leistungsfähiger.




**Anlegen eines .NET-Projekts** in Visual Studio Code (Bild 3)




**.NET-Konsolen-Anwendung** als Projekt in Visual Studio Code (Bild 4)




**Eine Blazor Web-App** als Projekt in Visual Studio Code (Bild 5)

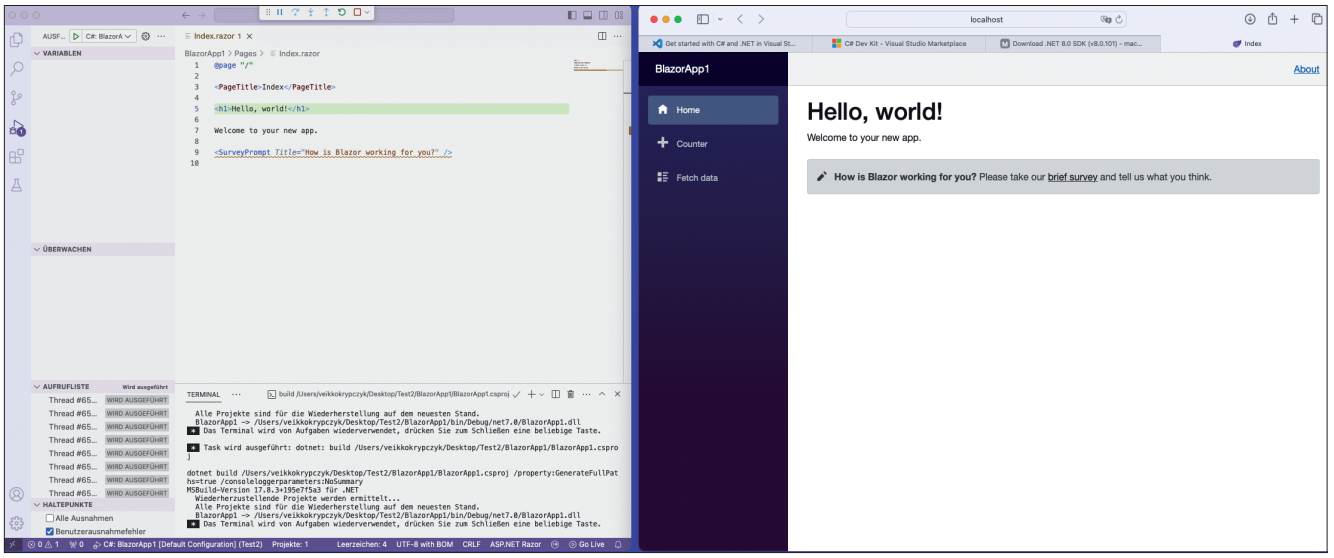
Für die vollständige Nutzung des C# Dev Kits ist eine Anmeldung mit einem Microsoft-Konto erforderlich, die man am besten gleich nach der Installation der Extension vornimmt. Nach der Installation der Extension besteht die Möglichkeit, .NET-Projekte mit Visual Studio Code zu erstellen. Klicken Sie auf den entsprechenden Menüpunkt, erhalten Sie die Auswahl der verfügbaren .NET-Projekttypen (Bild 3). Darunter befinden sich ASP.NET Core, Blazor- und Konsolen-Anwendungen.

Der erste Hallo-Welt-Test ist eine .NET-Konsolen-Applikation. Wählen Sie diesen Projekttyp aus der Liste der angebotenen Projekte aus, werden Sie gebeten, einen Speicherort und einen Projektnamen zu definieren. Auf Basis dieser Angaben generiert das .NET Command Line Interface (CLI) ein entsprechendes Projekt, inklusive Dateien (Bild 4). Die Konsolen-App kann direkt aus VS Code heraus gestartet werden. Klicken Sie auf das entsprechende Symbol, dann werden der Quellcode an den Compiler übermittelt und eine lauffähige Applikation generiert. In der Debugging-Konsole von VS Code sehen Sie die Ausgabe der Konsolen-App (hier: *Hello World*). Nach Installation der Extension steht in VS Code ein „nachgebauter" Projektmappen-Explorer zur Verfügung. Dieser bietet zu jeder Zeit einen Überblick über das .NET-Projekt. Beim Navigieren durch den Quellcode, bei typischen Refactoring-Maßnahmen und beim Schreiben des Quellcodes werden Entwickler durch eine Code-Vervollständigung unterstützt.

