TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

# Multiplatformní aplikace v prostředí .NET

## Bakalářská práce

*Studijní program:* B2646 Informační technologie
*Studijní obor:* Informační technologie

*Autor práce:* **Michal Baňkowski**
*Vedoucí práce:* Ing. Jan Kraus, Ph.D.
Ústav mechatroniky a technické informatiky

Liberec 2022

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

## Zadání bakalářské práce

# Multiplatformní aplikace v prostředí .NET

*Jméno a příjmení:* **Michal Baňkowski**
*Osobní číslo:* M18000067
*Studijní program:* B2646 Informační technologie
*Studijní obor:* Informační technologie
*Zadávající katedra:* Ústav mechatroniky a technické informatiky
*Akademický rok:* **2021/2022**

**Zásady pro vypracování:**

1. Seznamte se s možnostmi unifikovaného vývoje aplikací pro různé cílové platformy (Windows resp. Linux desktop, webový prohlížeč a nativní mobilní aplikace) a v rešeršní části práce podrobnější porovnejte a vyhodnoťte vlastnosti a omezení různých současných nástrojů.
2. S využitím stávající .NET knihovny pro přístup k datům navrhněte a samostatně implementujte ukázkové aplikace pro mobilní správu měřicích přístrojů a základní zpracování dostupných dat.
3. Při řešení se snažte o maximální opakovatelnost využití kódu datové, business a případně i prezentační vrstvy a demonstrujte vybrané způsoby implementace grafického rozhraní pro různé výše uvedené cílové platformy.
4. Své poznatky shrňte a přehlednou formou prezentujte v závěrečné zprávě.

| | |
|---|---|
| *Rozsah grafických prací:* | dle potřeby dokumentace |
| *Rozsah pracovní zprávy:* | 30–40 stran |
| *Forma zpracování práce:* | tištěná/elektronická |
| *Jazyk práce:* | Čeština |

**Seznam odborné literatury:**


[1] BISWANGER, Gregor a Robert MUEHSIG. Build cross platform desktop apps with ASP.NET Core (Razor Pages, MVC, Blazor). *GitHub* [online]. 2021 [cit. 2021-9-28]. Dostupné z: https://github.com/ElectronNET/Electron.NET

[2] ORTINEAU, David. The New .NET Multi-platform App UI [online]. [cit. 2021-9-28]. Dostupné z: https://devblogs.microsoft.com/xamarin/the-new-net-multi-platform-app-ui-maui/

[3] RAMEL, David. First .NET 6 Preview Introduces Blazor Desktop [online]. [cit. 2021-9-28]. Dostupné z: https://visualstudiomagazine.com/articles/2021/02/17/net-6-preview-1.aspx


| | |
|---|---|
| *Vedoucí práce:* | Ing. Jan Kraus, Ph.D.<br>Ústav mechatroniky a technické informatiky |
| *Datum zadání práce:* | 12. října 2021 |
| *Předpokládaný termín odevzdání:* | 16. května 2022 |

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

doc. Ing. Josef Černohorský, Ph.D.
vedoucí ústavu

V Liberci dne 12. října 2021

# Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Jsem si vědom toho, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

11. května 2022 Michal Baňkowski

# Poděkování

Chtěl bych poděkovat vedoucímu této práce, panu Ing. Janu Krausovi, Ph.D. za cenné rady, ochotu a zpříjemnění tvorby této práce.

Dále bych chtěl poděkovat komunitě a nadšencům pro tvorbu aplikací za pomoc překonání mnoha překážek.

# Multiplatformní aplikace v prostředí .NET


## Abstrakt

Tato práce se zabývá seznámením se s vývojem multiplatformních aplikací v prostředí .NET. Zkoumá jednotlivé vývojové prostředí, které porovnává v rámci vlastností a struktury. Zmíněná prostředí korespondují s nástupem nejnovější verze technologie .NET, jenž umožňuje nový způsob tvorby multiplatformních aplikací. Výstupem práce jsou aplikace, které tuto technologii využívají pro přístup k datům měřících přístrojů a jejich zpracování. V aplikacích je kladen důraz na opakovatelnost kódu včetně využití rozdílů jednotlivých prostředí. Aplikace jsou v závislosti na podpoře prostředí testovány na platformách Windows, Linux a Android.



## Klíčová slova:

Multiplatformní aplikace, vývojové prostředí, operační systém, technologie .NET, platforma, kód, uživatelské prostředí


# Multiplatform applications in .NET enviroment


## Abstract

This work is focused on multiplatform application development using .NET platform. It studies development environments, which it compares in terms of features and structure. These environments revolve around new version of .NET technology, which creates new ways to multiplatform development. Output of this work are applications, which uses this technology in order to access data of measuring instruments and process them. The main focus is on code repeatability, including use of differences of each environment. Applications are tested on platforms Windows, Linux and Android, depending on platform support of each environment.



## Key words:

Multiplatform application, development environment, operating system, .NET technology, platform, code, user interface

# Obsah

# Seznam obrázků

# Seznam zkratek

| | |
|---|---|
| .NET | Soubor softwarových technologií |
| SDK | Software development kit |
| XAML | Extensible Application Markup Language |
| CS | CSharp Code file |
| HTML | Hypertext Markup Language |
| CSS | Cascading Style Sheets |
| MAUI | Multi-platform App UI |
| UWP | Universal Windows Platform |
| WASM | Web Assembly |

# Seznam tabulek



# Seznam zdrojových kódů

# 1. Úvod

V moderní době jsou informační technologie nedílnou součástí naší společnosti. Rozmanitost těchto zařízení je obrovská, jedná se o telefony, desktopové počítače, laptopy, chytré televize, chytré hodinky, počítače v automobilech a mnoho dalších. Každý druh zařízení je v něčem odlišný od ostatních, může se lišit hardwarem, operačním systémem nebo vstupy a výstupy. Pro tvorbu aplikací jsou tyto rozdíly problematické a nutí vývojáře pracovat přímo s daným zařízením. Aby se těmto problémům předešlo, zaměřují se vývojáři na nový způsob tvorby aplikací, multiplatformní vývoj.

