Dr. Holger Schwichtenberg

Blazor 9.0

Moderne Webanwendungen und hybride Cross-Platform-Apps mit .NET 9.0, C# 13.0 und Visual Studio 2022

www.IT-Visions.de®
Dr. Holger Schwichtenberg

Buchversion:	9.0.0 vom 01.11.2024
Verlag:	www.IT-Visions.de, Fahrenberg 40b, D-45257 Essen
Sprachliche Korrektur:	Dorothea Fleischer, Heike Rickert, Matthias Bloch (M.A.)
ISBN:	978-3-934-27939-1
Bezugsquellen:	*www.IT-Visions.de/Buch/CS13*

Für Heidi, Maja und Felix

1 Inhaltsverzeichnis (Hauptkapitel)

2 Inhaltsverzeichnis (Details)

3 Vorwort

Liebe Leserinnen und Leser,

ich entwickele Webanwendungen seit Mitte der 1990er Jahre. Zunächst habe ich damals mit dem Common Gateway Interface (CGI) und Perl, danach mit dem Internet Database Connector (IDC) und Active Server Pages (ASP) sowie Visual Basic Webclasses programmiert. Nach einer Zwischenetappe in der Java-Welt mit Java Server Pages (JSP), Servlets und Applets folgte dann für mich ASP.NET in diversen Ausprägungen (Webforms, AJAX, Dynamic Data und MVC), schließlich Silverlight, danach ganz viel JavaScript/TypeScript (mit jQuery, AngularJS, Angular, vue.js, Svelte und diversen anderen Bibliotheken und Frameworks). Seit dem Jahr 2016 verwende ich in vielen Softwareentwicklungsprojekten ASP.NET Core und seit 2019 auch Blazor, während andere weiterhin JavaScript-basierte Frameworks verwenden.

Der Grund dieser Vorrede: Ich habe wirklich schon eine Menge Webframeworks gesehen und verspüre dennoch (oder auch gerade deswegen) einige Begeisterung für die "neue Sau", die durch das Webdorf getrieben wird; diese "Sau" namens Blazor.

Dieses Buch behandelt die Version **Blazor 9.0**, die im Rahmen von **.NET 9.0 und ASP.NET Core 9.0** läuft. Diese erste Ausgabe des Buchs ist kurz vor der RTM-Version erschienen auf Basis von Release Candidate 2 und wird kurz nach dem Release von .NET 9.0 in einer aktualisierten Version veröffentlicht.

Alle Ausführungen im Buch und den abgedruckten Beispielen gelten – sofern nicht ausdrücklich anders erwähnt – sowohl für **Blazor Web Apps (Blazor Static Server-Side-Rendering, Blazor Server, Blazor WebAssembly)** sowie **Blazor Hybrid (Blazor Desktop und Blazor Blazor MAUI)**. Es gibt nur wenige Unterschiede zwischen Architekturen, und auf die wird an entsprechender Stelle im Buch hingewiesen (Falls ich irgendwo einmal eine der drei Varianten in meinem Text vergessen habe sollte, zu erwähnen, fragen Sie gerne bei mir nach).

Dieses Buch richtet sich **sowohl an Neueinsteiger in Blazor 9.0 als auch Umsteiger von älteren Versionen**. Es wäre nicht wirtschaftlich, zwei getrennter Bücher dafür zu produzieren. Daher bitte ich die Leser, die neu einsteigen um Verständnis, dass es im Buch Hinweise die Änderungen zu früheren Versionen gibt.

Mit Blazor konnten wir schon einige Single-Page-Webanwendungen **in Rekordzeit entwickeln**, denn man ist als Entwickler damit sehr produktiv. Wenn man lange Erfahrung mit den verschiedenen Microsoft-Webframeworks einerseits und JavaScript-basierten Frameworks andererseits hat, ist man mit Blazor wirklich sehr schnell. Ich denke aber auch, dass Entwickler mit weniger Vorerfahrungen in Blazor eine gute Technik finden werden.

In den letzten Jahren haben wir bei **www.IT-Visions.de** bei unseren Kunden im ganzen deutschsprachigen Raum immer mehr klassische Windows-Desktop-Entwickler, die bisher mit Windows Forms oder WPF gearbeitet haben, auf Webtechniken umgeschult; getrieben von Chefs oder externen Beratern, die eine "Alles ins Web"- und "Cross-Platform"-Strategie verkündet haben. Mangels technischer Alternativen zur Entwicklung von Single-Page-Web-Applications (SPAs) war die Technikentscheidung für den Browser immer JavaScript bzw. TypeScript, mit Angular, React, vue.js oder anderen Bibliotheken. Auf dem Server konnten die Entwickler oft in der .NET-Welt bleiben mit ASP.NET Core WebAPI. Manchmal stand aber auch hier JavaScript via node.js auf dem Plan. Es waren nicht wenige .NET-Entwickler, die auch nach intensiver Motivation und Schulung keinerlei Gefallen an der JavaScript-Welt finden konnten.

Blazor ist die **Hoffnung für alle JavaScript-Hasser** – wenngleich man in der Praxis mit Blazor noch nicht ohne etwas JavaScript auskommt. Aber JavaScript ist wieder – wie früher – nur gelegentlich eingestreut und nicht mehr das Zentrum der Webanwendung.

Natürlich gibt es auch bei Blazor noch einige Hürden zu überwinden, wie immer bei den ersten Versionen einer neuen Technik.

Ich plane, in Zukunft weitere Versionen dieses Buchs zu veröffentlichen mit weiteren Aspekten zum Thema Blazor und auch zu den Basistechniken aus ASP.NET Core und .NET. Auch die Weiterentwicklungen von Blazor werde ich mit diesem Buch dokumentieren.

Das Buch **richtet sich sowohl an Neueinsteiger in Blazor** als auch **Umsteiger von älteren Blazor-Versionen**, denn es gibt immer Hinweise darauf, ab welcher Version einzelne Features verfügbar sind.

Das Buch setzt aber voraus, dass Sie .**NET, C# und Visual Studio schon kennen**. Ich biete dazu aber auch weitere Bücher an, siehe Kapitel "Notwendige Vorkenntnisse".

Dieses Fachbuch wird vertrieben auf folgenden Wegen (Ich nenne neben dem Verkaufspreis auch, wie viel – bzw. wenig – ich als Autor von den jeweiligen Händlern erhalte. Der Rest ist Gewinn der Händler):

- Gedruckt bei Amazon.de für 69,99 Euro (der Autor erhält 25,30 Euro):
  www.amazon.de/exec/obidos/ASIN/3934279392/itvisions-21
- Kindle-E-Book bei Amazon.de für 59,99 Euro (der Autor erhält 19,62 Euro):
  www.amazon.de/exec/obidos/ASIN/B0CLKZJKTX/itvisions-21
- PDF **inkl. Updates** bei Leanpub.com ab 49,99 Dollar (der Autor erhält 39,99 Dollar):
  www.leanpub.com/Blazor90
- Als Teil des E-Book-Buch-Abos zusammen mit anderen aktuellen Fachbüchern ab 99,00 Euro im Jahr inkl. aller Updates (der Autor erhält den kompletten Preis):
  www.IT-Visions.de/BuchAbo

> **Tipp:** Käufer bei Leanpub.com können jederzeit Aktualisierungen des PDF-Buchs (gleiche Hauptversion) kostenfrei dort beziehen. Käufer bei Amazon erhalten die PDF-Ausgabe einmalig kostenfrei (siehe Kapitel "Über dieses Fachbuch"). E-Book-Abonnenten haben jederzeit Zugriff auf alle aktuellsten Ausgaben der Fachbücher von Dr. Holger Schwichtenberg.

Da diese Preise in Anbetracht der vielen Stunden Arbeit an diesem Werk leider nicht nennenswert dazu beitragen können, den Lebensunterhalt meiner Familie zu bestreiten, ist dieses Projekt ein Hobby. Dementsprechend kann ich nicht garantieren, wann es Updates zu diesem Buch geben wird. Ich werde dann an diesem Buch arbeiten, wenn ich neben meinem Beruf als Softwarearchitekt, Berater und Dozent und meinen sportlichen Betätigungen noch etwas Zeit für das Fachbuchautorenhobby übrig habe.

Falls mir in diesem Buch oder den zugehörigen Downloads menschliche Fehler passiert sind, möchte ich mich dafür schon jetzt in aller Form bei Ihnen entschuldigen. Bitte geben Sie mir einen freundlichen, genau beschriebenen Hinweis auf meine Fehler. Ich freue mich immer über konstruktives Feedback und Verbesserungsvorschläge. Bitte verwenden Sie dazu dieses Kontaktformular:
www.dotnet-doktor.de/Leserfeedback

> **Tipp:** Ich belohne Sie mit E-Books für gemeldete Fehler, siehe Kapitel "Über dieses Fachbuch/Ihre Belohnung, wenn Sie helfen, dieses Buch zu verbessern".

Ich helfe Ihnen gerne, Ihren eigenen Programmcode zu schreiben bzw. Fehler in Ihrem Code zu finden, aber ich hoffe, Sie verstehen, dass ich dies nicht ehrenamtlich tun kann. Wenn Sie **technische Hilfe** zu Blazor, .NET und JavaScript oder anderen Themen rund um die Entwicklung und den Betrieb von Anwendungen (Desktop, Web und Mobile) sowie Server und Cloud benötigen, stehe ich Ihnen im Rahmen meiner beruflichen Tätigkeit für die Firma *www.IT-Visions.de* (Beratung, Schulung, Support, Softwareentwicklung) gerne zur Verfügung. Bitte wenden Sie sich für ein Angebot an das jeweilige Kundenteam. Bitte kontaktieren Sie die Firmen aber nicht für Feedback und Verbesserungsvorschläge zu diesem Buch, da dieses Buch meine Privatsache ist.

Die Beispiele zu diesem Buch können Sie herunterladen auf der von mir ehrenamtlich betriebenen **Leser-Website** unter: *www.dotnet-doktor.de/Leser*

Dort müssen Sie sich zunächst registrieren. Bei der Registrierung wird ein Losungswort abgefragt. Bitte geben Sie dort **LostInSpace** ein (siehe auch Kapitel "Programmcodebeispiele zum Download").

Herzliche Grüße aus Essen, dem Herzen der Metropole Ruhrgebiet

Holger Schwichtenberg

4 Über den Autor

- Studienabschluss Diplom-Wirtschaftsinformatik an der Universität Essen
- Promotion an der Universität Essen im Fachgebiet komponentenbasierter Softwareentwicklung
- Seit 1996 in der IT tätig als Softwareentwickler, Softwarearchitekt, Berater, Dozent und Fachjournalist
- Fachlicher Leiter des Expertenteams bei *www.IT-Visions.de* in Essen
- Über 95 Fachbücher bei verschiedenen Verlagen, u.a. Carl Hanser Verlag, O'Reilly, APress, Microsoft Press und Addison Wesley sowie im Selbstverlag
- Mehr als 1500 Beiträge in Fachzeitschriften und Online-Portalen
- Gutachter in den Wettbewerbsverfahren der EU vs. Microsoft (2006-2009)
- Ständiger Mitarbeiter der Zeitschriften iX (seit 1999), dotnetpro (seit 2000) und Windows Developer (seit 2010) sowie beim Online-Portal heise.de (seit 2008)
- Regelmäßiger Sprecher auf nationalen und internationalen Fachkonferenzen (z.B. BASTA!, Developer Week, .NET Developer Conference, MD DevDays, Microsoft TechEd, Microsoft Summit, Microsoft IT Forum, OOP, .NET Architecture Camp, IT Tage, enterJS, Advanced Developers Conference, DOTNET Cologne, iterate=>ruhr, Community in Motion, DOTNET-Konferenz, VS One, NRW.Conf, Windows Forum, Container Conf)
- Auszeichnungen und Zertifikate von Microsoft:
  - Microsoft Most Valuable Professional (MVP), ununterbrochen ausgezeichnet seit 2004
  - Microsoft Certified Solution Developer (MCSD)
- Thematische Schwerpunkte:
  - Softwarearchitektur, mehrschichtige Softwareentwicklung, Softwarekomponenten
  - Visual Studio, Continous Integration (CI) und Continous Delivery (CD) mit Azure DevOps
  - Microsoft .NET (.NET Framework, .NET Core, modernes .NET), C#, Visual Basic
  - .NET-Architektur, Auswahl von .NET-Techniken
  - Einführung von .NET, Migration auf .NET
  - Webanwendungsentwicklung und Cross-Plattform-Anwendungen mit HTML/CSS, JavaScript/TypeScript und C# sowie Webframeworks wie Angular, Vue.js, Svelte, ASP.NET (Core) und Blazor
  - Verteilte Systeme/Webservices mit .NET, insbesondere WebAPI, gRPC und WCF/CoreWCF
  - Relationale Datenbanken, XML, Datenzugriffsstrategien
  - Objekt-Relationales Mapping (ORM), insbesondere ADO.NET Entity Framework und Entity Framework Core
  - PowerShell
  - Architektur- und Code-Reviews
  - Performance-Analysen und -Optimierung
  - Entwicklungsrichtlinien

- Ehrenamtliche Community-Tätigkeiten:
  - Vortragender für die International .NET Association (INETA) und .NET Foundation
  - Betrieb diverser Community-Websites:
    www.dotnet-lexikon.de
    www.dotnetframework.de
    www.windows-scripting.de
    www.aspnetdev.de
    u.a.
- Firmenwebsite: *www.IT-Visions.de*
- Weblog: *www.dotnet-doktor.de*
- Kontakt für Anfragen zu Schulung und Beratung sowie Softwareentwicklungsprojekten:
  E-Mail *kundenteam@IT-Visions.de*
  Telefon *0201 / 64 95 90 – 50*
- Kontakt für Feedback zu diesem Buch:
  www.dotnet-doktor.de/Leserfeedback

5 Über dieses Fachbuch

5.1 Versionsgeschichte dieses Buchs

Die Versionsgeschichte dieses Buch finden Sie in einem eigenen Kapitel am Ende des Buchs.