Im nächsten Test wird eine Blazor-App erstellt. Web-Applikationen dürften ein typischer Anwendungsfall für die Nutzung von VS Code außerhalb einer Windows-Umgebung sein. Auch in diesem Fall wird das Projekt-Setup durch das .NET CLI durchgeführt, und es werden wieder die erforder-

**Tabelle 1: .NET-MAUI-Apps mit VS Code entwickeln**

| Entwicklungssystem | Unterstützte Zielsysteme |
|---|---|
| Windows | Windows, Android |
| macOS | Android, iOS, macOS |
| Linux | Android |




**Entwickler-Layout für eine Web-App**, links: Quellcode-Editor, rechts: Browser (Bild 6)

lichen Ordner und Dateien generiert. Anschließend wird das Project in VS Code geöffnet (Bild 5). Auch diese Web-App soll direkt in VS Code kompiliert und im Debug-Modus gestartet werden. Schon mit der Installation wurde ein lokaler Webserver eingerichtet, sodass sich die Web-App direkt auf dem Entwicklungsrechner ausführen lässt. Ein typisches Layout des Arbeitsplatzes für die Webentwicklung sind das Nebeneinander von Quellcode und laufender Web-App – am besten auf einem großen oder auf zwei angeschlossenen Monitoren, siehe Bild 6.

Die Produktivität der Entwicklung kann gesteigert werden, wenn Änderungen im Quellcode ohne einen Neustart der App (Beenden, neu kompilieren, Neustart) unmittelbar übernommen werden. Diese Funktion wird als Hot Reload bezeichnet. In dieser Beziehung ist VS Code noch nicht so weit wie sein „großer Bruder" Visual Studio [5].

Hot Reload funktioniert in VS Code aktuell nur mit den folgenden Projekttypen:

- Console
- Test Projects
- Class Library Projects
- ASP.NET Core (nur *.cs*-Dateien)

Mit .NET-MAUI- oder Unity-Projekten funktioniert diese hilfreiche Funktion noch nicht.

