

Hochschule Offenburg University of Applied Sciences

Embedded Frontend-Entwicklung mit .Net Core und Blazor

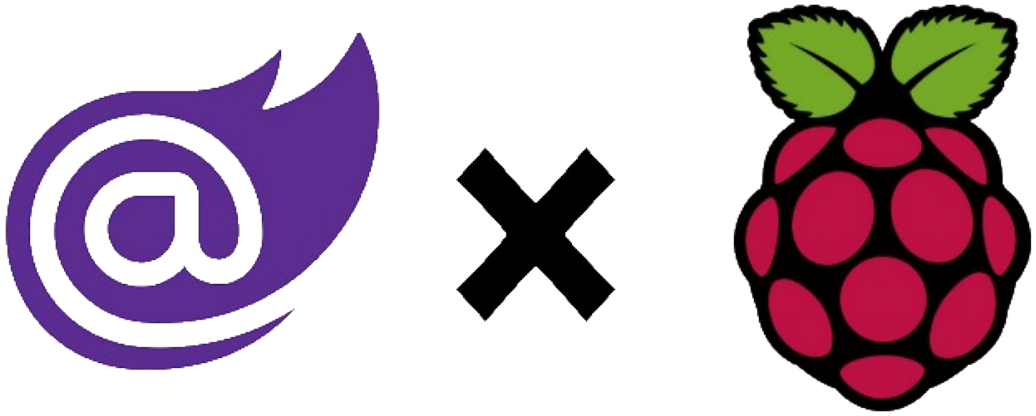
Inhalt

- Einführung
- Stand der Technik
- Blazor
- Laufzeitanalyse
- Fazit / Ausblick

Aufgabenstellung

Einführung

- Frontend-Entwicklung für ein Non-Deeply Embedded System mit Blazor
- Entwicklung auf dem Raspberry Pi 4B
- Laufzeitanalyse



Motivation
Einführung

- Entwicklung einer GUI
- Lange Entwicklungszeiten
- Wenige Verfügbarkeit von Experten auf dem Arbeitsmarkt

```
int** array2d = new int* [50];
for (uint32_t i{}; i < 50; ++i){
    array2d[i] = new int[50];
}

for (uint32_t i{}; i < 50; ++i) {
    for (uint32_t j{}; j < 50; ++j) {
        array2d[i][j] = (i + j) * 2;
    }
}

for (uint32_t i{}; i < 50; ++i) {
    delete[] array2d[i];
}

delete[] array2d;
```

```
int[,] array2d = new int[50, 50];
for (i = 0; i < 50; ++i)
{
    for (int j = 0; j < 50; ++j)
    {
        array2d[i, j] = (i + j) * 2;
    }
}
```

Qt
Stand der Technik

- Framework zur Erstellung von GUI´s auf mehreren Betriebssystemen
- Entwickelt 1990 von Havard Nord und Eirik Chambe-Eng
- In C++ entwickelt wurden
- Intension: Eine Codebasis für alle Betriebssysteme

Alternative zu Qt

Blazor

Die Anwendungsschicht in .NET Core….

Die Persistenz-Schicht in .NET Core….