Hinweis: Die Versionsgeschichte ist eine wichtige Referenz für die Leser, die sich aktuelle Versionen des Buchs beschaffen (z.B. über Leanpub.com) und wissen wollen, was sich geändert hat. Wenn Sie das Buch erstmalig lesen, müssen Sie die Versionsgeschichte nicht lesen.

5.2 Hinweis zu den Vertriebswegen

Dieses Fachbuch wird vertrieben auf folgenden Wegen (Ich nenne neben dem Verkaufspreis auch, wie viel – bzw. wenig – ich als Autor von den jeweiligen Händlern erhalte. Der Rest ist Gewinn der Händler):

- Gedruckt bei Amazon.de für 69,99 Euro (der Autor erhält 25,30 Euro):
  www.amazon.de/exec/obidos/ASIN/3934279392/itvisions-21
- Kindle-E-Book bei Amazon.de für 59,99 Euro (der Autor erhält 19,62 Euro):
  www.amazon.de/exec/obidos/ASIN/B0CLKZJKTX/itvisions-21
- PDF-E-Book **inkl. aller Buch-Updates** bei Leanpub.com ab 49,99 Dollar (der Autor erhält 39,99 Dollar):
  www.leanpub.com/Blazor90
- Als Teil des E-Book-Buch-Abos zusammen mit anderen aktuellen Fachbüchern ab 99,00 Euro im Jahr **inkl. aller Buch-Updates** (der Autor erhält den kompletten Preis):
  www.IT-Visions.de/BuchAbo

Hinweise: Ich habe mich für den Vertriebsweg des gedruckten Buchs über Amazon entschieden, weil ich dort ständig Updates zu dem Buch einreichen kann. Per Print-on-Demand erhalten Leser dann immer das topaktuelle Buch. Oft liefert Amazon dennoch am Tag nach der Bestellung das Buch schon aus. Der Vertrieb dieses Buch über klassische IT-Verlage, die leider heutzutage immer noch größere Auflagen vorproduzieren, ist für ein sehr agiles Softwareprodukt wie Blazor keine Alternative mehr.

Ich nenne dabei auch den Erlös für den Autor, weil ich sehr häufig Leser treffe, die fälschlicherweise denken, der wesentliche Teil des Buchpreises komme dem Autor zu Gute. Das ist leider nicht so, außer bei Leanpub.com oder eigenen Vertriebswegen wie meinem Buchabo. Daher denke ich, dass es sinnvoll ist, dies transparent zu machen.

Tipp: Käufer bei Leanpub.com können jederzeit Aktualisierungen des PDF-Buchs (gleiche Hauptversion) kostenfrei dort beziehen. Käufer bei Amazon erhalten die PDF-Ausgabe einmalig kostenfrei (siehe Kapitel "Bezugsquelle des PDF-E-Books"). E-Book-Abonnenten haben jederzeit Zugriff auf alle aktuellsten Ausgaben der Fachbücher von Dr. Holger Schwichtenberg

5.3 Bezugsquelle des PDF-E-Books für Amazon-Kunden

Wenn Sie dieses Buch in gedruckter Form oder als Kindle-Ausgabe bei Amazon erworben haben, können Sie zusätzlich eine PDF-Version des Buchs **kostenfrei** erhalten.

Leiten Sie dazu Ihren Kaufbeleg von Amazon an folgende E-Mail-Adresse weiter:

PDFBuchZugabe@dotnet-doktor.de

Geben Sie dabei bitte Vorname, Name, Firma und E-Mail-Adresse an.

Sie erhalten dann binnen 1-2 Wochen das auf Sie personalisierte PDF-Dokument. Dieses Angebot gilt innerhalb von 6 Monaten nach dem Kauf des Buchs bei Amazon.

5.4 Bezugsquelle für Aktualisierungen

Leanpub-Kunden können jederzeit Aktualisierungen des PDF-Buchs (gleiche Hauptversion!) kostenfrei bei Leanpub.com beziehen.

Käufer der Kindle- oder Druck-Version können die sich ständig aktualisierende PDF-Version zum Preis von 19,99 Dollar (zzgl. 7% Mehrwertsteuer) unter folgender Webadresse beziehen:

https://leanpub.com/Blazor90/c/LostInSpace

Hinweise: Leider erlauben Amazon u.a. Buchhändler aufgrund der Buchpreisbindungsgesetze in Deutschland den Autoren grundsätzlich nicht, dass Leser eine Aktualisierung im Kindle-Format oder in gedruckter Form vergünstigt erhalten können.

Bitte beachten Sie auch, dass die ISBN-Regularien erfordern, dass man bei einer Titeländerung bei neuer Produktversion eine neue ISBN vergeben und damit auch ein neues Buchprojekt bei Amazon und Leanpub erstellt werden muss.

5.5 Hinweise zur Breite und Tiefe dieses Buchs – Sie haben Einfluss!

Ein Fachbuch, das ein riesengroßes Themengebiet wie Blazor behandelt, kann nicht jedes Teilgebiet und jeden Aspekt der Technik behandeln, zumindest nicht in gleicher Tiefe. Dann würde solch ein Fachbuch über eintausend Seiten, in einigen Fällen sogar mehrere tausend Seiten umfassen.

Ich denke, dass ich nach aktuellem Stand der Technik und meinem Wissenstand etwa 2.500 Buchseiten zu Blazor schreiben könnte. Würden Sie so ein dickes (und entsprechend teures) Buch kaufen und lesen wollen?

Wie jeder Fachautor lese auch ich immer wieder Kritik, dass ein Leser ein bestimmtes Thema nicht oder nicht in ausreichender Tiefe behandelt sei in dem Buch. Das ist aus der Sicht des einzelnen Lesers sicherlich gerechtfertigt, aber wie jeder Fachautor muss ich eben zwingend eine Auswahl der Themen treffen. Gerne dokumentiere ich hier, wie ich persönlich diese Auswahl für meine Bücher treffe:

- Ich behandele im Buch die Themen, die wir in unserer Firma selbst in der Praxis in den Softwareentwicklungsprojekten brauchen.
- Ich behandele zusätzlich die Themen, die unsere Kunden in Schulungen behandelt haben möchten oder in Beratungsgesprächen ansprechen.
- Ich behandele Themen, die ich spannend finde.

Folglich sind die Themen, die ich im Buch nicht oder nur kurz behandele für uns und unsere Kunden nicht relevant bzw. so selbsterklärend, dass es keine Fragen dazu gibt.

Natürlich kann das für Sie anders sein. Sie können mir immer gerne schreiben, wenn Sie ein Thema im Buch behandelt haben möchten. Ich sammele diese Anregungen und wenn es mehrere Zuschriften zu einem Thema gibt, dann kommt das Thema weit oben auf die Prioritätenliste. Ich denke, das ist ein faires Verfahren.

5.6 Geplante Kapitel

Die Reihenfolge der für folgende Buchversionen geplanten Kapitel ist hier zunächst alphabetisch angeordnet und entspricht nicht der Reihenfolge, in der die Kapitel erscheinen werden.

- Autorisierung mit Rollen
- Anpassbarer Ladevorgang bei Blazor WebAsssembly (seit Blazor 6.0)
- Custom Event Arguments (seit Blazor 6.0)
- DotNet.createJSObjectReference()

- Eigene Basisklassen für Razor Components mit @inherits
- Datei-Upload bei Blazor WebAssembly und Blazor MAUI via Webservice (ist bereits im Code von MiracleList_BW enthalten, aber im Buch noch nicht beschrieben)
- IComponentActivator zur Kontrolle der Komponenteninstanziierung (seit Blazor 5.0)
- Installation von Blazor-Anwendungen auf Linux-Servern
- Integration von Blazor in JavaScript-Anwendungen (seit Blazor 6.0)
- Inkrementelle Migration auf Blazor mit .NET Upgrade Assistant und Microsoft.AspNetCore.SystemWebAdapters
- IL-Linker-Konfigurationsdatei (LinkerConfig.xml)
- JavaScript-Interop mit IJSInProcessRuntime als Alternative zu der im Buch behandelten IJSRuntime (in Blazor WebAssembly)
- JavaScript beim Static Server-Side-Rendering (SSR) mit <PageScript> (seit Blazor 8.0)
- Blazorise Components (kostenfrei) → *https://blazorise.com*
- Synchrone JavaScript-Interop mit IJSInProcessRuntime (in Blazor WebAssembly seit Blazor 5.0)
- LESS und SCSS mit Blazor
- NavigateToLogin() und NavigateToLogout() in NavigationManagerExtensions (seit Version 7.0)
- Open ID Connect-Authentifizierung: flexiblere Open ID Connect-Authentifizierung in Blazor WebAssembly (seit Version 7.0)
- PreserveWhiteSpace
- QR-Codes darstellen
- Razor-Rendering für Versand von HTML-E-Mail
- Routing mit @attribute [Route(Constants.CounterRoute)]
- Routing-Anpassungen (*https://chrissainty.com/building-a-custom-router-for-blazor/amp/)*
- SQLite in Blazor WebAsssembly (möglich seit Blazor 6.0)
- SQLite in Blazor Hybrid
- Telerik REPL for Blazor *https://blazorrepl.telerik.com*
- Testen von Blazor-Anwendungen mit Razor-Komponententests über bUnit alias Razor.Components.Testing.Library → **Beispiele zu bUnit finden Sie bereits im MiracleList-Beispiel, das Sie herunterladen können!**
- Testen von Blazor-Anwendungen mit End-To-End-GUI-Tests im Webbrowser über Microsoft Playwright → **Beispiele zu Playwright finden Sie bereits im MiracleList-Beispiel, das Sie herunterladen können!**
- TypeScript in Blazor Apps
- Validierung: Anpassung der CSS-Klassen (seit Blazor 5.0)
- Visual Studio Code als alternatives Werkzeug zu Visual Studio
- Web Crypto API / SubtleCrypto: Hashing-Algorithmen (seit .NET 7.0)
- Web Components Custom Elements mit RegisterAsCustomElement() (seit Blazor 6.0 im Preview, seit Blazor 7.0 einsatzreif)

Hinweis: Mit dieser Liste kommender Themen möchte ich gleichzeitig klarstellen, welche Themen Sie **derzeit** nicht im Buch finden, damit Sie als Leser keine falschen Erwartungen haben.

5.7 Programmiersprache in diesem Buch

Als Programmiersprache kommt in diesem Buch C# zum Einsatz, weil C# die einzige offiziell von Microsoft unterstützte Programmiersprache für Blazor ist.

Als zweite Programmiersprache für Blazor gibt es nur F# über eine Erweiterung [*https://github.com/fsbolero/Bolero*], die außerhalb von Microsoft entwickelt wird und daher kein offizielles Produkt von Microsoft ist.

5.8 Notwendige Vorkenntnisse

Webprogrammierung ist ein großes, komplexes Gebiet. In diesem Buch konzentriere ich mich auf ASP.NET Blazor im engeren Sinne.

Als Vorkenntnisse sollten Sie zumindest ein gutes Grundlagenwissen in folgenden Gebieten mitbringen:

- HTTP
- HTML
- CSS
- JavaScript
- .NET Core
- ASP.NET Core
- C#
- Entity Framework Core
- Visual Studio

Ich habe drei Bücher geschrieben, durch die Sie dieses Grundlagenwissen erlangen können.

Für die ersten fünf Punkte ist dies:

Moderne Webanwendungen für .NET-Entwickler: Server-Anwendungen, Web APIs, SPAs & HTML-Cross-Platform-Anwendungen mit ASP.NET, ASP.NET Core, JavaScript/TypeScript und Angular

ISBN 3960090153

https://www.amazon.de/exec/obidos/ASIN/3960090153/itvisions21

- Gedruckt: 49,90 Euro
- Kindle: 39,99 Euro

Wenn Sie C# lernen wollen, möchte ich Ihnen mein Buch "C# Crashkurs" empfehlen:

C# 13.0 Crashkurs

- PDF-E-Book bei Leanpub.com ab 19,99 Dollar
- gedruckt (Print-on-Demand) bei Amazon.de für 39,99 Euro
- Kindle-E-Book bei Amazon.de für 29,99 Euro
- *www.IT-Visions.de/CSharpBuch*

Um Entity Framework Core zu lernen, möchte ich Ihnen mein Buch "Moderne Datenzugriffslösungen mit Entity Framework Core" empfehlen:

Moderne Datenzugriffslösungen mit Entity Framework Core 9.0

- Gedruckt bei Amazon.de für 79,99 Euro
- Kindle-E-Book bei Amazon.de für 69,99 Euro
- PDF-E-Book bei Leanpub.com für 49,99 Dollar
- *www.IT-Visions.de/EFCBuch*

5.9 Vermeidung des Einsatzes kostenpflichtiger Software

Der Autor dieses Buch weiß, dass viele, aber nicht alle Leser dieses Buchs professionelle Softwareentwickler sind und hat daher darauf geachtet, dass Sie keine Software kaufen müssen, um die Inhalte dieses Buchs nachzuvollziehen.

Blazor selbst als Teil von .NET kostenfrei. Bei Visual Studio können Sie zum Lernen die kostenfreie Community-Edition einsetzen. Für den kommerziellen Einsatz brauchen Sie in Unternehmen, die die von Microsoft jeweils definierten Umsatzgrenze überschreiten, eine kostenpflichtige Version.

Darüberhinaus verzichtet das Buch im Wesentlichen auf die Beschreibung kostenpflichtiger Produkte. Eine Ausnahme bildet das Kapitel "Komponentenbibliotheken von Drittanbietern". **Das MiracleList-Fallbeispiel in diesem Buch ist komplett ohne kommerzielle Komponenten implementiert.**

5.10 Hinweis zur Erwähnung der Blazor-Varianten in diesem Buch

Vor .NET 6.0 gab es nur zwei Blazor-Arten: Blazor Server und Blazor WebAssembly. Mit .NET 6.0 erschien eine dritte Blazor-Art: Blazor Hybrid, die man auch noch mal differenzieren kann in hybride Apps in WPF und Windows Forms (Blazor Desktop) und in .NET MAUI (Blazor MAUI). Seit Blazor 8.0 gibt es auch noch Blazor Static Server-Side-Rendering (SSR) und Blazor United, wobei letzteres kein offizieller Begriff ist, sondern von Microsoft nur zu Beginn von .NET 8.0 verwendet wurde, um die Integration und den nahtlosen Übergang von Blazor Server und Blazor WebAssembly zu beschreiben.