Multiplatformní vývoj aplikací umožňuje sjednotit aplikaci pro mnoho odlišných zařízení. Použití tohoto způsobu umožňuje tvůrcům aplikací ušetřit čas, který by věnovali tvorbě stejné aplikace pro vícero prostředí, a také kapacitu, která vzrůstá s každým novým systémem. V rámci těchto aplikací je kladen největší důraz na kompatibilitu a rychlost aplikace. O průlom v multiplatformním vývoji aplikací se snaží společnost Microsoft se svou novou verzí technologie .NET. Technologie .NET je již součástí velkého množství zařízení a mnoho aplikací s touto technologií pracuje. [1]

Tato práce se zabývá touto technologii a její nejnovější verzi .NET 6. Zaměřuje se na nové a vysoce očekávané vývojové prostředí .NET MAUI i na již funkční prostředí Blazor a Xamarin, které díky nové verzi této technologie zaznamenaly změny. Mimo produkty značky Microsoft se práce také zaměřuje na vývojová prostředí třetích stran, jenž zastupuje platforma Uno Platform. Každé vývojové prostředí se v mnohém liší a používá unikátní způsob tvorby aplikací.

Výstupem práce jsou tři aplikace, které se zaměřují na unikátní prvky svých prostředí. Vytvořené aplikace umožňují pracovat se vstupním souborem, jenž obsahuje konfigurační data měřících přístrojů elektrické energie. Jejich vzhled a chování napodobuje již existující programy, které kladou důraz na jednoduché používání, výkon a stabilitu. Každá z aplikací obsahuje vlastní uživatelské rozhraní zaměřené na grafické schopnosti jednotlivých prostředí. Součástí aplikací je samostatná knihovna, jenž obstarává výstupní aplikace bez úprav pro daný systém.

# 2. Multiplatformní vývojové prostředí

Pro tvorbu multiplatformních aplikací je nejprve potřeba nalézt vhodné vývojové prostředí, poskytující multiplatformní tvorbu. Zprovoznění jedné aplikace na odlišných platformách není jednoduché, jelikož je zapotřebí pracovat se zařízeními, jenž se navzájem liší strukturou i operačními systémy. Řešení poskytující jednotný kód pro všechna zařízení se snaží poskytnout společnost Microsoft za pomoci technologie .NET, která umožňuje exekuci jednotného kódu pro všechna zařízení s podporou technologie .NET.

Rozdíly vícero platforem se však objevují nejenom v softwarovém prostředí, ale také v přístupu k hardwaru. Vstupní a výstupní periferie zařízení se značně liší s platformou. Mezi tyto periferie patří například gyroskop, dotykový displej nebo kamera v rámci mobilních zařízení, a naopak pro desktopové zařízení se může jednat o klávesnici, USB port nebo touchpad. Schopnost pracovat s odlišnými vstupními a výstupními prvky není jediným důležitým faktorem. V rámci softwarového zpracování aplikace je také zapotřebí zajistit responzivitu a aplikace správně zobrazit. Mnoho odlišných zařízení používá jiný poměr displeje nebo monitoru, který ovlivňuje také rozlišení. Vzhledem k růstu popularity a nárokům pro tvorbu multiplatformních aplikací vzrůstají i požadavky pro tvůrce těchto systémů a výsledné prostředí poté může být uspěchané a obsahovat mnoho nedostatků, ba naopak chytře vymyšlené a vhodnou volbou na trhu pro vývojáře.

Následující vývojová prostředí pracují se systémem .NET verze 6, jenž je vzrůstající novinkou zaměřenou právě na multiplatformní tvorbu. S nástupem této verze vznikla i nová multiplatformní vývojová prostředí od společnosti Microsoft, která ukazují možnosti práce s technologií .NET, včetně starších vývojových prostředí, jež jsou o tuto změnu obohaceny. Příchod nové verze technologie .NET zaznamenaly i menší vývojářské skupiny a společnosti, které používají tuto novou verzi pro své vlastní řešení multiplatformního prostředí a doplňků. Následující prostředí jsou nejpopulárnějšími a nejznámějšími nástroji pro práci s touto technologií.

## 2.1. Blazor

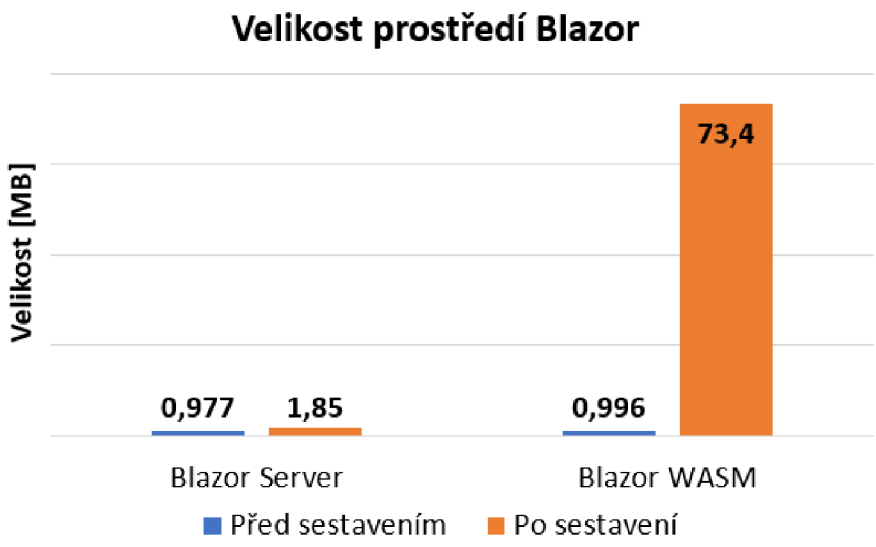
Nástroj Blazor je součástí open-source webové technologie ASP.NET Core od společnosti Microsoft. Tato webová technologie umožňuje tvorbu webových aplikací, které mohou být provozovány na systémech Windows, Linux a macOS. Webové aplikace Blazor umožňují hostování aplikací přímo v cloudu v několika režimech a jsou ojedinělé svým zpracováním. Pracují se softwarovou architekturou Model-View-Controller, jenž rozděluje datový model, uživatelské rozhrání a řídící logiku aplikace. Technologie APS.NET Core pracuje nejen se svým nástrojem Blazor, ale také s konkurenčními technologiemi. Mezi tyto technologie a knihovny se řadí Angular od společnosti Google, React od společnosti Meta a také Bootstrap od Bootstrap Core Team. [2]