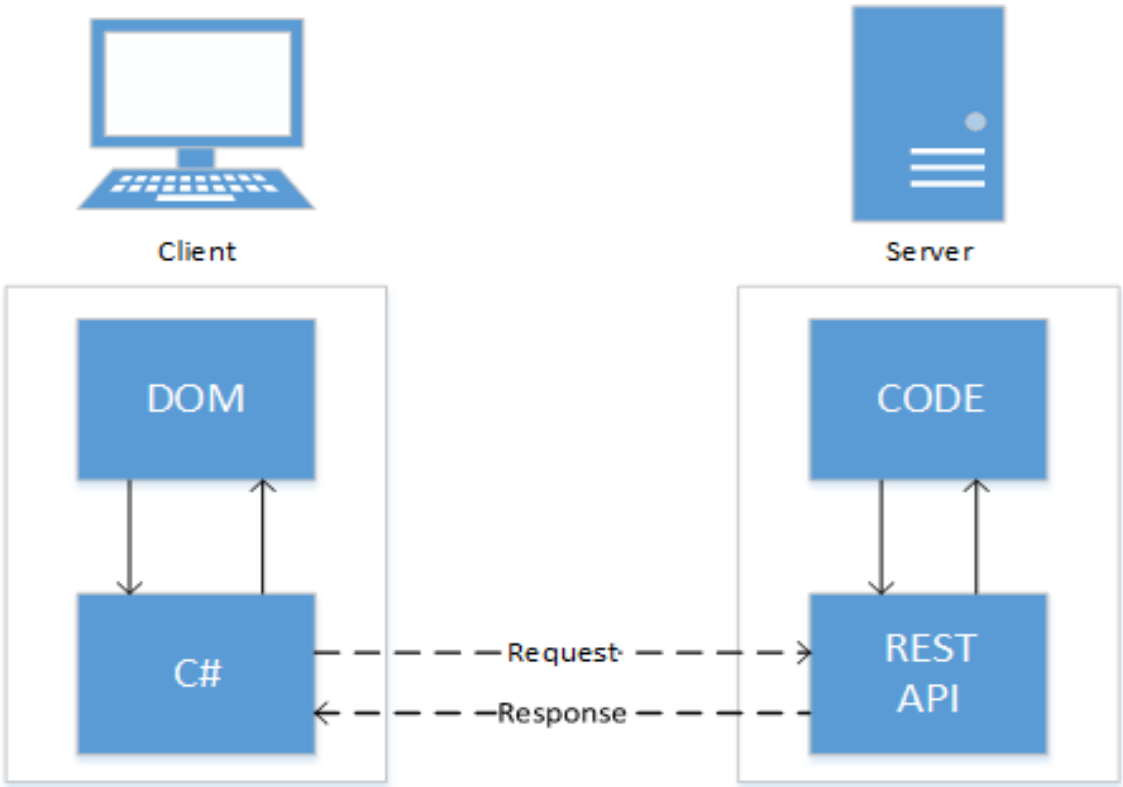
Was ist mit der Benutzeroberfläche?

Was ist Blazor?

Blazor

- Web-Framework von Microsoft
- Erschienen 2019 (Server-Architektur)
- C# im Browser
- Server- und WebAssembly-Architektur
- Intension: Eine Codebasis