Es wird im Buch vorerst noch einige Stellen geben, wo ich nicht alle Blazor-Arten explizit erwähne. Es ist mir als Hobby-Autor aus zeitlichen Gründen vorerst nicht möglich, das ganze Buch nach diesen Stellen zu durchkämmen. Was per Volltextsuche auffindbar war, habe ich erledigt.

Bitte: Geben Sie mir gerne einen freundlichen Hinweis auf Stellen, wo eine Erwähnung von Blazor MAUI, Blazor Desktop, Blazor Static SSR oder Blazor United fehlt, siehe nächstes Unterkapitel.

5.11 Ihre Belohnung, wenn Sie helfen, dieses Buch zu verbessern!

Wenn Sie Fehler in diesem Buch finden, bin ich Ihnen nicht nur wirklich sehr dankbar, sondern Sie bekommen auch eine Belohnung in Form von aktualisierten oder weiteren E-Books.

Fehlerart	E-Book-Guthaben
Inhaltlicher Fehler	Pro Fehler 5 Euro
Sprachlicher Fehler	Pro Fehler 2 Euro

Ein Beispiel: Wenn Sie zwei inhaltliche Fehler und zehn Rechtschreibfehler in diesem Buch finden, dann haben Sie bei mir 30 Euro gut. Dafür können Sie dann eins meiner selbstverlegten Bücher als E-Book bekommen.

Die selbstverlegten Bücher finden Sie unter *www.IT-Visions.de/Verlag*

Melden Sie die Fehler unter *www.dotnet-doktor.de/Leserfeedback*

Schreiben Sie dabei, welches E-Book Sie wünschen. Das Buch schicke ich Ihnen dann per E-Mail zu.

Tipp: Auch Fehler auf meiner persönlichen Website *www.dotnet-doktor.de* und der Firmenwebsite *www.IT-Visions.de* zählen mit!

Die Fehlermeldung zählt nur, wenn nicht ein anderer Leser dies bereits gemeldet hat und es daher in der aktuellen Auflage schon korrigiert ist.

Ich freue mich auf Ihre Fehlermeldung!

6 Programmcodebeispiel zum Download

Die Beispiele zu diesem Buch können Sie als Visual Studio-Projekte herunterladen.

Hinweis: Bitte beachten Sie, dass nicht jede einzelne Zeile Programmcode, die Sie in diesem Buch finden, in den herunterladbaren Projekten enthalten sein kann. Die Projekte bilden funktionierende Lösungen. In diesem Buch werden auch alternative Lösungen für Einzelfälle diskutiert, die nicht unbedingt zu einer Gesamtlösung passen.

6.1 Webadresse für Downloads

Den Download der Beispiele zu diesem Buch finden Sie auf der Leser-Website unter der Webadresse:

www.dotnet-doktor.de/Leser

Dort müssen Sie sich bitte einmalig registrieren. Bei der Registrierung wird ein **Losungswort** abgefragt, das Sie als Käufer dieses Buchs ausweist. Bitte geben Sie dort **LostInSpace** ein.

Durch die Registrierung erhalten Sie dann ein persönliches **Kennwort** per E-Mail zugesendet, das Sie danach immer wieder für die Anmeldung zum Leserportal nutzen können.

Bei Problemen mit der Registrierung nutzen Sie bitte das auf der o.g. Webseite verlinkte FAQ.

6.2 Übersicht über die Beispiele

Das Downloadpaket zu diesem Buch besteht aus mehreren Ordnern (siehe Abbildung).

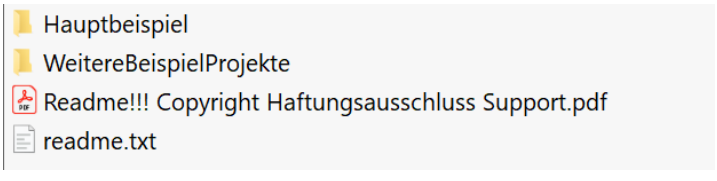


Abbildung: Inhalt des Download-Pakets zu diesem Buch

Die folgende Tabelle zeigt, wo Sie in dem herunterladbaren ZIP-Paket die Beispiele der einzelnen Kapitel des Fachbuchs finden.

Es war eine bewusste Entscheidung, in diesem Buch nicht für jedes einzelne Kapitel ein eigenes Blazor-Projekt zu schaffen, sondern die meisten Beispiele in ein echtes Fallbeispiel "MiracleList" zu integrieren.

Dies hat folgende Vorteile:

- Auf diese Weise sind komplexere, praxisnahe Beispiele möglich.
- Die Aktualisierung der Beispiele auf neue NuGet-Paket-Versionen und API-Änderungen geht schneller (dies ist wichtig in der heutigen Zeit, in der es ständig neue Versionen, auch mit Breaking Changes, gibt).

Ordner unter "Hauptbeispiel/"	Programmcode aus Kapitel(n)	Hinweise
/MiracleList_Blazor /src/MiracleList_BU	Alle	Blazor United-Beispiele einschließlich der Blazor United-Implementierung des Fallbeispiels "MiracleList" auf Basis der Vorlage "Blazor Web App". Die Anwendung läuft im Auto-Modus, wechselt also von Blazor Server zu Blazor WebAssembly.
/MiracleList_Blazor /src/MiracleList_BS	Alle	Blazor Server-Beispiele einschließlich der Blazor Server-Implementierung des Fallbeispiels "MiracleList". Dieses Projekt

Ordner unter "Hauptbeispiel/"	Programmcode aus Kapitel(n)	Hinweise
		basiert noch auf der alten Projektvorlage "Blazor Server", läuft aber unter der aktuellen Blazor-Version.
/MiracleList_Blazor /src/MiracleList_BW	Alle	Blazor WebAssembly-Beispiele einschließlich der Blazor WebAssembly-Implementierung des Fallbeispiels "MiracleList". Dieses Projekt basiert noch auf der alten Projektvorlage "Blazor WebAssembly", läuft aber unter der aktuellen Blazor-Version..
/MiracleList_Blazor /src/MiracleList_BD	Alle	Blazor WPF-Desktop-Beispiele einschließlich der Blazor Desktop-Implementierung des Fallbeispiels "MiracleList"
/MiracleList_Blazor /src/MiracleList_BM	Alle	Blazor MAUI-Beispiele einschließlich der Blazor MAUI-Implementierung des Fallbeispiels "MiracleList"
/MiracleList_Blazor /src/MiracleListAPI_Proxy	Fallbeispiel	Generierte Proxy-Klasse für MiracleList-Backend-WebAPI, wird verwendet in MiracleList_BW und MiracleList_BM
/MiracleList_Blazor /src/ITVisions.Blazor	JavaScript-Interoperabilität Razor Class Library Fallbeispiel	Razor Class Library mit eigenen Blazor-Steuerelementen und Hilfscode für JavaScript-Interoperabilität; wird in allen vier Implementierung von MiracleList verwendet. Realisiert u.a. die vielfach in diesem Buch verwendete Hilfsklasse BlazorUtil.cs mit zugehöriger JavaScript-Datei BlazorUtil.js
/MiracleList_Blazor /src/MLBlazorRCL	Alle	Diese Razor Class Library stellt gemeinsam genutzte Inhalte (Klassen, Razor Components sowie statische Webelemente wie Grafiken und eine CSS-Datei) für das MiracleList-Fallbeispiel und ergänzende Beispiele aus diesem Buch bereit. Dieses Projekt wird in allen vier Implementierung von MiracleList verwendet.
/MiracleList_Blazor /src/SamplesRCL	Alle	Ergänzende Beispiele aus dem Buch, die nicht in das MiracleList-Szenario fallen. Dieses Projekt wird in allen vier Implementierung von MiracleList verwendet. Diese Beispiele sind im Menü "Blazor-Beispiele außerhalb der MiracleList" aufrufbar.
/MiracleList_Blazor /src/MiracleList_Backend	Fallbeispiel	Backend (ASP.NET Core WebAPI und ASP.NET Razor Pages)
/MiracleList_Blazor /Test/UnitTests	Bisher nicht im Buch behandelt	Automatisierte Tests (Unit Tests) der MiracleList-Geschäftslogik (BL), die wahlweise als isolierte Tests oder als

Ordner unter "Hauptbeispiel/"	Programmcode aus Kapitel(n)	Hinweise
		Integrationstests gegen die Datenbank laufen können
/MiracleList_Blazor /Test/WebserviceTests	Bisher nicht im Buch behandelt	Automatisierte Tests des MiracleList-Backend-Services (WebAPI und SignalR)
/MiracleList_Blazor /Test/BlazorTests	Bisher nicht im Buch behandelt	Automatisierte Tests der Blazor-Komponenten in MiracleList mit bUnit
/MiracleList_Blazor /Test/ SeleniumTests	Bisher nicht im Buch behandelt	Automatisierte End-to-End-GUI-Tests der MiracleList-Anwendung innerhalb von Webbrowsern mit Selenium
/MiracleList_Blazor /Test/ PlaywrightTests	Bisher nicht im Buch behandelt	Automatisierte End-to-End-GUI-Tests der MiracleList-Anwendung innerhalb von Webbrowsern mit Microsoft Playwright

Ordner unter "WeitereBeispiel Projekte"	Programmcode aus Kapitel(n)	Hinweise
/NET9_Demos	Allen Kapiteln, in denen es Neuerungen in Blazor 9.0 gibt	Beispiele zu neuen Funktionen in Blazor 9.0.
/NET8_Demos	Static Server-Side-Rendering und Tipps & Tricks/QuickGrid	Beispiel zu Blazor Static SSR und dem QuickGrid-Steuerelement, die inhaltlich nicht in das MiracleList-Beispiel passen.
/BlazorWebservices	Zugriff auf Webservices/WebAPIs	Die Projektmappe "BlazorWebservices" umfasst die Implementierung von WebAPIs und Google RPC-Diensten und den Aufruf aus Blazor-WebAssembly. Eigenständiges Beispiel, das die Datenbank der fiktiven Fluggesellschaft World Wide Wings (*www.world-wide-wings.de)* verwendet.
/WebformsToBlazor	Umstieg von Webforms auf Blazor	Eigenständiges Beispiel für den Umstieg von ASP.NET Webforms zu ASP.NET Core Blazor, das die Datenbank der fiktiven Fluggesellschaft World Wide Wings (*www.world-wide-wings.de)* verwendet.
/BlazorMicroApps	Integration von Blazor-Apps in eine Webforms-Anwendung	Eigenständiges Beispiel, das keine Datenbank verwendet, sondern auf den Standardprojektvorlagen mit Daten im RAM basiert.
/WebComponentsMitAngular_ITVElements	Interoperabilität zwischen .NET und JavaScript	TypeScript-Quellcode für die dort verwendeten Angular-basierten Komponenten

Ordner unter "WeitereBeispiel Projekte"	Programmcode aus Kapitel(n)	Hinweise
/WebAssemblyPrerendering	Tipps & Tricks/Render-Modi bei Blazor WebAssembly	Beispiel für das Prerendering einer Blazor WebAssembly-Anwendung

6.3 Eingesetztes CSS-Framework

Alle Beispiele verwenden die CSS-Klassen des CSS-Frameworks **Bootstrap** [*https://getbootstrap.com*] zur Formatierung. Dies ist natürlich nur eine von vielen Gestaltungsoptionen in Blazor. Sie können auch andere Design Systeme wie Tailwind, Bulma oder Microsoft Fluent UI einsetzen.

Beispiele für CSS-basierte Design-Systeme:

- Twitter Bootstrap [*https://getbootstrap.com*]
- Google Material Design (Angular Material *https://material.angular.io/*, Vuetify *https://vuetifyjs.com/en/*)
- Microsoft Fluent Design System / Fluent UI [*https://developer.microsoft.com/fluentui/*]
- Tailwind CSS [*https://tailwindcss.com/*]
- Ant Design [*https://ant.design/*]
- Bulma [*https://bulma.io/*]
- eFrolicss [*https://efrolic.github.io/css/*]
- Lion [*https://github.com/ing-bank/lion*]

6.4 Das Fallbeispiel "MiracleList"

Dieses Kapitel führt in das zentrale Blazor-Fallbeispiel "MiracleList" ein, das an vielen Stellen in diesem Buch verwendet wird.

6.4.1 Szenario

Das Fallbeispiel ist sehr praxisnah, weil das Anwendungsszenario der Nachbau einer existierenden, sehr erfolgreichen, Anwendung ist.

Im Jahr 2015 zahlte Microsoft für die Übernahme des Berliner App-Herstellers Wunderlist mehr als 100 Millionen US-Dollar [*www.heise.de/newsticker/meldung/Microsoft-uebernimmt-Berliner-Startup-6Wunderkinder-2678017.html*].

Abbildung: Das MiracleList-Logo

Hinweis: Mittlerweile hat Microsoft Wunderlist sterben lassen zugunsten einer Neuimplementierung unter dem Namen "Microsoft To Do" [*https://todo.microsoft.com/*]. Dieses Buch behandelt – als Erinnerung an das verblichene Wunderlist – eine Aufgabenverwaltung "MiracleList" mit ähnlicher Navigation.

Der angemeldete Benutzer in MiracleList kann eine Liste von Aufgabenkategorien erstellen und in jeder Kategorie eine Liste von Aufgaben anlegen. Eine Aufgabe besteht aus einem Titel, einer Notiz, einem

Eintragsdatum, einem Fälligkeitsdatum und kann als erledigt markiert werden. Über die Funktionen von Wunderlist hinaus kann eine Aufgabe drei (A, B oder C) statt nur zwei Wichtigkeitsgrade (Wichtig ja/nein) sowie einen Aufwand (Zahl) besitzen. Bewusst besitzt der Aufwand keine Maßeinheit; der Benutzer kann selbst entscheiden, ob er den Aufwand in Stunden, Tagen oder nur in relativen Werten, wie z.B. "1" (für niedrig) bis "10" (für hoch), vergeben will. Wie bei Wunderlist kann eine Aufgabe Teilaufgaben besitzen, wobei eine Teilaufgabe nur einen Titel und einen Status besitzt.