### 2.1.1. Vlastnosti

Technologie Blazor nabízí provozování webové aplikace v podobě dvou hostitelských modelů. Prvním modelem je Blazor Server. Tento model je zaměřený na zpracovávání dat na straně webového serveru. Klient posílá požadavky na server a ten se poté snaží udržet s klientem spojení za použití ASP.NET Core softwarové knihovny SignalR a při úspěšném spojení posílá klientovi požadovaná data. Pro veškerou práci v serverové aplikaci je zapotřebí udržovat si neustálé připojení a klient se musí pravidelně ověřovat. Toto řešení obsahuje mnoho výhod a také nevýhod. Mezi hlavní výhody se řadí velice rychlé načítání a rychlost zpracování složitých dat, která závisí pouze na výkonu serveru, nikoli klientského zařízení. Problém však může nastat při přetížení serveru, kde veliké množství dat může server zpomalit, čímž se zvětší odezva pro všechny připojené klienty. Z toho důvodu jsou nároky na server vyšší, což zvýší i náklady na jeho provoz. Další omezení souvisí s bezpečností aplikace. V modelu Blazor Server se data nachází bezpečně na serveru mimo dosah klienta. Klient si může o data zažádat, ale obdrží vždy pouze žádaná data, nikoli všechna data aplikace. Webová aplikace s tímto modelem tak neposkytuje offline podobu a v případě výpadku sítě nebo ztráty signálu neumožňuje aplikaci poskytovat. Tento problém však řeší druhý hostitelský model. [3]

Druhým modelem je Blazor WebAssembly, zkráceně WASM. Tento model využívá server pro posílání sestaveného balíčku s aplikací klientovi. Obsah balíčku včetně veškerých dat aplikace je poté zobrazen klientovi ve webovém prohlížeči. Jelikož server

poskytuje klientovi balíček obsahující celou aplikaci, nelze kompletně zamezit přístup k datům aplikace, a proto toto řešení není vhodné pro aplikace uchovávající citlivá data. V případě, kde citlivá data vlastní klient a přidává jej do aplikace, je tato metoda velice užitečná a nabízí spoustu výhod. Hlavní výhodou je možnost používat staženou aplikaci v offline podobě. Webovou aplikaci lze také posílat po síti pomocí služby CDN (Síť pro rozprostření obsahu) bez nutnosti ASP.NET Core webového serveru. Rychlost zpracování dat také může být vyšší pro klienta, jelikož není omezen více požadavky včetně nutnosti udržování spojení, jako v případě modelu Blazor Server. Tyto výhody však silně ovlivňují kapacitu, která je pro Blazor WebAssembly mnohonásobně větší (viz Obrázek 1). [3]




**Obrázek 1:** Velikost prostředí Blazor v závislosti na hostitelském modelu

Nadcházející variantou je použití hostitelského modelu Blazor Hybrid. Tento nový model se vyvíjí s příchodem technologie .NET verze 6 a nabízí způsob tvorby nativních aplikací pomocí webového formátu. Aplikace s modelem Blazor Hybrid obsahuje strukturu Blazor, ale operuje ve formě nativní aplikace pro dané zařízení, nikoli webové. Pro tento úkol používá prvky ze systému .NET, včetně technologie, která promítá webové prostředí do nativní aplikace. Tvorba uživatelského prostředí není omezena na webové prvky a umožňuje přidat do aplikace také prvky nativní. Mezi tyto grafické prvky se řadí prvky ze sad Windows Forms, .NET MAUI, nebo také WPF. První úspěšný pokus hybridní Blazor aplikace je obsažen v kapitole 2.3 [3]

Všechny hostitelské metody poskytují řadu funkcí a vlastností pro webové aplikace operující na systémech Windows, Linux a macOS se snahou vyvinout novou metodu podporující nativní prostředí. Díky systému .NET si lze tyto aplikace spustit v jakémkoliv zařízení, které tento systém podporuje. Souhrn těchto vlastností je uveden v tabulce, viz Tabulka 1.