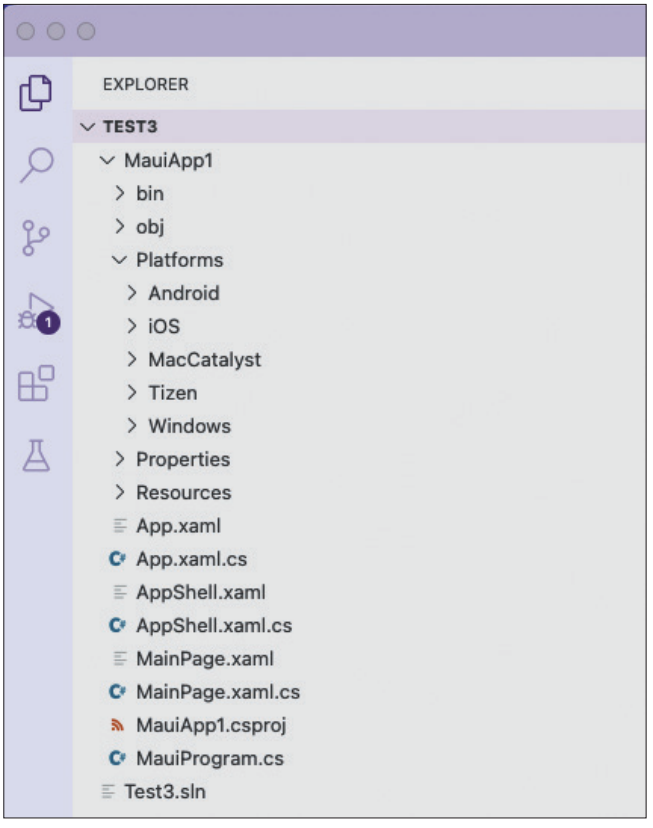
## .NET-MAUI-Support

Gerade für App-Entwickler, welche mit .NET MAUI arbeiten, könnte VS Code eine Option sein. Dafür ist es erforderlich, die *.NET MAUI extension for VS Code* [6] zu installieren. Sie bietet folgende Funktionen:

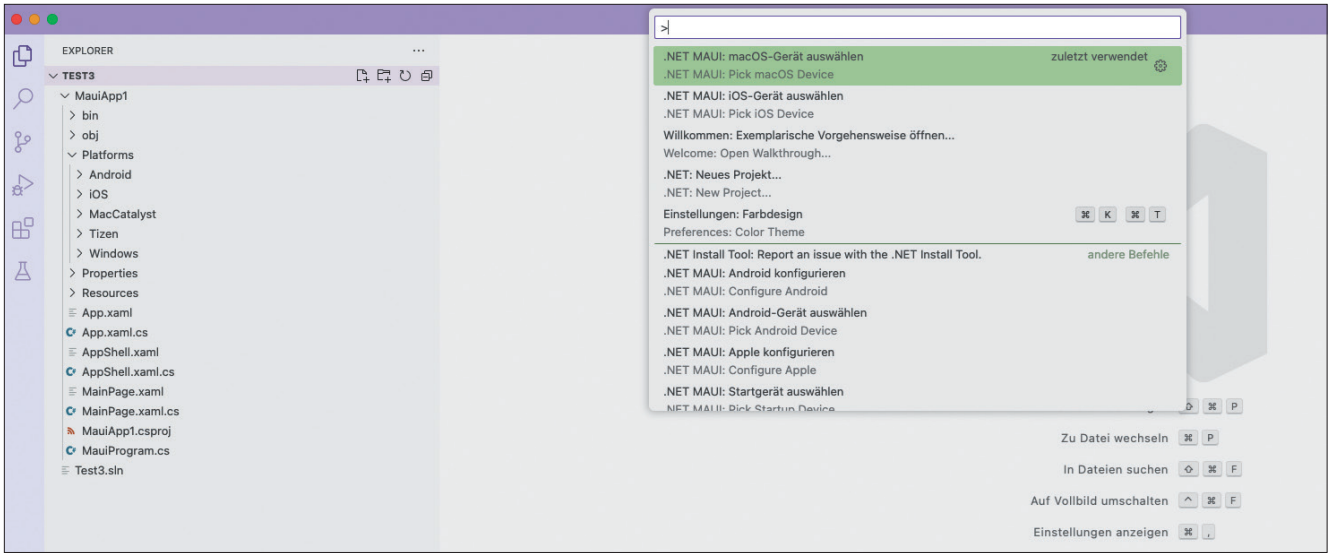
- Debugging auf Emulator, Simulator oder anderen unterstützten Geräten.
- Schnelles Ändern der Bereitstellungsziele.
- Nutzung aller Funktionen des C# Dev Kits (Solution Explorer, Test Explorer, Code-Navigation und Refactoring sowie Roslyn-basierte Sprachfunktionen) für .NET-MAUI-Projekte.
- Bearbeiten von XAML-Benutzeroberflächen mit einfacher Syntaxhervorhebung und Code-Vervollständigung.

In Bezug auf das Entwicklungssystem und die Zielsysteme gelten die Kombinationen gemäß Tabelle 1.