Einige Details, die das Original noch zusätzlich beherrscht (z.B. die Suche nach Hashtags, das Duplizieren und Drucken von Listen sowie der Aufgabenaustausch zwischen Benutzern), sind in MiracleList noch nicht berücksichtigt. Wir können ja in einem kleinen Buch nicht eine 100-Millionen-Dollar-App nachprogrammieren – zumindest nicht komplett 😉.

Einige Funktionen wie anklickbare Hyperlinks in Aufgabentexten sind nicht realisiert, um einen Missbrauch der für alle Nutzer offenen Website zu vermeiden.

6.4.2 MiracleList-Client-Features

Nicht alle MiracleList-Funktionen sind in bereits in allen Clients implementiert. Grund dafür ist hauptsächlich Zeitmangel bei mir. **Bitte beachten Sie, dass MiracleList nur zu Test- und Weiterbildungszwecken (Schulungen, Fachbücher) dient!**

> **Hinweis:** In der folgenden Tabelle bedeuten die zusammengelegten Zellen, dass die vier in diesem Buch vorgestellten Blazor-Varianten bei diesen Punkten eine gemeinsame Codebasis besitzen. Wie dies realisiert ist, erfahren Sie im Kapitel "Fallbeispiel MiracleList".

✓ vorhanden ☹ noch nicht vorhanden ☒ nicht möglich wg. Sandbox bzw. Plattformbeschränkungen

Client	Angular	Angular	Angular	Vue.js	Blazor Server	Blazor Web-Assembly und Blazor United	Blazor Desktop	Blazor MAUI	Blazor Web-Assembly *Tutorial*
Umgebung	Browser	Electron	Cordova	Browser	Browser	Browser	WPF	MAUI	Browser
Codequelle	GitHub	GitHub	GitHub	GitHub	Buch	Buch	Buch	Buch	GitHub
Anmeldung	✓	✓	✓	✓		✓			✓
Anmeldung an verschiedenen Backends	☹	☹	☹	☹		✓			☹
Statusanzeige für Backendsysteme auf der Anmeldeseite	☹	☹	☹	☹		✓			☹
Permanenter Backend-Status in Hauptansicht	✓	✓	✓	☹		✓			☹
Permanenter Signal-Hub-Status in Hauptansicht	☹	☹	☹	✓		✓			☹
Aufgabenkategorien	✓	✓	✓	✓		✓			✓
Aufgaben	✓	✓	✓	✓		✓			✓
Unteraufgaben	✓	✓	✓	✓		✓			☹
Aufgaben-Detailansicht ohne Bearbeitung	✓	✓	✓	☹		☹			☹
Aufgaben bearbeiten mit Validierung	✓	✓	✓	✓		✓			✓
Aufgaben löschen	✓	✓	✓	✓		✓			✓
Aufgaben zuordnen per Drag&Drop	☹	☹	☹	✓		✓			☹
Aufgaben sortieren per Drag&Drop	✓	✓	✓	✓		☹			☹
Datei-Upload zu Aufgaben	☹	☹	☹	☹		✓			✓
Kontextmenü für Aufgaben	✓	✓	✓	☹		✓			☹
Ausblenden erledigter Aufgaben	☹	☹	☹	☹		✓			☹
Liste der fälligen Aufgaben	✓	✓	✓	☹		☹			☹
Aufgaben suchen	✓	✓	✓	☹		✓			☹
Aufgabenexport direkt ins Dateisystem	☒	✓	☹	☒	☒	☒	✓	✓	☒
Generierung eines Word-Dokuments	☒	☹	☒	☒	☒	☒	✓	✓ (nur Windows)	☒
Fenster-Synchronisation mit Push-Nachrichten via SignalR	☹	☹	☹	☹		✓			✓
Toast-Benachrichtigungen	☹	☹	☹	✓		✓			✓
Ansicht und Bearbeitung der Aufgaben in einem DataGrid	☹	☹	☹	☹		✓			☹
Mehrsprachigkeit	☹	☹	☹	☹		☹			☹
Anmeldung persistiert im Browserspeicher	☹	☹	☹	✓		✓			✓
Aktuelle Kategorie persistiert im Browserspeicher	☹	☹	☹	☹		✓			✓
Progressive Web App	☹	☒	☒	☹	☒	✓	☒	☒	☹

Abbildung: MiracleList-Client-Features

6.4.3 Technische Basis für MiracleList

Das MiracleList-Backend ist mit ASP.NET Core auf Basis von .NET und C# realisiert.

Es gibt derzeit insgesamt sieben Implementierungen des Frontends:

- Vue.js mit TypeScript
- Angular mit TypeScript
- Blazor WebAssembly mit .NET/C#
- Blazor Server mit .NET/C#

- Blazor Web App mit Auto-Modus (Blazor United)
- Blazor Desktop mit .NET/C#
- Blazor MAUI mit .NET/C#

In diesem Buch werden nur die Blazor-Varianten besprochen.

6.4.4 Webadressen

Die fertige Webanwendung läuft öffentlich im Internet in der Microsoft-Cloud "Azure":

- Backend mit ASP.NET Core für Blazor WebAssembly- und Angular-Clients:
  https://miraclelistbackend.azurewebsites.net
- Webanwendung "MiracleList"-Frontend mit Blazor Server und C#:
  https://miraclelist-bs.azurewebsites.net
- Webanwendung "MiracleList"-Frontend mit Blazor WebAssembly und C#:
  https://miraclelist-bw.azurewebsites.net
- Webanwendung "MiracleList"-Frontend mit Blazor Web App (Blazor United) und C#:
  https://miraclelist-bu.azurewebsites.net
- Webanwendung "MiracleList"-Frontend mit Vue.js und TypeScript:
  https://miraclelist-vue.azurewebsites.net
- Frontend mit Angular und TypeScript (Webanwendung und Cross-Platform-Desktop-Clients für Windows, Linux und macOS):
  https://miraclelist.azurewebsites.net

6.4.5 Projektmappenaufbau

Die MiracleList-Projektmappe (MiracleList.sln) besteht aus zahlreichen Projekten, die logisch in sechs Projektmappenordner sortiert sind:

- **Backend**: Alle Projekte für das MiracleList-Backend
- **Blazor Apps Head Projects**: Die verschiedenen Implementierungen des Frontends mit verschiedenen Blazor-Varianten
- **Blazor Apps Shared Code:** Gemeiner Code der Head-Projekte
- **Solution Items**: Konfigurationsdateien für Editor, Paketmanager und Build (.editorconfig. .gitignore, nuget.config, Directory.Build.props u.a.), Hilfspakete sowie Skripte für DevOps-Prozesse
- **Tests**: Unit Tests, Integrationstests und Browser-UI-Tests
- **Tools**: Hilfsprojekte für das Anlegen der Datenbank im DevOps-Prozess und ein Diagramm der Geschäftsobjektklassen

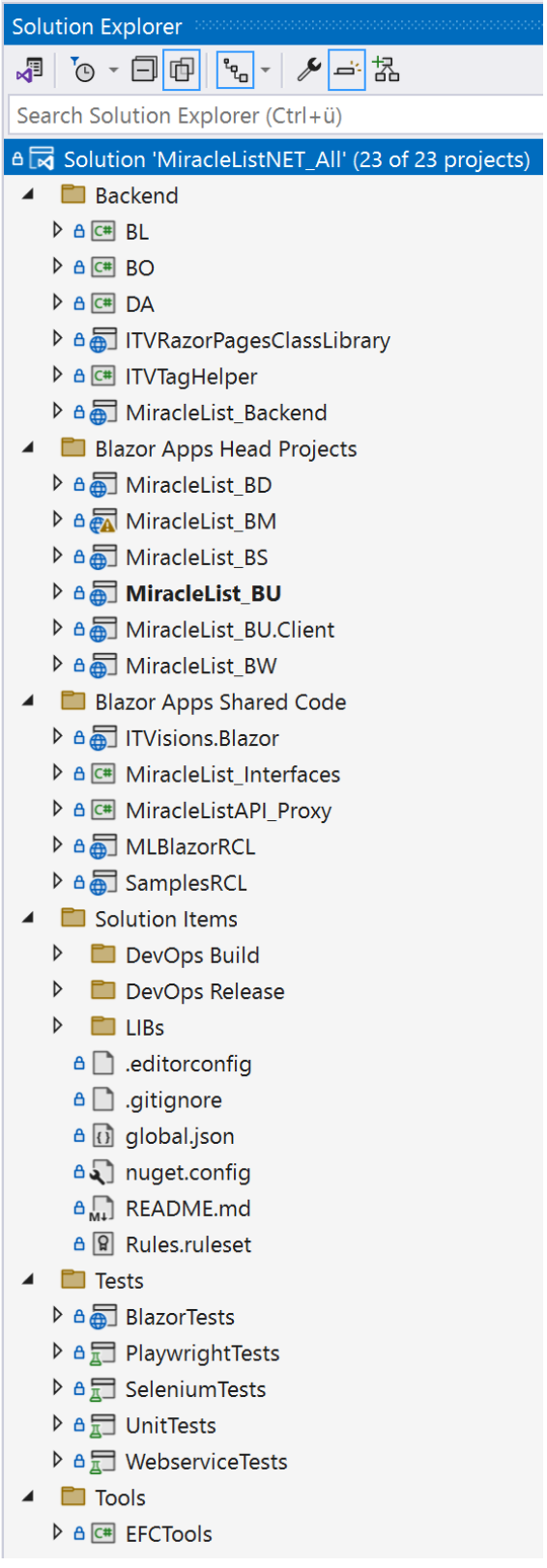


Abbildung: Aufbau der MiracleList-Projektmappe

6.4.6 MiracleList-Bildschirmmasken

Dieses Kapitel erläutern die wesentlichen Funktionen der MiracleList-Anwendung aus Benutzersicht, gibt Ihnen aber auch technische Hinweise.

Die Anmeldeseite fragt nach E-Mail-Adresse und Kennwort. Damit es leicht ist, MiracleList als Demonstrations-Anwendung zu nutzen, ist eine explizite Benutzerregistrierung ausdrücklich nicht vorgesehen. **Wenn es für diese E-Mail-Adresse noch kein Benutzerkonto gibt, wird automatisch ein neues Benutzerkonto mit einigen Beispielaufgaben angelegt.** Merken Sie sich das Kennwort für zukünftige Anmeldungen. Das Kennwort kann nicht geändert werden. Das Benutzeranmeldeverfahren ist bewusst vereinfacht, um die Hürde zum Einstieg in die Anwendung gering zu halten.

Die Blazor Server-Implementierung bietet bei "Server" Datenbankserver an. Diese können Sie in der appsettings.json-Datei unter "ConnectionString" konfigurieren.

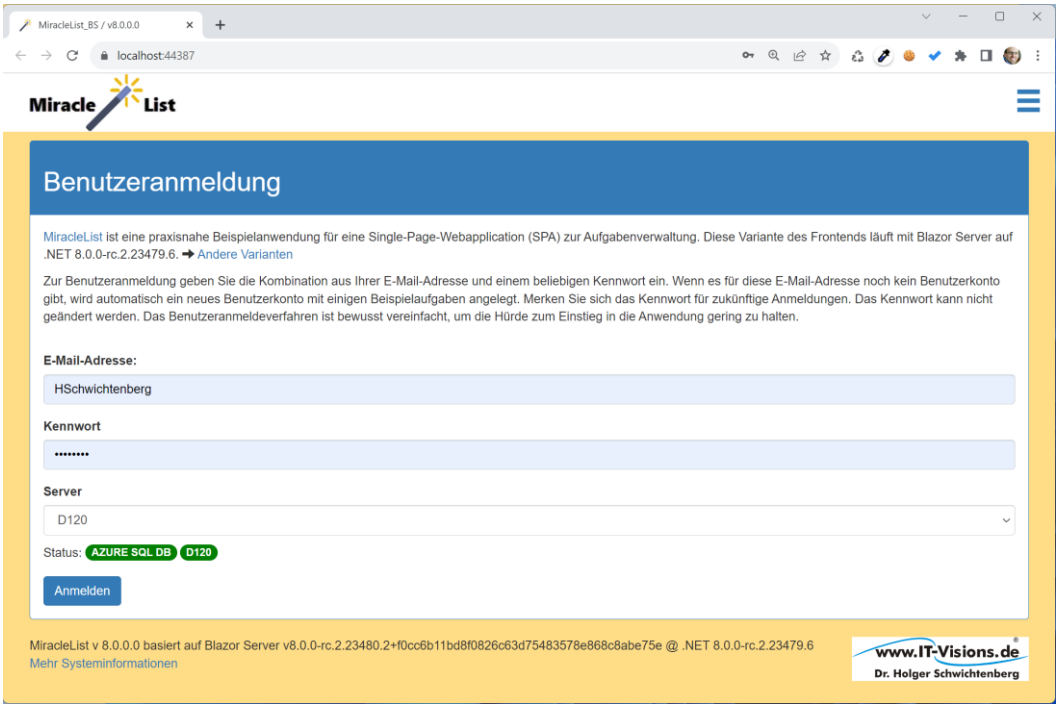


Abbildung: Anmeldeseite bei der Blazor Server-Implementierung des MiracleList-Frontends mit Auswahl zwischen verschiedenen Datenbankservern

Die Blazor WebAssembly-Implementierung bietet bei "Server" statt der Datenbankserver Application Server an.