**Tabulka 1:** Porovnání hostitelských modelů Blazor [3]

| Vlastnost | Blazor Server | Blazor WASM | Blazor Hybrid |
|---|---|---|---|
| Plně kompatibilní .NET API | ANO | NE | ANO |
| Přímý přístup k serveru | ANO | NE | NE |
| Malý přenos dat | ANO | NE | NE |
| Zabezpečení kódu na serveru | ANO | NE | NE |
| Offline verze | NE | ANO | ANO |
| Statické hostování | NE | ANO | NE |
| Zpracování dat u klientů | NE | ANO | ANO |
| Přístup k nativním prvkům | NE | NE | ANO |

### 2.1.2. Struktura

Blazor má jedinečnou strukturu, která využívá mnoho základů webové tvorby a přidává k ní použití .NET systému společně s jazykem C#. Jazyk C# slouží jako zástupce běžně používaného jazyku JavaScript a zároveň umožňuje přístup ke knihovnám .NET. Tvorba uživatelského prostředí je řešena pomocí kódu HTML, podporující HTML 5 včetně všech dynamických elementů. Grafické úpravy elementů jsou standardně řešeny kaskádovými styly CSS. Dynamické úpravy a ovládání je řešeno pomocí kódu C# a společně s HTML a CSS tvoří jednotlivé komponenty Blazor aplikace. [4]

Souborová struktura je v Blazor aplikacích řešena pomocí komponent. Komponenta je soubor formátu razor, která zabaluje HTML, CSS a C# kód do jednotné struktury. Tyto komponenty jsou pro webovou aplikaci součástí stránky, ve které jsou od sebe odděleny a neovlivňují se navzájem. Komponentou tedy může být záhlaví nebo zápatí stránky, menu obsahující navigaci nebo jednotlivé obsahy stránek. Blazor pomocí této struktury

nepoužívá odkazování na další stránky, ale mění pouze komponenty uvnitř aplikace a izoluje tak všechny části webu. Toto řešení umožňuje velice přehledně operovat pouze s danou částí webové aplikace, bez narušení okolního zobrazení. Aby bylo možné umístit komponenty do aplikace, je nutné prvně zajistit vhodné rozložení neboli prostor, kde se dané komponenty mohou promítat bez kolizí mezi sebou. [5]

### 2.1.3. Shrnutí

Prostředí Blazor je unikátním řešením umožňující tvorbu webových aplikací s použitím jazyku C# a přístupu k .NET knihovnám. Toto prostředí lze operovat na operačních systémech Windows, Linux a macOS a s technologií .NET lze zobrazit obsah těchto aplikací nejen na těchto systémech, ale na všech zařízení s podporou systému .NET. Nová verze technologie .NET pomáhá tomuto prostředí se dál rozvíjet a šířit svůj způsob tvorby v nové podobě hostitelského modelu Blazor Hybrid, jenž má potenciál změnit způsob tvorby nativních aplikací.

Struktura Blazor aplikace umožňuje izolovat tvorbu jednotlivých částí aplikace s minimálním dopadem na celek, avšak rozvržení aplikace musí být bez chyb, což celkový koncept jednoduché izolace značně komplikuje. Tvorba webových aplikací také vyžaduje pokročilé schopnosti v tvorbě webového prostředí a programování, včetně znalosti technologie .NET. Pro nezkušené vývojáře je vlastní rozložení aplikace velice složité na realizaci vzhledem k tomu, že Blazor aplikace obsahuje svoji vlastní verzi technologie Bootstrap, ze které primárně čerpá. Veškerá volně dostupná rozložení pro webové aplikace nezohledňují strukturu Blazor a většinu je nutné značně předělat, nebo je nelze použít.

## 2.2. .NET MAUI

.NET Multi-platform App UI, zkráceně MAUI, je nový open-source nástroj od společnosti Microsoft, který umožňuje tvorbu mobilních a desktopových aplikací v jednotném prostředí. Nástroj MAUI vznikl společně s příchodem šesté verze technologie .NET a je jejím hlavním představitelem. Zaměřuje se na největší pokrok této verze a tou je schopnost sjednotit strukturu aplikace pro různá zařízení s lišícím se operačním systémem. Umožňuje tvorbu aplikací pro operační systémy Android, iOS, macOS a Windows. Aktuálně je tento nástroj stále ve vývoji a je možné, že zmíněné vlastnosti nebudou korespondovat s aktuální verzí. [6]

### 2.2.1. Vlastnosti

.NET MAUI vznikl ze systému Xamarin, jenž je mnoho let používaný nástroj pro tvorbu mobilních aplikací a je popsán v kapitole 2.5. Oba tyto nástroje se nadále rozvíjejí nezávisle na sobě i přestože jsou si velice podobné a sdílí stejné technologie. Pokud bude nástroj .NET MAUI úspěšný, může nahradit Xamarin a stát se prioritním prostředím společnosti Microsoft pro tvorbu mobilních a zároveň i multiplatformních aplikací. V rámci technologie .NET používá .NET MAUI jednotné API pro všechny podporované operační systémy. Toto prostředí pracuje s prvky, které lze zobrazit ve všech zmíněných operačních systémech s jejich přirozeným vzhledem a zároveň zpracovává veškerý kód na zařízeních s podporou technologie .NET. Grafické prvky se v každém prostředí tváří jinak dle svého systémového vzhledu, ať už se jedná o tlačítka nebo ukazatele času, ale jsou vytvořeny a ovládány jednotným kódem. [7]

I přestože kód je sdílen pro všechny systémy, každý systém vyžaduje vhodné zabalení aplikace do balíčku pro své prostředí. Aplikace lze poté do zařízení nainstalovat a ze zařízení spustit. Tento problém řeší .NET MAUI s použitím systémových balíčků .NET SDK pro každý operační systém. Balíčky mají přístup ke sjednocené knihovně .NET 6 Base Class Library, odkud je možné jednotně získávat data pro každou platformu. [7]

Stejně jako lze sjednotit kód a jeho funkčnost pro všechny platformy, umožňuje .NET MAUI kód také pro různé platformy oddělit. Některá zařízení mohou mít jiné vstupní periferie vzhledem k systému a hardwaru, jenž používají. V takovém případě je nutné specifikovat cílenou platformu, která má k těmto periferiím přístup. Jedná se především o čtečky otisku prstu, gyroskop nebo kompas. [7]