WebAssembly
Blazor



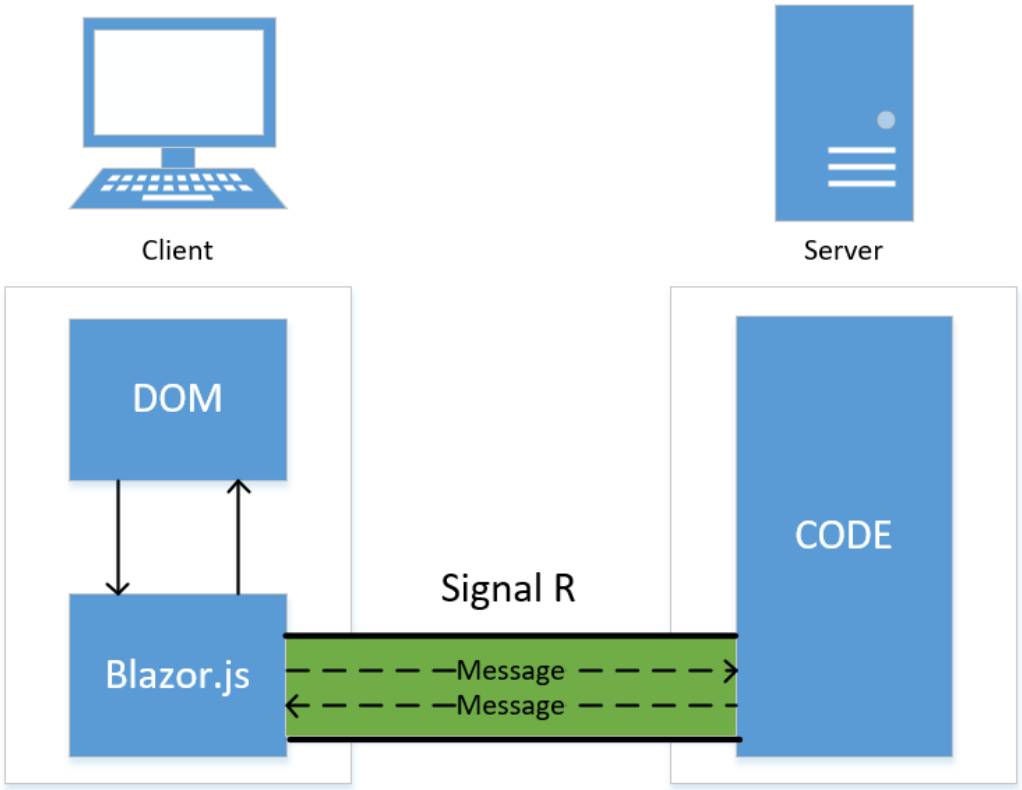
WebAssembly: Vor/Nachteile

Blazor

- Vorteile:
  - Sehr skalierbar
  - Sehr performant
  - Nicht Abhängig vom Server

- Nachteile:
  - Große Datei, die geladen werden muss
  - Lange Ladezeiten beim ersten Aufruf
  - Code ist auf dem Client sichtbar

Server
Blazor



Server: Vor/Nachteile

Blazor

- Vorteile:
  - Kurze Ladezeiten
  - Komplett Browser unabhängig
  - Client hat keinen Zugriff auf den Sourcecode
  - Thin-Client

- Nachteile:
  - Nicht skalierbar
  - Verzögerungen in der Benutzeroberfläche bei langer Latenz
  - Es muss immer eine Verbindung zum Server bestehen