Wenn die MiracleList-Anwendung in Visual Studio gestartet wird, wird als Server/Backend zusätzlich "localhost" zur Auswahl gestellt, um sich mit einer lokalen Version des WebAPI-Backends bzw. einer lokalen Version der Datenbank, statt dem in Azure gehosteten Varianten zu verbinden. Unter der Backend-Server-Auswahl sieht man eine Zeile mit Zustandskennzeichen ("Badges") für die Server. Ein grünes Abzeichen bedeutet, dass der Server verfügbar ist. Grau bedeutet, der Zustand wird gerade ermittelt. Bei einem roten Abzeichen ist der Server nicht verfügbar.

Hinweis: Wenn Sie localhost als Backend-Server nutzen wollen, müssen Sie auf Ihrem Rechner das Backend (Projekt MiracleList_Backend.csproj) zuvor gestartet und auch die Datenbank angelegt haben (siehe dazu Readme.md)! Wenn Sie das Backend auf einem anderen Port laufen lassen, müssen Sie in /wwwroot/AppSettings.json den Port ändern!

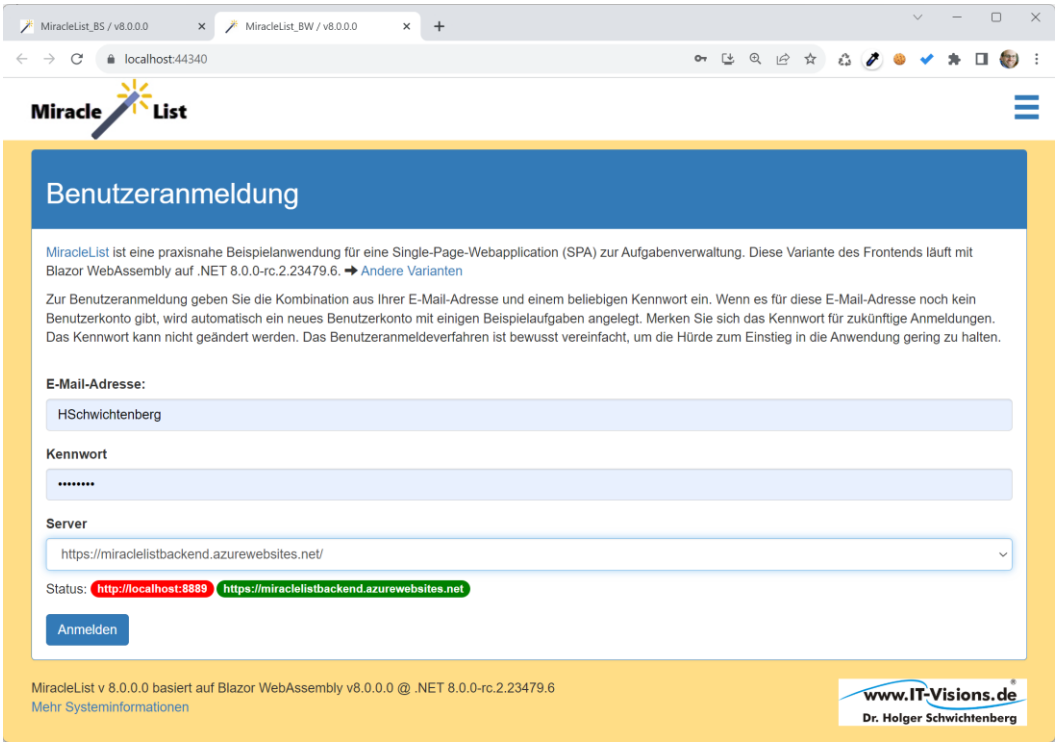


Abbildung: Anmeldeseite bei der Blazor WebAssembly-Implementierung des MiracleList-Frontends. Beim Start der Blazor WebAssembly-Anwendung auf "localhost" wird auch das lokale Backend angeboten, das aber hier nicht gestartet ist (daher rot!)

Wenn Sie die Blazor United-Implementierung starten, werden Sie diesen Effekt feststellen:

- Die Startseite zeigt beim allerersten Aufruf in der Fußzeile an, dass Sie mit Blazor Server läuft.
- Dennoch werden dort keine Datenbanken zur Auswahl angeboten, sondern Application Server, denn eine 2-Tier-Datenbankzugriff würde nach dem Wechsel zu Blazor WebAssembly nicht mehr funktionieren.
- Wenn Sie die Webanwendung aber dann ein zweites Mal aufrufen, steht in der Fußzeile nur noch ganz kurz "Blazor Server". Dann wechselt die Anzeige auf "Blazor WebAssembly" (zudem sehen Sie in der Titelleiste den Zusatz ".Client"), weil in der Zwischenzeit alle für Blazor WebAssembly notwendigen Dateien nachgeladen wurden.

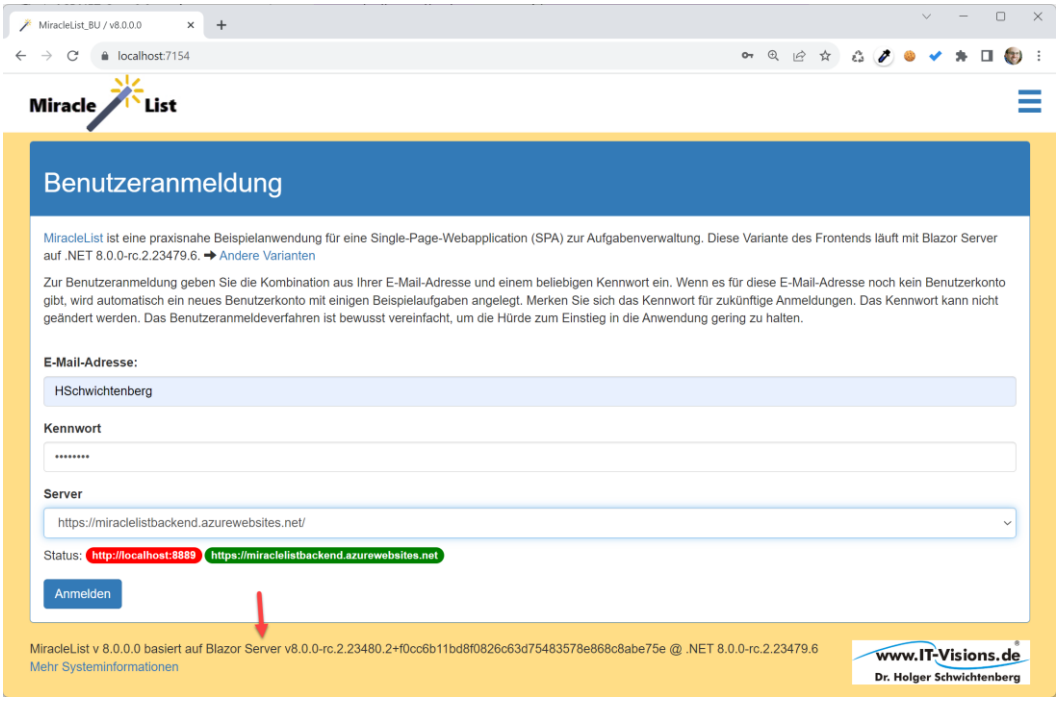


Abbildung: Anmeldeseite bei der Blazor United-Implementierung des MiracleList-Frontends beim ersten Aufruf

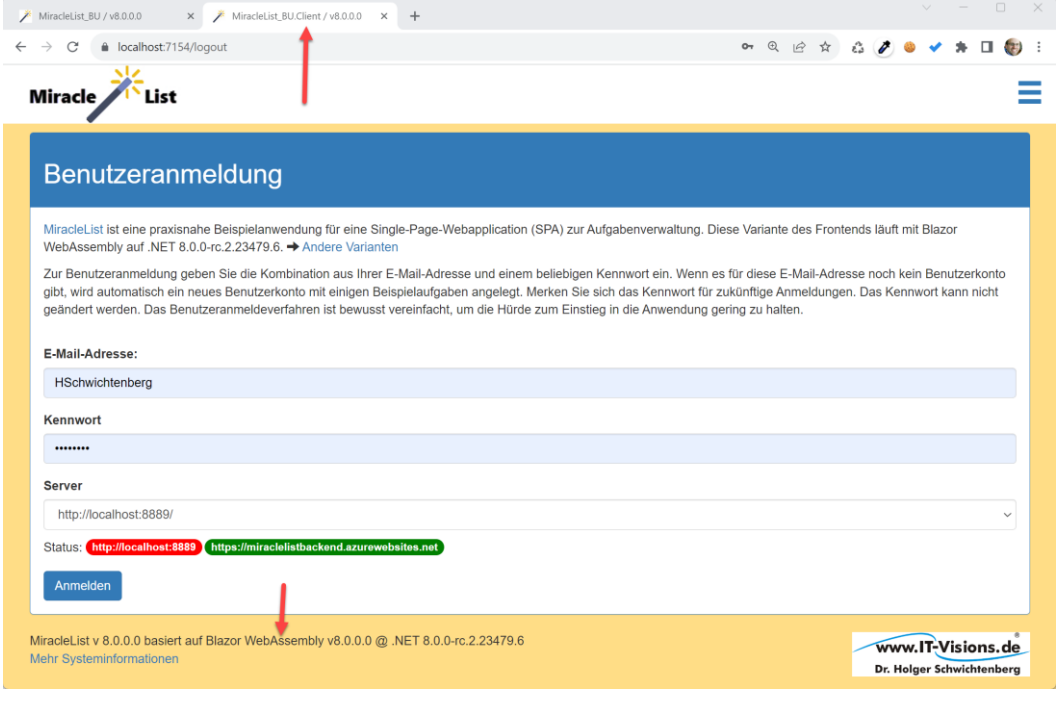


Abbildung: Anmeldeseite bei der Blazor United-Implementierung des MiracleList-Frontends beim zweiten Aufruf

Bei Blazor Desktop stehen ebenfalls Datenbankserver zur Auswahl, denn hier sind keine Webservices/Application Server als Zwischenschicht notwendig. Die Blazor MAUI-Implementierung

verwendet hingegen wieder das 3-Tier-Modell, denn nicht alle Plattformen können direkt Datenbankmanagementsystem aufrufen.

Abbildung: MiracleList in der Blazor Desktop-Variante, gehostet in einer Windows-Desktop-Anwendung mit Windows Presentation Foundation (WPF) mit direktem Datenbankzugriff wie bei Blazor Server

Der angemeldete Benutzer kann eine Liste von Aufgabenkategorien erstellen und in jeder Kategorie eine Liste von Aufgaben anlegen.

Hinweis: Neue Benutzer erhalten automatisch eine Reihe von Beispielaufgaben!

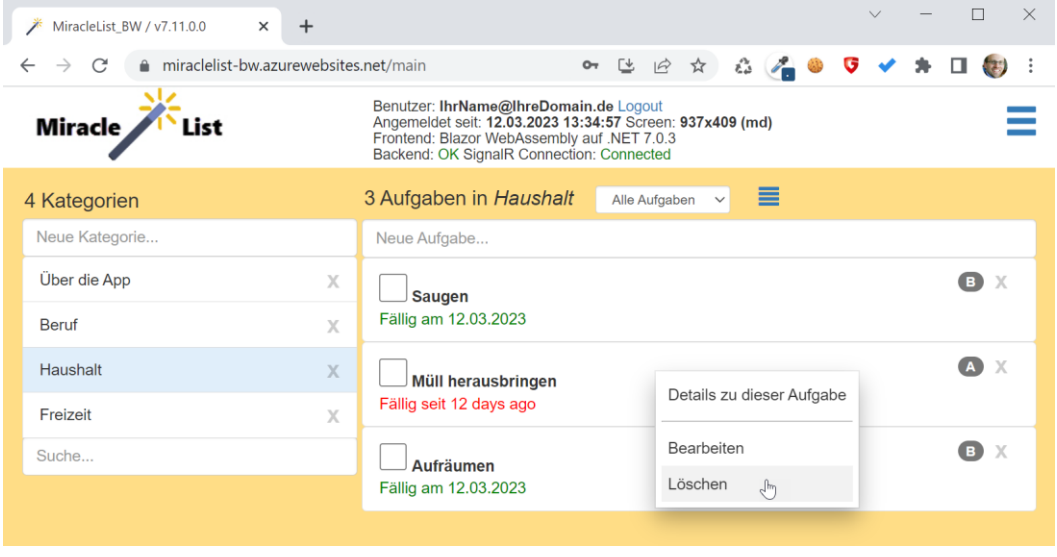


Abbildung: Hauptansicht der MiracleList-Anwendung mit Aufgabenfilter und Kontextmenü für die einzelnen Aufgaben

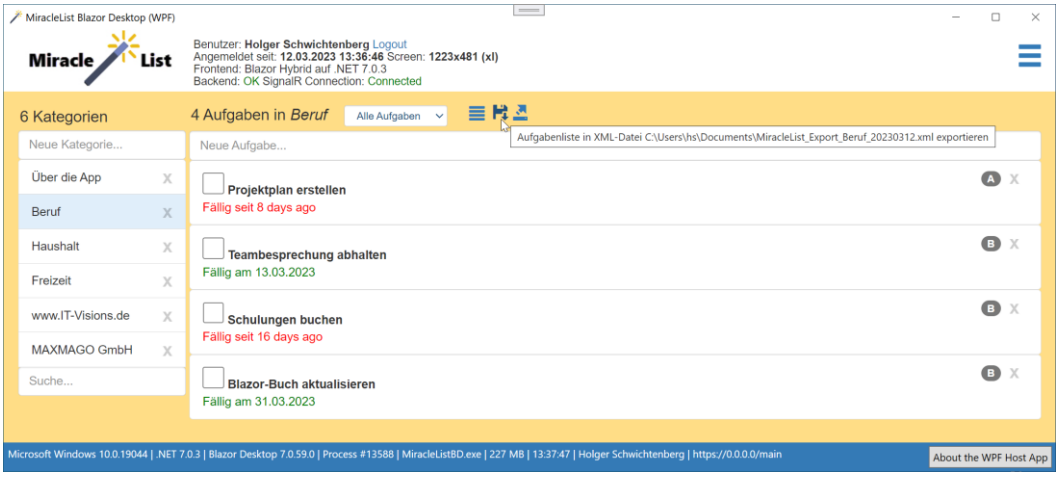


Abbildung: Hauptansicht der MiracleList-Anwendung in der in WPF gehosteten Variante "Blazor Desktop" mit zusätzlicher Export-Funktion in Dateisystem