### 2.2.2. Struktura

Struktura .NET MAUI pochází ze systému Xamarin. Obdobně jako Xamarin používá pro tvorbu aplikace jazyk C# společně s XAML, který obsahuje veškeré grafické prvky a jejich vlastnosti. Tyto grafické prvky se syntaxí podobají nejen systému Xamarin, ale také mnoha dalším .NET nástrojům, což vývojářům usnadňuje přechod z jiného prostředí. Na rozdíl však od ostatních prostředí, .NET MAUI neposkytuje grafický editor a veškerý vzhled je tvořen manuálně. Veškeré chování aplikace je zachováno v souborech CS, které používají jazyk C# a základy objektově orientovaného programování pro vykonávání úkonů v aplikaci. Kód v těchto souborech může napřímo pracovat s grafickými prvky a korespondovat na vstup uživatele. [8]

Prostředí .NET MAUI aplikace je rozděleno na jednotlivé stránky, kde každá stránka obsahuje společně kód XAML a CS. Cizí zdroje, jako jsou obrázky, fonty nebo další soubory, jsou uloženy v oddělené složce, kde se na ně odkazuje v kódu stránky. Tyto odloučené soubory se poté spolu s kódem zabalí do vhodného balíčku pro každou platformu. Při ukládání souborů do balíčku se data upravují dle platformy a cílového zařízení, příkladem úpravy může být například komprese. [9]

### 2.2.3. Shrnutí

.NET MAUI je velice očekávané a populární vývojové prostředí, ale stále se jedná o nástroj, který je ve vývoji. Nabízí nový způsob unifikované tvorby aplikací pro různé platformy, který může silně oslovit vývojáře mobilních aplikací, ale pro úspěch se potřebuje zbavit mnoha nedostatků, které toto vývojové prostředí doprovází. I přes veškerou snahu nové verze technologie .NET není tvorba aplikace jednoduchá a vyžaduje mnoho zásahů pro zajištění kompatibility napříč všemi zařízeními.

Ve vývojovém prostředí neexistuje grafický editor, což je pro tvorbu aplikace velice zásadní. Manuální tvorba uživatelského prostředí používá sice jednotné API, což pomáhá zjednodušit tvorbu aplikace, ale trpí v mnoha grafických ohledech. Počet prvků, jenž lze aktuálně použít, je velice malý a v případě operačního systému Windows vypadají mnohé prvky velice nevhodně z toho důvodu, že se prvky používají spíše na mobilních zařízení, kde mají vhodnější reprezentaci. Platformy, jako jsou Windows a macOS, nejsou primárně zaměřené na tyto mobilní prvky ve svém systému, což vede k tomuto nedostatku. Z toho důvodu je vhodné použít grafické balíčky cizích firem a

šikovných autorů, které poskytují vlastní reprezentaci grafických prvků a nespoléhat se na nativní grafické zobrazení.

I přes mnoho nevýhod má .NET MAUI také svoji světlou stránku. Díky své XAML a CS struktuře je možné řešit složitější úlohy, vyžadující vyšší programátorské schopnosti. Tato struktura se zaměřuje na objektově orientované programování a umožňuje nejen pracovat s grafickou stránkou, ale také se složitými daty. Dokáže pracovat s daty v rámci modelů, primárně modelu MVVM. Použití datového modelu a struktury XAML a CS souborů umožňuje autorům z jiných vývojových prostředí snadno migrovat aplikaci do .NET MAUI a poskytovat ji i na jiných operačních systémech.

## 2.3. .NET MAUI Blazor

.NET MAUI Blazor je hybridní podoba prostředí .NET MAUI, která sjednocuje .NET MAUI a Blazor. Sloučení těchto prostředí umožňuje tvorbu desktopových a mobilních aplikací pro platformy iOS, Android, macOS a Windows. Cílené platformy a způsob předání aplikace mezi různá zařízení je identické s prostředím .NET MAUI, jedná se o kompilaci balíčků z technologie .NET, která se potom předá v instalační nebo spustitelné formě do zařízení. Na rozdíl od běžné .NET MAUI aplikace je uživatelské prostředí tvořeno pomocí HTML kódu společně s použitím kaskádových stylů, stejně tak jako v aplikacích Blazor. V tomto případě není použit server a za pomoci dostupných knihoven je webové prostředí promítáno v podobě nativní aplikace. Tento nástroj koresponduje s hostitelským modelem Blazor Hybrid, jenž je popsán v kapitole 2.1.1. [10]

### 2.3.1. Vlastnosti

Sjednocení dvou vývojových prostředí .NET MAUI a Blazor je možné pomocí prvku BlazorWebView. Tento prvek dokáže zobrazit webové rozhrání souborů razor v nativní podobě prostředí .NET MAUI [11]. Vzhledem k univerzálnosti webové technologie a technologie ASP.NET Core se očekává, že s příchodem dalších platforem v prostředí .NET MAUI se bude rozšiřovat počet platforem i pro tento hybridní nástroj. Veškerý kód je tvořen pomocí HTML, CSS a C#, stejně jako v prostředí Blazor, s možností použití nativních prvků. Obdobně jako je tomu v Blazor WASM aplikaci, jsou i zde veškerá data v rámci stylů a responzivity uložena v Blazor struktuře. Ostatní data, jako jsou obrázky, loga, videa a další externí soubory, jsou obsažena v balíčcích společně s daty Blazor. [10]

I přestože má tento nástroj za úkol zamezit problémům, s nimiž se .NET MAUI potýká, jako je například špatná tvorba uživatelského prostředí, obsahuje i nedostatky, které souvisí s nástrojem Blazor. Aplikační logika, která je v .NET MAUI řešena pomocí CS souborů, není pro Blazor část aplikace použitelná. Z pohledu Blazor aplikace se však nabízí mnoho výhod. Pro přístup k aplikaci již není potřeba použít server. Aplikace má přístup k zařízení a možnost pracovat s jeho prvky, jako jsou senzory nebo kamera. Rychlost aplikace se také může výrazně navýšit vzhledem k tomu, že server zde nehraje

roli v rychlosti, a oproti WASM webové aplikaci zde nepřichází zpomalení ze strany prohlížeče. [10]