Javascript-Interoperation

Blazor

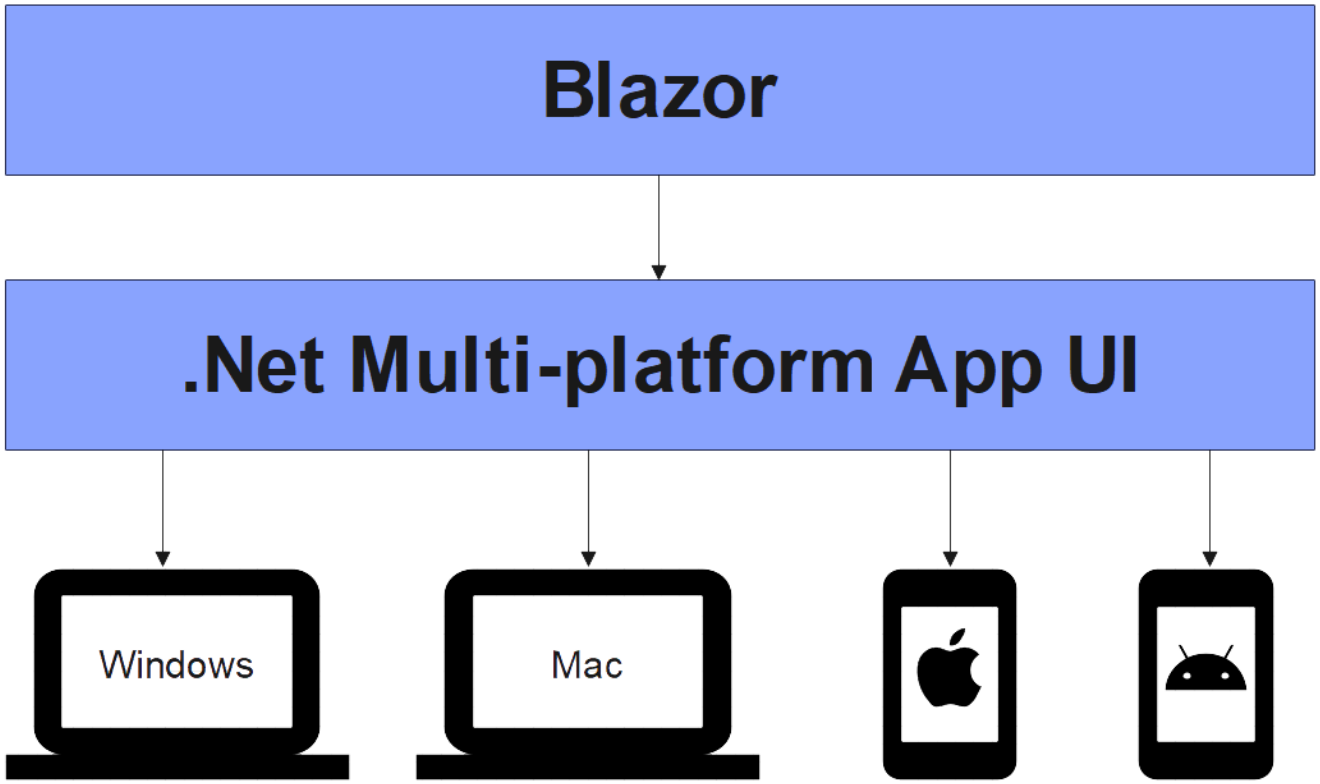
- Javascript Runtime
- **ValueTask<TValue> InvokeAsync<TValue>(string, object?[]?)**
- **ValueTask<TValue> InvokeAsync<TValue>(string, CancellationToken, object?[]?)**

- Javascript Invokable
- **DotNet.invokeMethod (string, string, object[])**
- **DotNet.invokeMethodAsync (string, string, object[])**

Blazor Maui

Blazor

- Abseits vom Web, alle Betriebssysteme?



Einleitung
Laufzeitanalyse

- Blazor WebAssembly
- Blazor Server
- Qt

- Bedingungen
- Szenarien
- Ergebnisse

Bedingungen
Laufzeitanalyse

- Default Konfigurationen der IDE
- Alle Szenarien wurden auf dem Raspberry Pi implementiert und ausgeführt
- Nur die IDE und ein zusätzlicher Browser waren offen
- Jedes Szenario wurde 1000 mal ausgeführt