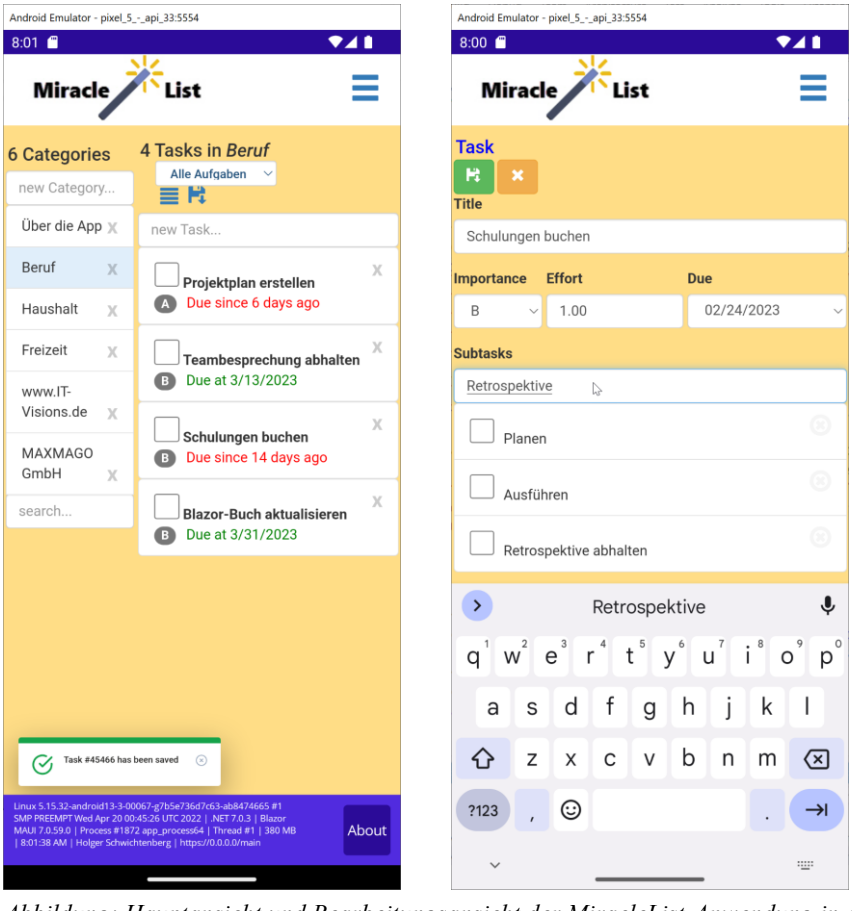


Abbildung: Hauptansicht und Bearbeitungsansicht der MiracleList-Anwendung in einem Android-basierten SmartPhone (implementiert mit Blazor MAUI)

Durch Klick auf eine Aufgabe kommt der Benutzer zur Bearbeitungsansicht, die rechts als dritte Spalte erscheint. Einige Eingabefelder besitzen Validierungsbedingungen, bei deren Nicht-Einhaltung die Felder rot umrandet werden und es erscheint eine Fehlermeldung. Beim Titel erscheint auch ein rotes x. Auch das Hochladen von Dateien zu einer Aufgabe ist möglich.

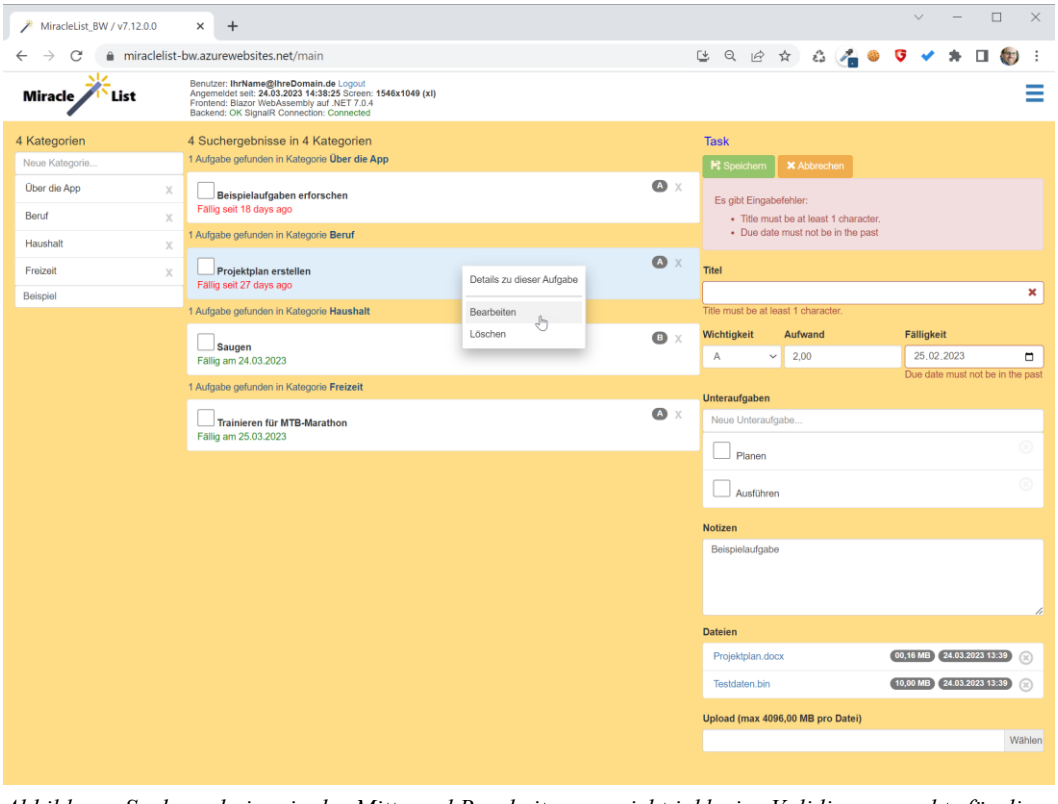


Abbildung: Suchergebnisse in der Mitte und Bearbeitungsansicht inklusive Validierung rechts für die ausgewählte Aufgabe

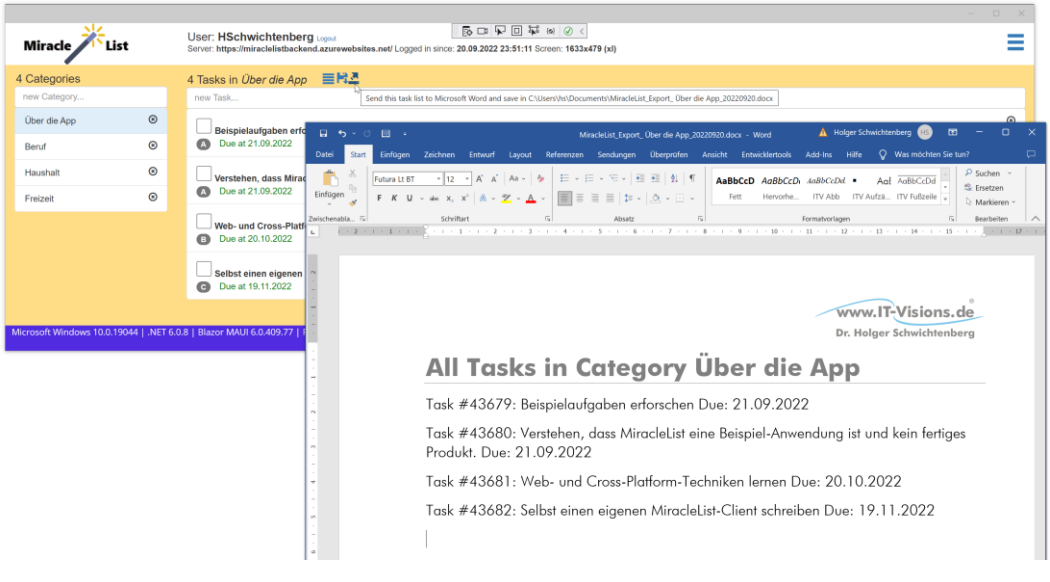


Abbildung: Eine Export-Funktion in ein Microsoft Word-Dokument via COM-Objektmodell gibt es nur in Blazor Desktop (immer auf Windows) und Blazor MAUI (wenn es auf Windows läuft)

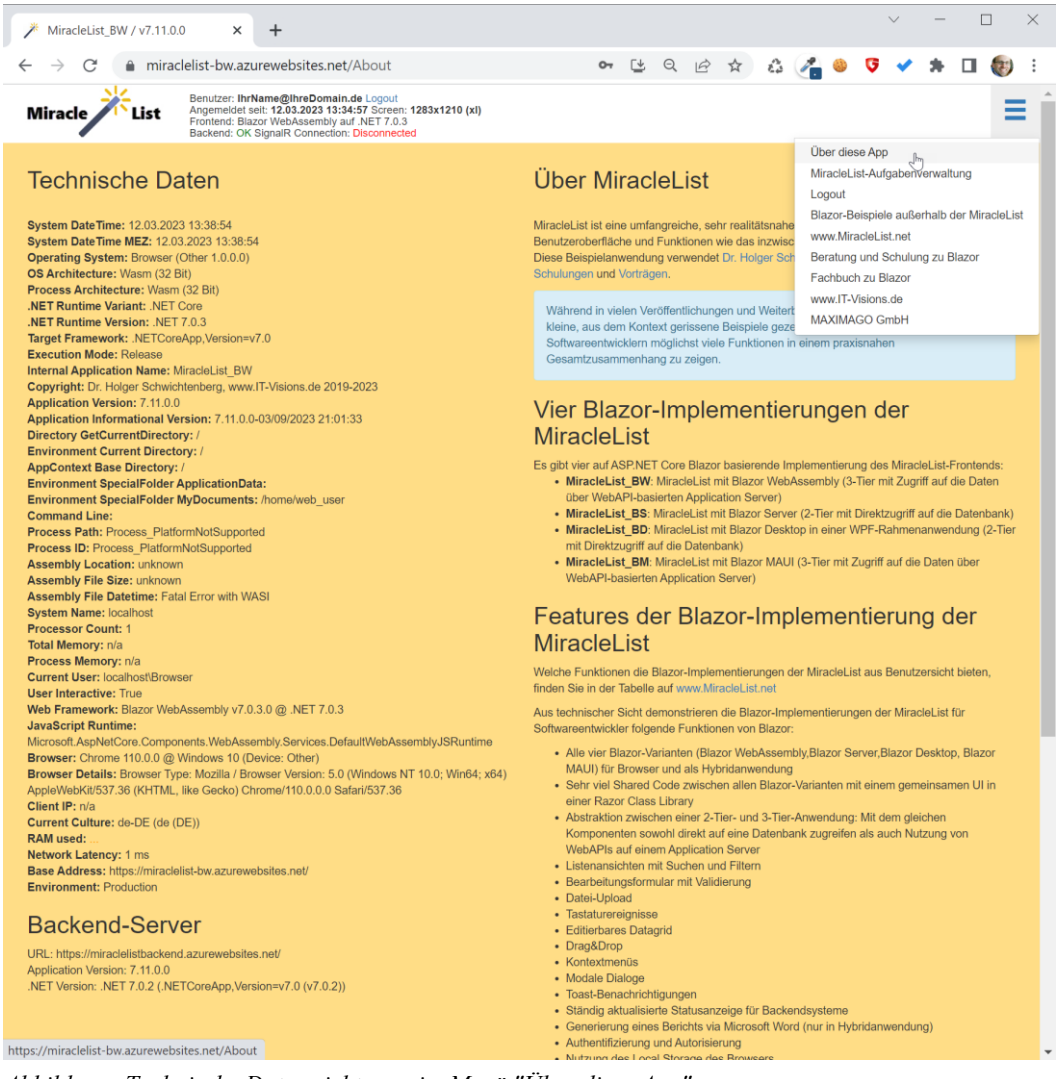


Abbildung: Technische Daten sieht man im Menü "Über diese App"

6.5 Weitere Beispiele innerhalb der MiracleList-Webanwendung

Viele Beispiele in diesem Buch stammen direkt aus dem Anwendungsszenario der Aufgabenverwaltung "MiracleList". Allerdings stammen nicht alle Beispiele dieses Buchs aus dem Szenario der Aufgabenverwaltung selbst. Einige Beispiele greifen nicht auf die Geschäftslogik und die Datenbank von MiracleList zurück. Dies hat zwei Gründe:

- Einige Beispiele passen inhaltlich einfach nicht in das Szenario "MiracleList".
- Aus didaktischen Gründen ist es an einigen Stellen geboten, einfachere Beispiele zu behandeln.

Alle Beispiele, die nicht Teil der Aufgabenverwaltung sind, finden Sie dennoch in den beiden Blazor-Projekten der MiracleList. Sowohl in der Blazor WebAssembly- als auch der Blazor Server-Variante rufen Sie diese unter der relativen URL /Demos oder den Menüpunkt "Blazor-Beispiele außerhalb der MiracleList" auf.