### 2.3.2. Struktura

Základ struktury prostředí vychází z .NET MAUI, obsahuje shodné načítací příkazy, používá technologii .NET verze 6 společně s jejími SDK balíčky, tvoří XAML a CS základ aplikace. BlazorWebView je pro .NET MAUI aplikaci zobrazena jako běžná funkce, která zprostředkovává zobrazená data. Pro svoji funkci vyžaduje hostitelskou stránku, jenž je hlavní načtená stránka, podobně jako stránka index ve webové aplikaci. Tato stránka obsahuje čisté HTML odkazující na JavaScript, který umožňuje zobrazit formát Blazor. Krom hostitelské stránky vyžaduje BlazorWebView také odkaz na klíčové komponenty, které tvoří rozložení aplikace. [12]

Zbylé soubory tvoří pouze vnitřní strukturu aplikace a obdobně jako v prostředí Blazor jsou tvořeny razor soubory. Tyto soubory se skládají z HTML kódu společně s kaskádovými styly a použitím C# kódu. Rozložení složek a jejich obsahů je s prostředím Blazor téměř identické, jedná se o vnořený prostor uvnitř běžné .NET MAUI aplikace, který částečně odlišuje struktury obou prostředí. [12]

### 2.3.3. Shrnutí

Kombinace vývojových prostředí .NET MAUI a Blazor umožňuje nový způsob tvorby nativních aplikací. Tento způsob může oslovit mnoho webových vývojářů a pomoci zjednodušit tvorbu pro desktopová a mobilní zařízení. Nejedná se však o změnu .NET MAUI, ale pouze o jeho rozšíření. Použití prvků ze dvou prostředí umožňuje mnohem více úkonů ze strany webové aplikace a také více grafických úprav ze strany nativní aplikace.

Veškerá responzivita a vzhled aplikace na různých rozlišení je zajištěna technologií Bootstrap. Další prvky aplikace jsou uložené společně s aplikací v jednotném balíčku lokálně v zařízení. Balíčky z prostředí .NET MAUI jsou obohaceny o dodatečné soubory prostředí Blazor, včetně webových knihoven.

## 2.4. Uno Platform

Uno Platform je multiplatformní vývojové prostředí vytvořené společností nventive společně s komunitou nadšenců do aplikační tvorby. Umožňuje tvorbu aplikací v podobě webové WASM aplikace, desktopové aplikace pro Windows, Linux, macOS a mobilní aplikace pro iOS a Android s jednotným kódem pro všechny zmíněné platformy. Jedná se o první a jediné multiplatformní prostředí, které umožňuje vývoj aplikací pro tolik operačních systémů zároveň, včetně možnosti webové aplikace. Tato open-source platforma podporuje mnoho balíčků třetích stran a je vhodná pro finální vývoj. Oproti konkurenčním prostředím je Uno Platform dlouholeté prostředí, jenž prošlo za poslední roky mnoha změnami a nyní nabízí multiplatformní tvorbu v prostředí šesté verze technologie .NET. [13]

### 2.4.1. Vlastnosti

Vzhledem k tomu, že se nejedná o prostředí vázáné se společností Microsoft, je obsaženo v podobě samostatného balíčku. Uno Platform lze provozovat v podobě .NET aplikace nebo v podobě Xamarin. I přesto, že se jedná o méně známou platformu, používá mnoho funkcí obsažených v konkurenčních prostředích a snaží se udržovat co nejpodobnější strukturu aplikace. Toto prostředí má přístup k mnohým vývojovým nástrojům jako je Windows Community Toolkit nebo Hot Reload. [13]

Uno Platform umožňuje tvořit vzhled uživatelské rozhraní dvěma způsoby. Prvním způsobem je použití nativního zobrazení. Nativní zobrazení umožňuje zobrazit daný prvek dle operačního systému. Primárně se jedná o kalendáře, hodiny a jim podobné prvky, které se rozdílně zobrazují v odlišných operačních systémech, zejména Android a iOS. Toto řešení zobrazení používá vývojové prostředí .NET MAUI a je velice užitečné pro tvorbu mobilních aplikací, ale v kombinaci s desktopovým prostředí je silně nevhodné. Uno Platform používá vykreslení uživatelského rozhraní na kreslící plochu. Plocha obsahuje prvky, jež jsou graficky i chováním identické, nehledě na platformě. Toto řešení se nazývá Pixel Perfect Rendering a zajišťuje příjemně vypadající vzhled pro všechna zařízení. [14]

### 2.4.2. Struktura

Uno Platform prostředí je tvořeno pomocí XAML a CS souborů. Tyto soubory napodobují veškeré známé prostředí .NET a mají i společnou syntaxi. Rozložení těchto souborů je tvořené podle šablony Uno Platform, která zajišťuje oddělení kódu aplikace od náležitostí pro každou použitou platformu. Kód lze tímto způsobem snadno izolovat, ale také může způsobit problém, kdy aplikace funguje správně jen na určitých zařízeních, která podporují danou funkci. [15]

Mnoho různých platforem vyžaduje mnoho různých balíčků. Pro mobilní a Apple platformy používá Uno Platform balíčky .NET SDK, pro operační systémy Android, iOS, MacCatalyst a MacOS. Tyto balíčky společně s knihovnami UNO tvoří instalační a spustitelnou podobu aplikace pro tyto operační systémy. Tvorba aplikace pro platformu Linux vyžaduje knihovny .NET společně s grafickými nástroji, jako je grafická knihovna Skia a editor GTK. V případě operačního systému Windows lze balíček vytvořit dvěma způsoby. Aplikaci lze sestavit s použitím platformy UWP nebo pomocí kombinace nástrojů Skia společně s technologií WPF. Webová aplikace v prostředí Uno Platform existuje pouze v podobě WASM aplikace, kde veškeré zpracování zajišťuje technologie ASP.NET Core.