Szenarien
Laufzeitanalyse

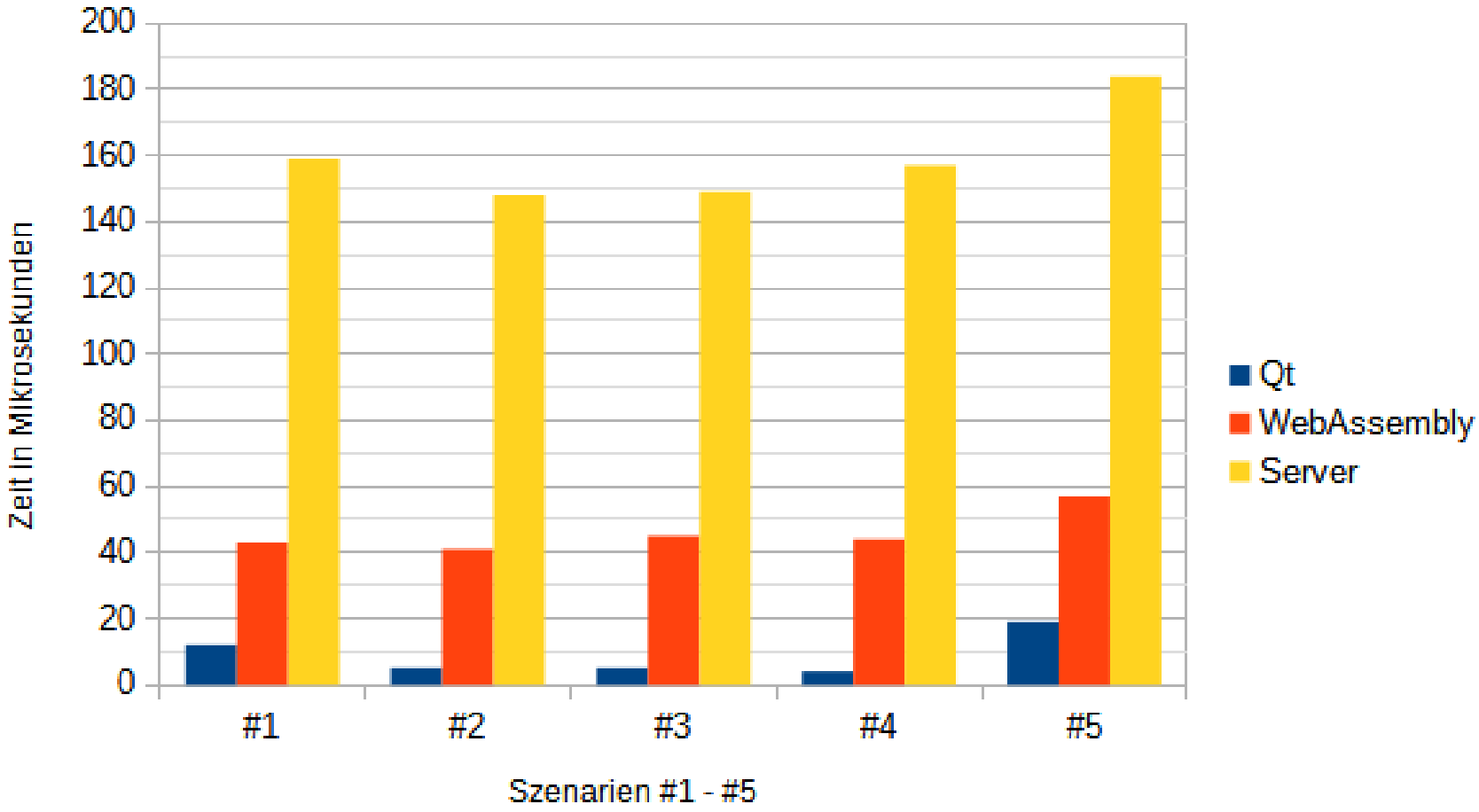
- Neun kleinere Code-Segmente
- Tabellenstruktur
- Baumstruktur

Szenarien
Laufzeitanalyse

- #1: Erzeugen einer leeren ComboBox
- #2: Erzeugen eines Labels
- #3: Erzeugen eines Buttons
- #4: Erzeugen einer CheckBox
- #5: Erzeugen einer TextBox
- #6: Erzeugen einer ComboBox mit fünf Elementen
- #7: Den Text einer TextBox lesen
- #8: Den Text eines Labels verändern
- #9: Den Text einer TextBox verändern