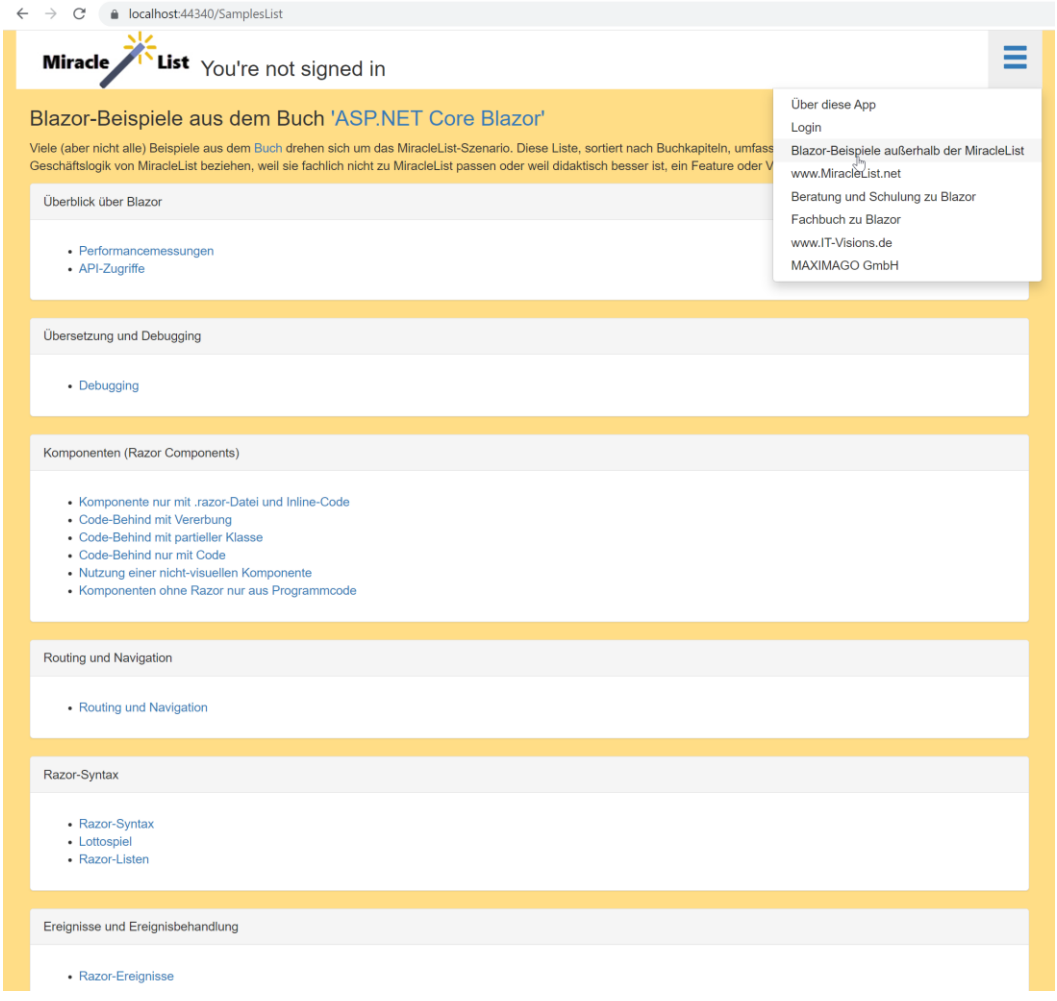


Abbildung: Die Blazor-Beispiele außerhalb des MiracleList-Szenarios sind in das Menü der MiracleList integriert und dort gut nach Buchkapitel sortiert.

Hinweis: Den Quellcode der meisten Beispiele in diesem Buch, deren URL mit "/Demos" beginnt, finden Sie in dem Projekt /src/Samples. Nur einige wenige Beispiele, die spezifisch sind für eine bestimmte Blazor-Variante, finden Sie in den spezifischen Projekten, also /src/MiracleList_BW/Demos und /src/MiracleList_BS/Demos sowie /src/MiracleList_BD/Demos.

6.6 Visual Studio 2022

Für die Entwicklung von Blazor-Anwendungen sollten Sie die Microsoft-Entwicklungsumgebung Visual Studio 2022 verwenden in der jeweils aktuellen Update-Version.

Wichtig: Für .NET 9.0 und Blazor 9.0 brauchen Sie mindestens Version 17.12 von Visual Studio 2022!

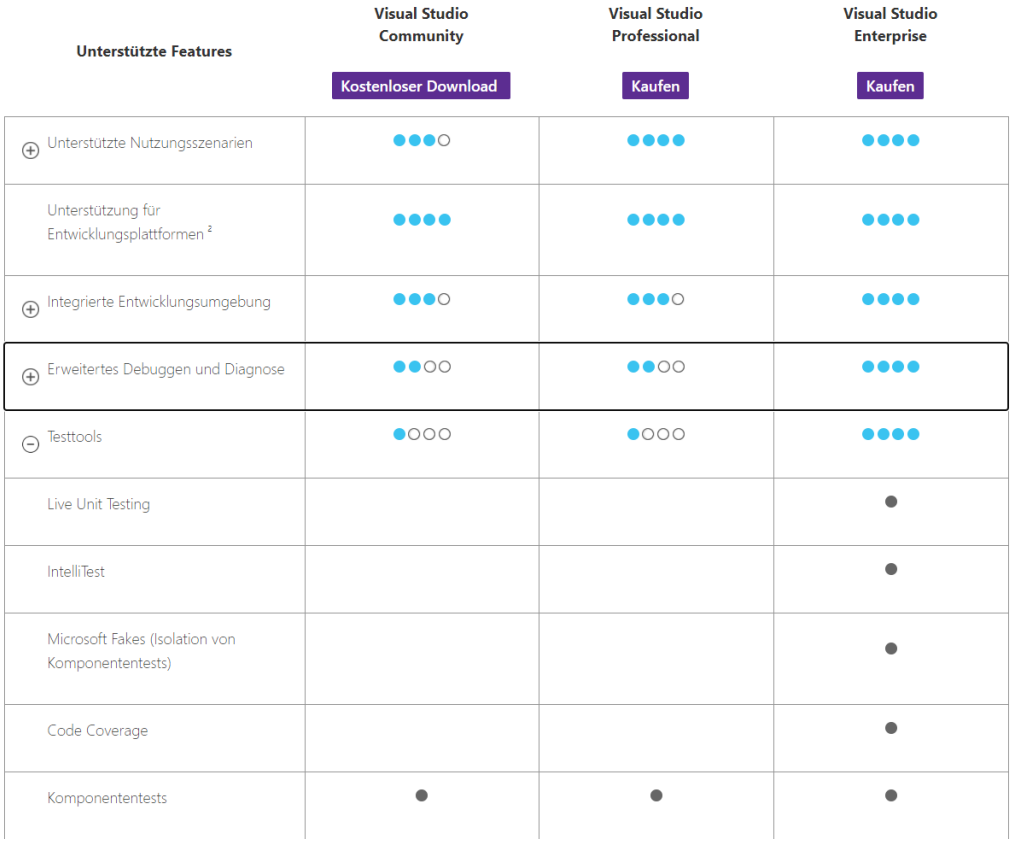
Bezugsquelle: *https://visualstudio.microsoft.com/de/vs*

6.6.1 Varianten von Visual Studio

Es gibt Visual Studio 2022 in drei Varianten: Community, Professional und Enterprise. Für die Entwicklung von Blazor-Anwendungen ist (für Einzelpersonen und Organisation bis fünf Entwickler) die kostenfreie Community-Variante ausreichend. Einige fortgeschrittene Werkzeugfunktionen (z.B. einige

Analyse- und Testfunktionen wie IntelliTrace, Live Unit Testing und Code Coverage) erhalten Sie nur mit der Enterprise-Version. Einen Vergleich der Fähigkeiten der Versionen finden Sie unter:

https://visualstudio.microsoft.com/de/vs/compare



Unterstützte Features	Visual Studio Community Kostenloser Download	Visual Studio Professional Kaufen	Visual Studio Enterprise Kaufen
⊕ Unterstützte Nutzungsszenarien	●●●○	●●●●	●●●●
Unterstützung für Entwicklungsplattformen [2]	●●●●	●●●●	●●●●
⊕ Integrierte Entwicklungsumgebung	●●●○	●●●○	●●●●
⊕ Erweitertes Debuggen und Diagnose	●●○○	●●○○	●●●●
⊖ Testtools	●○○○	●○○○	●●●●
Live Unit Testing			●
IntelliTest			●
Microsoft Fakes (Isolation von Komponententests)			●
Code Coverage			●
Komponententests	●	●	●

Abbildung: Ausschnitt aus dem Vergleich der Visual Studio-Varianten

Die aktuellen Preise finden Sie unter *https://visualstudio.microsoft.com/de/vs/pricing*.

Tipp: Kostenfrei können Sie die jeweils aktuelle Preview-Version verwenden, die es hier gibt: *https://visualstudio.microsoft.com/de/vs/preview*. Diese ist aber jeweils in einigen Punkten noch nicht ausgereift und der Einsatz für die professionelle Softwareentwicklung daher nicht empfohlen.

6.6.2 Stabile Version und Preview-Version

Man kann eine stabile Version und eine Preview-Version parallel auf einem Windows-System installieren.

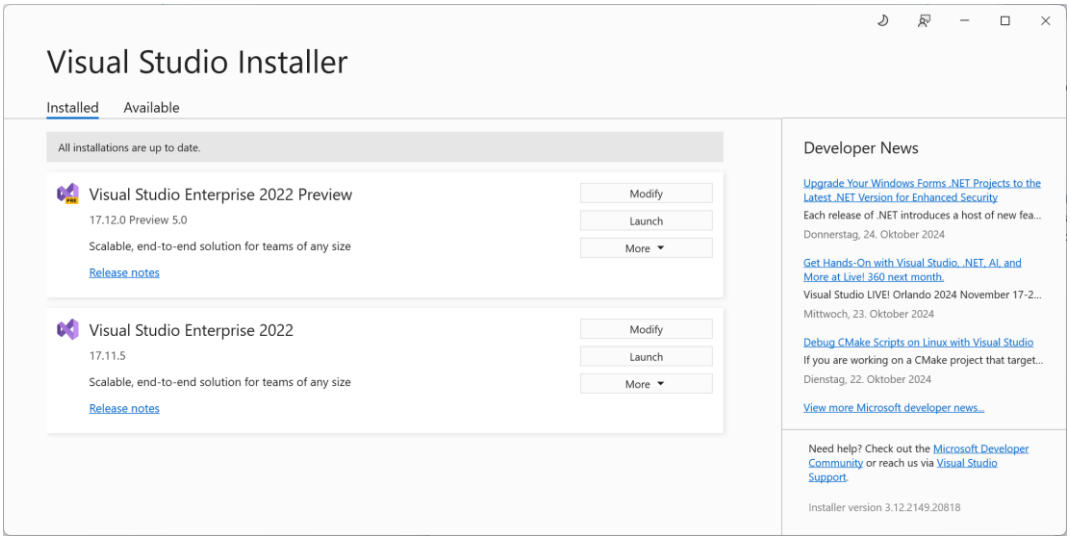


Abbildung: Parallelinstallation der stabilen Version und einer Preview-Version

Hinweis: Visual Studio 2022 entspricht der Version 17. Seit einigen Jahren aktualisiert Microsoft die Entwicklungsumgebung im Abstand von wenigen Wochen und vergibt dafür neue Nummern an der zweiten Stelle: 17.1, 17.2, 17.3, 17.4, 17.5 usw.

6.6.3 Workloads

Visual Studio besteht aus vielen Bausteinen, die Optionen bei der Installation darstellen. Vorgefertigte Zusammenstellungen von diesen Bausteinen heißen Workloads.

Hinweis: Sowohl Visual Studio als auch das .NET SDK kennen den Begriff "Workload". Ein Workload in Visual Studio ist aber nicht das gleiche wie im .NET SDK.

Für die Softwareentwicklung mit Blazor müssen Sie in Visual Studio den Workload "ASP.NET and web development" installieren. Der Workload "Azure development" wird nur benötigt, wenn Sie die Anwendung direkt in Microsoft Azure verbreiten möchten, ohne DevOps-Pipeline.

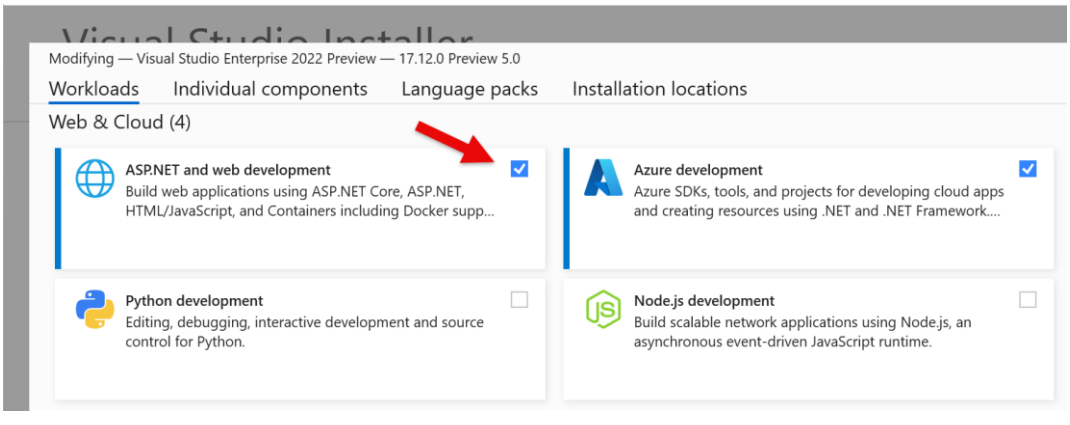


Abbildung: Notwendige Visual Studio-Workloads für Blazor

6.6.4 Sprachversionen

Verwenden Sie IMMER die englische Version von Visual Studio, nicht die deutsche Übersetzung. Diese hat einige Übersetzungsschwächen, die das Verständnis erschweren. Zudem ist das "googeln" nach Hilfe im Internet eingeschränkt, wenn Sie nach deutschen Begriffen suchen, weil Sie die genaue englische

Bezeichnung nicht kennen. Ich werde daher in diesem Buch immer nur die englischen Namen für Menüpunkte, Einstellungen etc. verwenden.

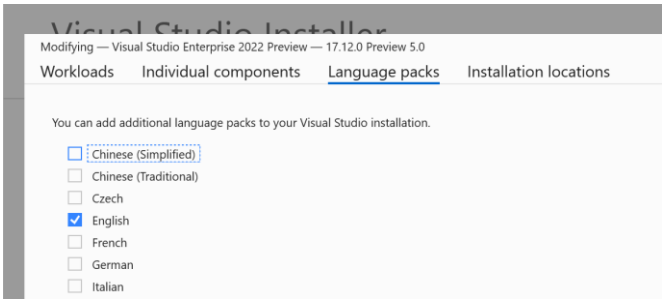


Abbildung: Die Sprachversion wählen Sie über "Language Packs" im Visual Studio-Setup

6.7 .NET SDK

Ein typisches Problem, das Sie nicht nur mit den Programmcodebeispielen in diesem Buch, sondern auch mit Ihren Projekten in der Praxis haben können, ist das Fehlen der notwendigen Installation des passenden .NET Software Development Kits auf Ihrem Entwicklungssystem. Dann kann Visual Studio die Projekte nicht laden, siehe folgende Abbildung.