### 2.4.3. Shrnutí

Uno Platform je jediným vývojovým prostředí, které dokáže propojit webové, mobilní a desktopové platformy v jednom kódu. Umožňuje přístup k mnoha balíčkům třetích stran, ať už se jedná o více grafických prvků, přístup k načítání dat nebo tvorbu grafů. Poskytuje tvorbu aplikací vhodných do produkce, kde je zajištěna bezpečnost a stabilita.

Díky technologii Pixel Perfect zajišťuje Uno Platform libovolný vzhled aplikací, bez omezení pro operační systém. Vykreslování prvků zajišťuje plynulejší interakce, vyšší kvalitu aplikace a zároveň jednoduší kód. Aby tato technologie fungovala, je však zapotřebí zprovoznění grafických nástrojů, které se liší v platformách. Vzhledem k tomu, že je toto prostředí obsaženo pouze jako balíček s nástroji, může být obtížné získat veškeré potřebné soubory a řešit vzniklé problémy se znatelně menší podporou, než poskytují produkty Microsoft.

## 2.5. Xamarin

Xamarin je dlouholeté vývojové prostředí umožňující tvorbu mobilních aplikací pro platformy Android a iOS. Jedná se o open-source prostředí, jenž se stalo základem vývoje mobilních aplikací. S narůstající popularitou multiplatformního vývoje umožňuje již i Xamarin multiplatformní tvorbu aplikace, společně s platformou Windows. V rámci aplikační tvorby poskytuje tento nástroj hotové prostředí pro tvorbu mobilních aplikací včetně neobvyklých platforem, jako jsou chytré hodinky nebo chytré televize. Co se týče multiplatformní podoby, a to zejména desktopu, je toto nové odvětví pouze v experimentální podobě. [16]

### 2.5.1. Vlastnosti

Tvorba multiplatformní aplikace je v prostředí Xamarin tvořena technologií Xamarin.Forms. Tato technologie je zaměřena na nativní prostředí pro každé zařízení s možností sdílení stejného kódu za pomoci technologie .NET. Sdílení kódu umožňuje tvořit jednotné nativní prostředí, které spoléhá na kvalitu systémového zobrazení prvků. Pro mobilní platformy je použití nativního zobrazení značnou výhodou, vzhledem k jednoduchosti prvků a snadnému vstupu pomocí dotykového displeje, avšak pro desktopové prostředí je nativní zobrazení silně nedostatečné. I při snaze tvorby dotykového prostředí Windows a tvorby mobilní podoby této platformy, jsou grafické nedostatky prvků znatelné, včetně omezení úprav a jejich vlastností. [17]

Jelikož se Xamarin zaměřuje na tvorbu aplikace v nativním prostředí, dokáže také pracovat s jedinečnými prvky daných zařízení. Obsahuje knihovnu Xamarin.Essentials, která stejně jako v převzaté podobě v .NET MAUI dokáže pracovat s prvky zařízení jako například akcelerometr, souborovým systémem, informacemi o zařízení, zámku obrazovky a podobně. Tato knihovna je plně podporována všemi platformami Xamarin.Forms a částečně podporována ostatními platformami jako jsou například watchOS a Tizen, kde je přístup k těmto prvkům omezený. [18]

### 2.5.2. Struktura

Multiplatformní aplikace Xamarin je složena ze dvou částí. Hlavní část je sdílený projekt, ze kterého všechny platformy čerpají. Tento projekt obsahuje veškeré náležitosti jako je aplikační logika, obsah aplikace a tvorbu uživatelského rozhrání.

Uživatelské rozhrání aplikace je tvořeno pomocí kódu XAML společně se zdrojovými kódy, jež tvoří aplikační logiku. Díky této struktuře lze tvořit aplikace na základech objektově orientovaného programování a aplikačních modelů, primárně modelu MVVM. Druhou částí jsou stavební projekty určené pro jednotlivé platformy. Tyto projekty jsou odkázány na sdílený obsah, ale zajišťují konstruktory a přístup ke všem knihovnám pro kompilování v daném prostředí. [19]

### 2.5.3. Shrnutí

Nástroj Xamarin vznikl za účelem tvorby mobilních aplikací. Vychází z něj a jeho přístupu k technologii .NET mnoho nových a aktuálnějších vývojových prostředí, což jej dělá méně populární volbou. I přesto však nabízí způsoby pro tvorbu multiplatformních aplikací se strukturou, které se drží i konkurenční produkty. Vzhledem k dlouholetému vývoji je toto prostředí již optimalizované pro své účely a obsahuje stabilní základ pro aplikace, včetně rozsáhlé dokumentace a podpory ze strany vývojářů.

Kde Xamarin strádá jsou však nové technologie a počet platforem, které podporuje. Jedná se pouze o mobilní platformy s možností použití operačního systému Windows, který se teprve adaptuje pro mobilní aplikace. Tento omezený počet platforem dělá z Xamarinu spíše mobilní vývojové prostředí nežli multiplatformní, i přes veškerou snahu technologie Xamarin.Forms.

Pro tvorba grafických prvků je zde použito nativního zobrazení. Toto řešení má své výhody, jako je jednotné zobrazení a ušetření práce s grafikou prvků, která je dána systémem. Bohužel tyto výhody vedou k omezení grafických nástrojů a nutí tvořit vzhled aplikace tak, aby vhodně korespondoval s každou platformou.