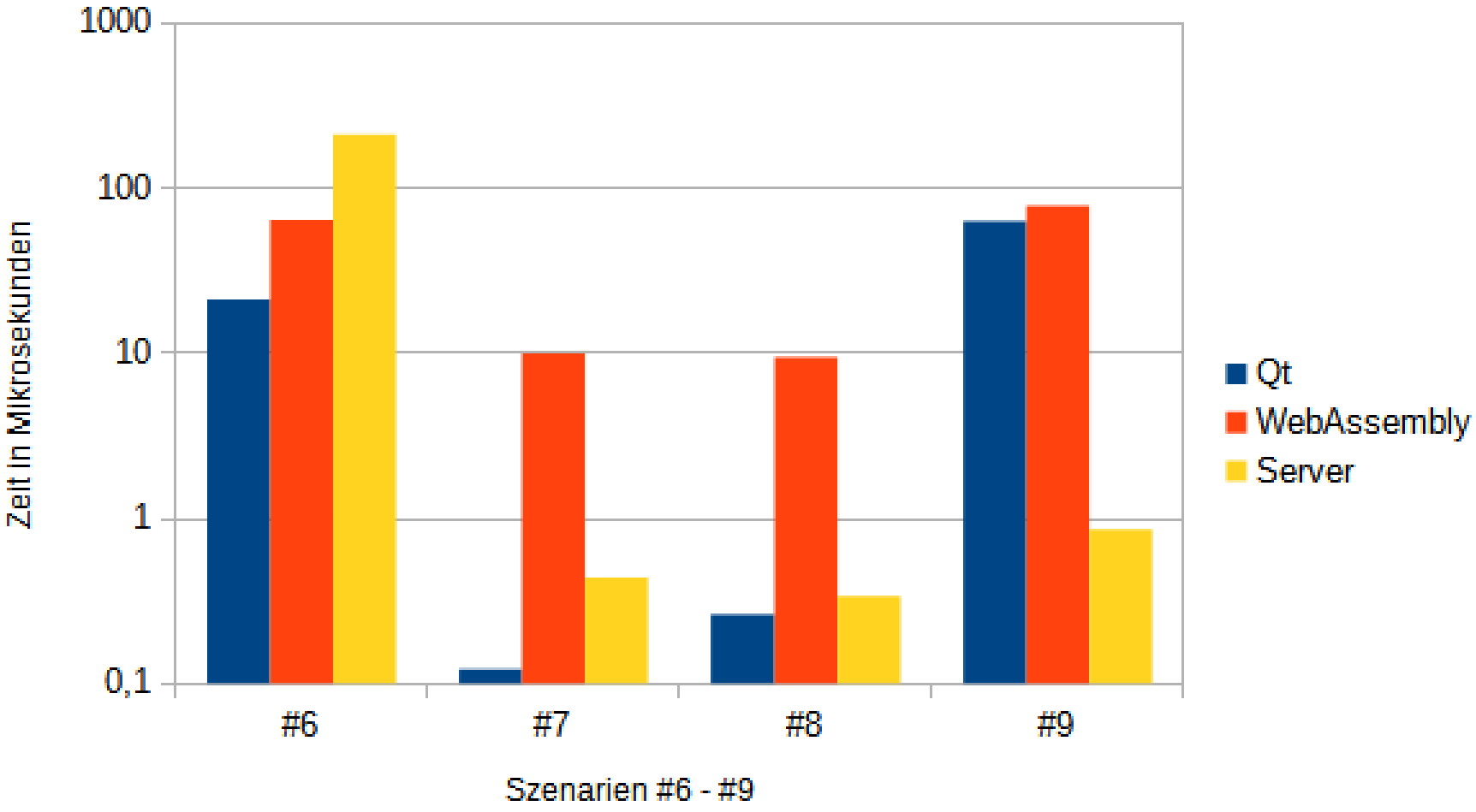
Ergebnisse – Szenarien #1 - #5

Laufzeitanalyse



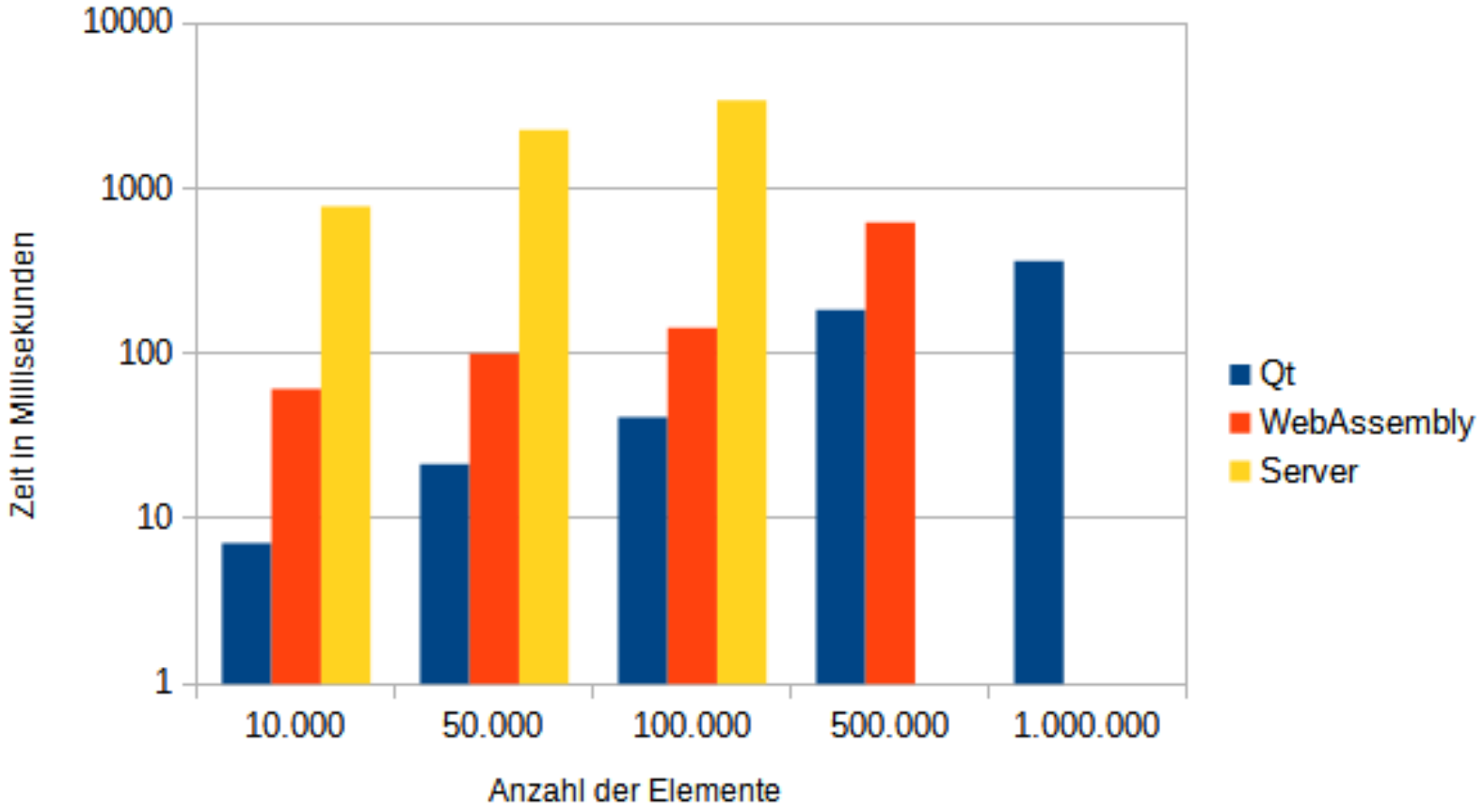
Ergebnisse – Szenarien #6 - #9

Laufzeitanalyse