Installieren Sie diese Extension in VS Code. Im Test taucht danach unterhalb der .NET-Projekte leider noch keine Projektvorlage für .NET MAUI auf (Hinweis: Wir haben mit der aktuellen Version von Visual Studio Code auf einem Intel- ▶




**.NET-MAUI-Projekt** in VS Code (Bild 7)




**Auswahl des Zielsystems** in VS Code für ein .NET-MAUI-Projekt (Bild 8)

Mac-PC gearbeitet). Abhilfe schafft hier hoffentlich das manuelle Nachinstallieren von .NET MAUI über das VS-Code-Terminal mit erhöhten Rechten mithilfe der folgenden Befehlszeilenfolge:

```
sudo dotnet workload install maui
```

Nach der Bestätigung mit der Eingabetaste und der Passworteingabe werden die .NET-MAUI-Workloads für alle relevanten Zielsysteme (Android, iOS, macOS) eingerichtet. Es kann eine kurze Weile dauern, bis die neuen Projektvorlagen MAUI Blazor Hybrid, MAUI App und MAUI Klassenbibliothek bereitstehen.

Erstellen Sie nun eine MAUI-App, indem Sie einen leeren Ordner und einen Namen für das Projekt auswählen. VS Code generiert daraufhin ein Projektskelett (Bild 7). Nach dem Einrichten der Systemvoraussetzungen für Android, iOS und macOS und insbesondere der aktuellen, zu .NET 8 kompatiblen Xcode-Version können Sie die passende Konfiguration auswählen und die App auf Android, iOS und macOS ausführen (Bild 8).

Der Vorteil dieser Systemkonfiguration ist, dass man Apps für Android, iOS und macOS auf einem Rechner ohne jede Virtualisierung erstellen kann.

Die .NET-MAUI-Extension befindet sich noch in einer frühen Entwicklungsphase (kleiner Version 1). Deshalb muss man noch mit einigen Einschränkungen leben, beispielsweise:

- rudimentäre Unterstützung für XAML,
- noch kein Hot Reload (siehe oben) sowie
- kein Support für Xcode und iOS-Beta-Versionen.

Wünschenswert wäre zusätzlich ein größerer Funktionsumfang, ein etwas komfortablerer Installationsprozess aller Abhängigkeiten sowie etwas mehr projektspezifischer Support für .NET MAUI, insbesondere eine Hot-Reload-Funktion und eine umfassende XAML-Unterstützung (Data Binding und so weiter).

## Fazit

Mithilfe eines Satzes von Erweiterungen rüstet man nach und nach wichtige Funktionen in VS Code nach, um plattformübergreifend mit diesem Editor auch größere .NET-basierte Projekte und Apps bearbeiten zu können. Das dürfte insbesondere für .NET-MAUI-Apps und Webapplikationen auf der Basis von Blazor von Interesse sein.

Im Moment existiert ein erstes funktionierendes Toolset, welches noch nicht den Komfort von Visual Studio auf dem Mac beziehungsweise unter Windows bietet. Erlangen die Erweiterungen die notwendige Reife, gibt es keinen Grund mehr, VS Code nicht einzusetzen, denn dieser Editor ist leichtgewichtig, schnell installiert und kann mit weiteren Extensions an individuelle Wünsche angepasst werden. ■

*[1] Visual Studio für Mac wird eingestellt, www.dotnetpro.de/SL2406DevKit1*
*[2] Download VS Code, https://code.visualstudio.com*
*[3] C# Dev Kit for Visual Studio Code, www.dotnetpro.de/SL2406DevKit2*
*[4] Download .NET 8 SDK, www.dotnetpro.de/SL2406DevKit3*
*[5] Hot Reload mit dem C# Dev Kit, www.dotnetpro.de/SL2406DevKit4*
*[6] .NET MAUI extension for VS Code, www.dotnetpro.de/SL2406DevKit5*

**Dr. Veikko Krypczyk**
ist Softwareentwickler, Trainer und Fachautor und unter anderem auf WinUI 3 und .NET MAUI spezialisiert. Sein Wissen gibt er über Fachartikel, Seminare und Workshops weiter. Sie erreichen ihn unter **https://larinet.com**.

**dnpCode** A2406DevKit