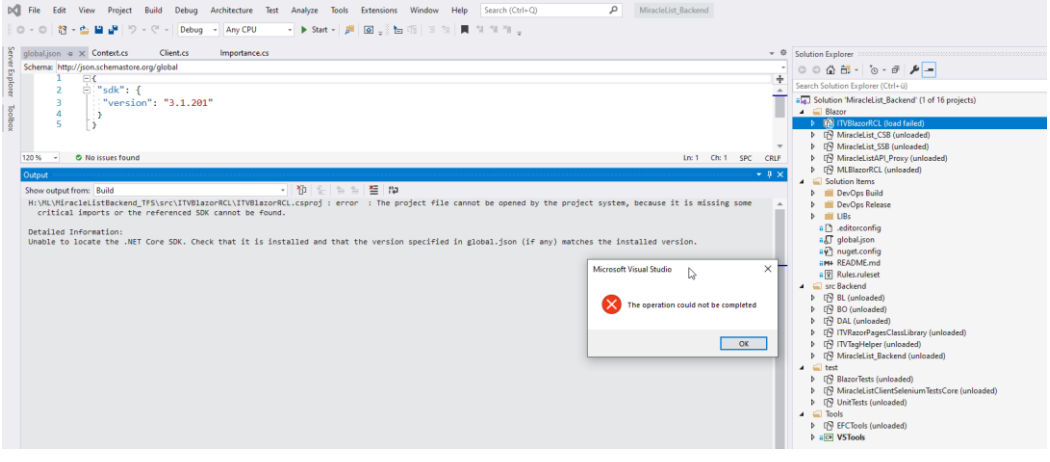


Abbildung: Die notwendige SDK-Version fehlt

Sie können die installierten SDK-Versionen an der Kommandozeile abrufen:

```
dotnet --list-sdks
```

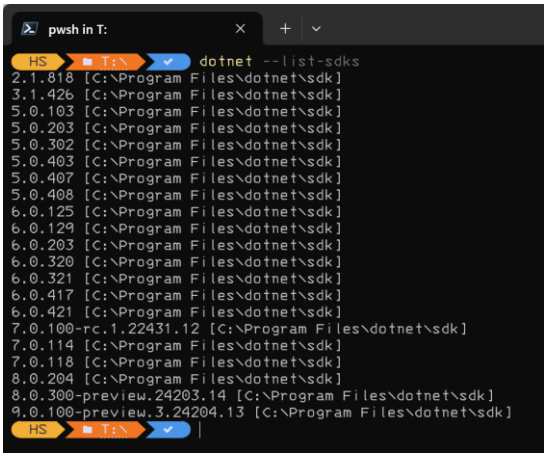


Abbildung: Abruf der Liste der auf einem System installierten .NET SDKs via dotnet --list-sdks

Seit .NET 6.0 ist es möglich, den Aktualitätsstand der installierten SDK- und Runtime-Versionen abzufragen. Der Befehl dazu lautet:

```
dotnet sdk check
```

Hinweis: Der Kommandozeilenbefehl *dotnet sdk check* ruft einen Webservice bei Microsoft auf. Das heißt aber auch: Auf einem Entwicklerrechner ohne Internetzugang (ja, sowas gibt es tatsächlich in einigen Firmen) funktioniert dotnet sdk check nicht.

Es gibt grundsätzlich zwei Verfahren, wie die .NET SDK-Version gewählt wird:

- Im Standard wird die aktuellste .NET SDK-Version verwendet (einschließlich Preview-Versionen). Dies kann zu unerwünschten Effekten beim Kompilieren führen, z.B. aufgrund von Veränderungen (Breaking Changes) in neueren SDK-Versionen.
- Wenn es eine Datei global.json in einem Dateisystemordner gibt, wird die dort festgelegte SDK-Version für diesen Ordner und alle Unterordner verwendet.

Hinweis: Der Einsatz einer global.json-Datei ist empfohlen. Sie erhalten z.B. in Azure DevOps folgenden Hinweis: "Unless you have locked down a SDK version for your project(s), a newer SDK might be picked up which might have breaking behavior as compared to previous versions."

Wenn Visual Studio die in der global.json festgelegte .NET SDK-Version nicht finden kann, haben Sie zwei Optionen:

- Installieren Sie die in der global.json-Datei verlangte .NET SDK-Version. Sie bekommen diese kostenfrei unter der Webadresse *https://dotnet.microsoft.com/download*.
- Sie ändern die SDK-Version in der global.json auf eines der bei Ihnen installierten .NET SDKs. Dabei müssen Sie keine exakte Versionsnummer angeben: Sie können mit "rollForward": "latestFeature" bzw. "rollForward": "latestMinor" bestimmen, dass aktuellere als die angegebene Version verwendet werden dürfen.

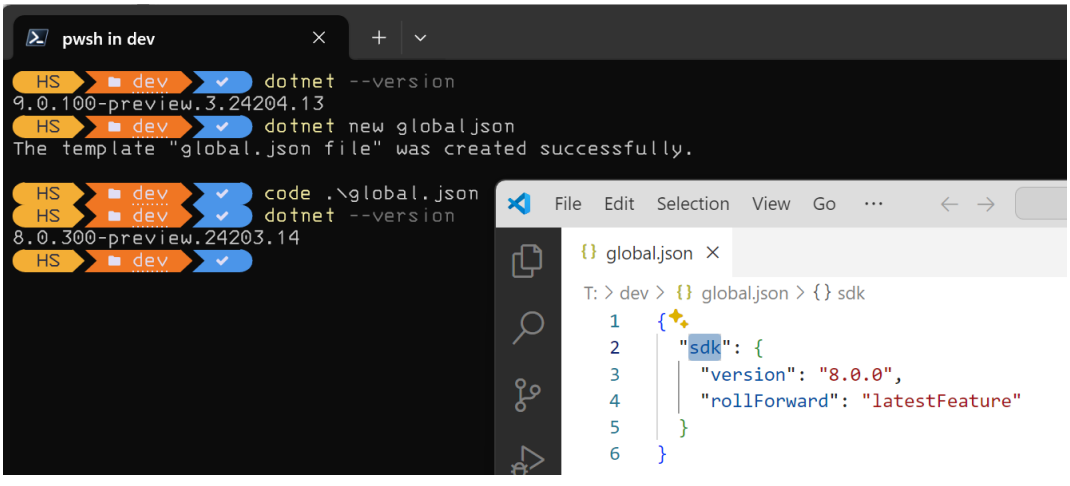


Abbildung: Erstellen einer global.json-Datei mit Einstellung der Verwendung der aktuellsten installierten SDK-Version zu der angegebenen Hauptversion.

Hinweise: Wenn die Eingabeaufforderung den Befehl dotnet nicht finden kann, haben Sie gar kein .NET SDK installiert. Nicht jede .NET SDK-Version ist mit jeder Version von Visual Studio kompatibel.

6.8 .NET WebAssembly Build Tools

Für einige Funktionen bei Blazor WebAssembly wie AOT-Kompilierung und Native Code-Aufruf werden zusätzlich zum .NET SDK die .NET WebAssembly Build Tools benötigt.

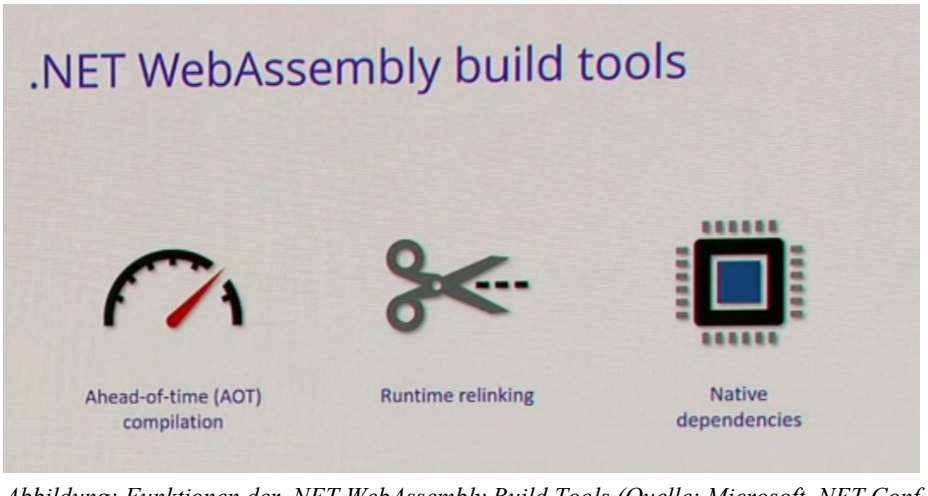


Abbildung: Funktionen der .NET WebAssembly Build Tools (Quelle: Microsoft .NET Conf 2021)

Die ".NET WebAssembly Build Tools" installiert man via Visual Studio (siehe Abbildung)

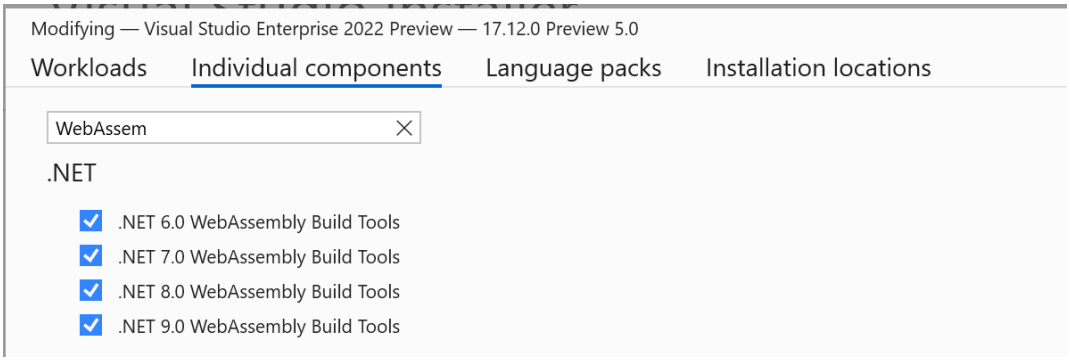


Abbildung: Installation der .NET WebAssembly Build Tools in Visual Studio

oder via Kommandozeile

```
dotnet workload install wasm-tools
```

6.9 Testen Ihrer PC-Konfiguration

In diesem Kapitel wird ein kurzer Test Ihrer Systemkonfiguration durchgeführt mit dem Ziel, festzustellen, ob Ihr System korrekt funktioniert. Es wird ein Blazor WebAssembly-Projekt erstellt.

Geben Sie im Suchdialog "WebAssembly" ein, siehe Abbildung. Wählen Sie "Blazor WebAssembly Standalone App". Nach der Auswahl des Namens (geben Sie ein "MiracleListBW" für den Projektnamen und "MiracleList" für die Projektmappe) und des von Ihnen wählbaren Zielpfades (wie immer sollte man einen Dateisystempfad ohne Leerzeichen wählen, das macht die Verwendung von Kommandozeilenwerkzeugen einfacher), kommt ein weiterer Dialog. Hier ist ".NET 8.0", "Authentication=None", "Configure for HTTPS", "Progressive Web App" und "Include sample pages" (damit man Boostrap als CSS-Framework und drei Beispielseiten bekommt= sowie "Do not use top-level statements" (damit man eine klassische .NET-Startklasse statt losem Startcode bekommt) zu wählen.

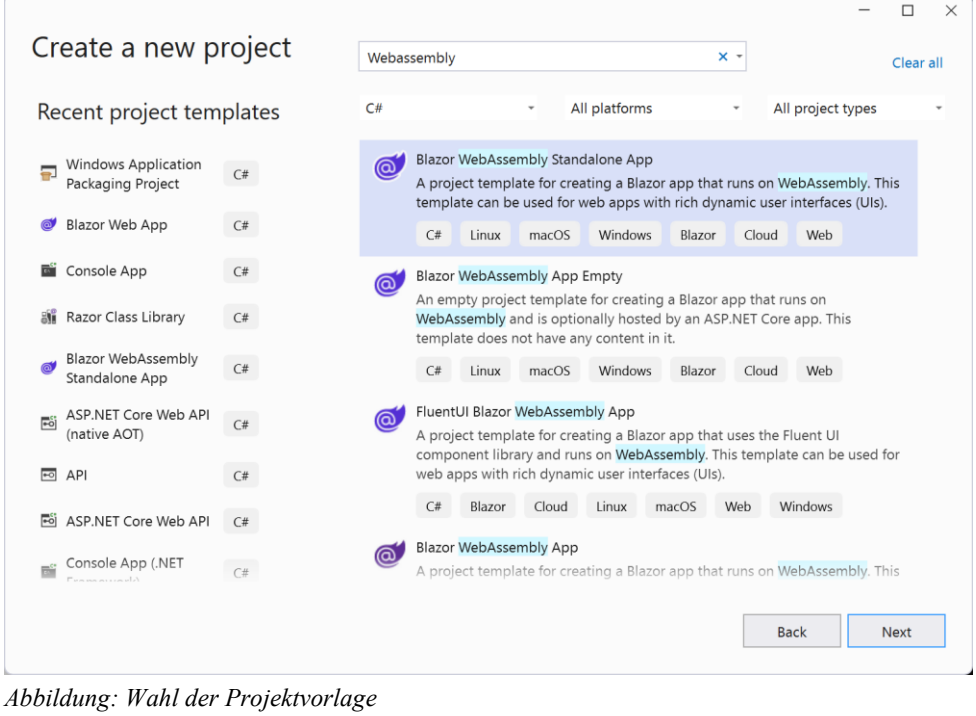


Abbildung: Wahl der Projektvorlage