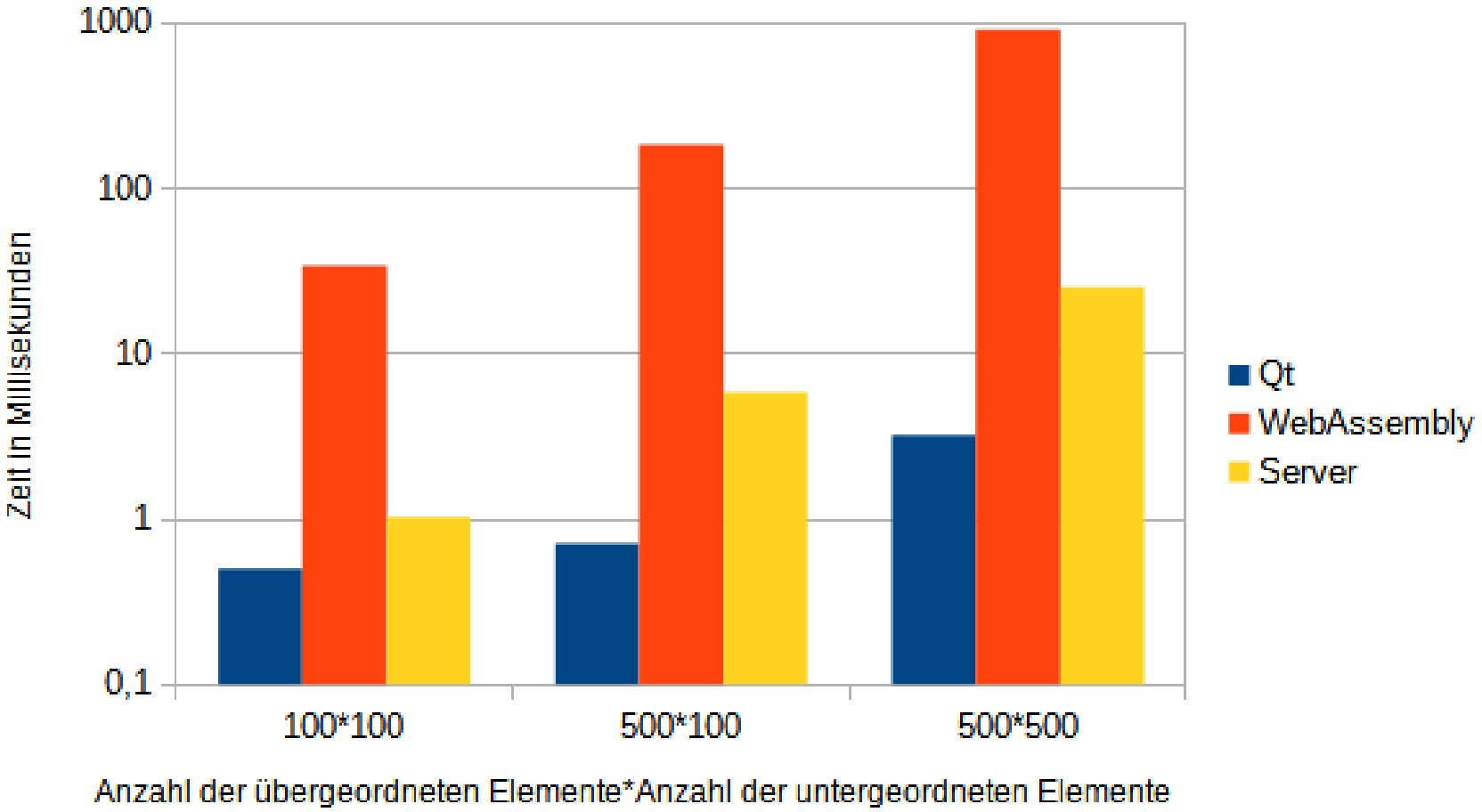
Ergebnisse – Tabellenstruktur

Laufzeitanalyse



Ergebnisse – Baumstruktur

Laufzeitanalyse



Fazit

Fazit/Ausblick

- Qt mit C++ war fast überall schneller
- Blazor ist unkomplizierter
- Blazor agiert im Web → Keine Software Installation notwendig

Ausblick

Fazit/Ausblick

- Für sehr performante Software → Qt
- Für einen schnellen Entwicklungsprozess → Blazor
- Blazor Maui