## 2.6. Porovnání vývojových prostředí

Všechna zmíněná prostředí sdílejí mnoho vlastností, mezi které patří primárně přístup k technologii .NET 6 a dalších .NET knihoven. Tato technologie umožňuje všem prostředím spustit aplikaci na libovolném zařízení, a to v podobě reálného zobrazení aplikace s pomocí .NET API nebo vytvoření balíčku, kde je kód a struktura aplikace upravena pro dané zařízení. Princip je u všech zmíněných vývojových prostředí identický. Kód aplikace je sdílen pro všechny aplikace, každá platforma si tento kód upraví do vhodné podoby a poté výslednou aplikaci spustí na svém zařízení.

### 2.6.1. Vývojové prostředí

V případě prostředí .NET MAUI, Uno Platform a Xamarin je veškerá tvorba aplikace zajištěna pomocí značkovacího jazyku XAML, společně s aplikační logikou v prostředí zdrojového kódu CS. Tato tvorba je běžně užívaná pro vývoj .NET aplikací i mimo multiplatformní prostředí.

V rámci programování chování aplikace je tato struktura velice populární, vzhledem k poskytnutí vhodného prostředí pro objektově orientované programování, které společně s vhodným IDE tvoří základ pro práci se složitými daty. Nevýhodou tohoto řešení je však snadný přístup k nativním grafickým prvkům z knihoven .NET, který mnohdy autoři multiplatformních prostředí používají, místo snahy o vlastní způsob tvorby grafických prvků. Napříč mnoha platformami existuje pouze malý počet prvků, které lze použít jednotně pro více platforem a co je hlavní, každý tento prvek vypadá na jiné platformě jinak, což je problematické pro tvorbu designu aplikace.

Z tohoto důvodu vytvořilo prostředí Uno Platform vlastní technologii, kde veškeré grafické prvky vykresluje. Vlastní vykreslení prvků aplikace umožňuje plnou kontrolu nad designem aplikace, bez úhony v jiné platformě. Alternativní způsob vykreslení aplikace používá prostředí .NET MAUI Blazor, kde je design aplikace tvořen jazykem HTML a veškerá struktura aplikace na něj navazuje. Tento způsob tvorby aplikací je dosud nevídaný a umožňuje zjednodušit tvorbu designu aplikací nebo i migrovat webové aplikace mezi desktopové a mobilní zařízení.

Pro vývojové prostředí není důležitá pouze tvorba aplikace, ale je také zapotřebí vzít v potaz rychlost, kapacitu a veškerá omezení jednotlivých prostředí. Tyto vlastnosti

silně závisí na cílových platformách a zařízeních, která kód aplikace vykonávají. Všechna vývojová prostředí .NET používají podobné nástroje a čerpají ze stejných knihoven, avšak značně se liší v množství platforem, které podporují. Některá prostředí nezamýšlejí rozšířit počet podporovaných platforem a zaměřují se pouze na své dosavadní platformy. Jedná se primárně o platformy od Microsoftu, používající nativní prvky, které se zdají být příčinou malého počtu podporovaných platforem. Srovnání podporovaných platforem je obsaženo v Tabulka 2.

**Tabulka 2:** Porovnání podporovaných platforem vývojových prostředí

| Vývojové prostředí | Web | Desktop | Mobile |
|---|---|---|---|
| Blazor | ANO | NE | NE |
| .NET MAUI | NE | ANO | ANO |
| Uno Platform | ANO | ANO | ANO |
| Xamarin | NE | ANO | ANO |

### 2.6.2. Webové aplikace

V rámci použitých multiplatformních vývojových prostředí se zaměřuje na webové aplikace prostředí Blazor a Uno Platform. Rozdílem mezi těmito dvěma nástroji je odlišný způsob provozování aplikace, včetně struktury. Uno Platform se zaměřuje na tvorbu multiplatformní aplikace v rámci nativní aplikace pro zařízení, přičemž umožňuje převést aplikaci do webové podoby, kdežto Blazor tvoří webovou aplikaci, kterou umožňuje poskytovat offline nebo v podobě hybridní aplikace.

Blazor umožňuje tvořit webovou aplikaci v rámci dvou hostitelských modelů Blazor Server a Blazor WebAssembly (WASM). Tyto hostitelské modely se velice liší svým přístupem k datům a kapacitě svého prostředí (viz Obrázek 1). Modely určují, kde budou zpracována data webové aplikace a kde je prezentovat. V rámci serverové aplikace posílá server klientovi informace, které si poté může zobrazit v prohlížeči a tváří se jako plně funkční webová aplikace, i přestože se jedná pouze o přeposlání zobrazení. Touto metodou pracuje pouze Blazor v režimu Blazor Server. Prostředí Uno Platform poskytuje webovou aplikaci pouze v podobě WebAssembly. Rozdílem oproti Blazor WASM je struktura aplikační logiky a způsob tvorby grafiky. Uno Platform

používá vlastní grafické nástroje pro vykreslení aplikace a čerpá z aplikační logiky aplikace, nikoli webové. Aplikaci nezpracovává ve formátu HTML a místo toho používá XAML a CS logiku, jenž rozšiřuje možnosti použití webových aplikací.

### 2.6.3. Desktopové a mobilní aplikace

Multiplatformní vývojová prostředí poskytují jednotný kód pro všechny podporované platformy, ale i tento kód vychází ze specifického odvětví. Logika a chování aplikací vychází z tvorby desktopových aplikací z historických důvodů, kde vývoj mobilních platforem pokračoval na základech desktopových platforem. Příkladem je mobilní operační systém Android, jenž vychází z jádra Linuxu. S touto logikou aplikace pracuje v rámci .NET aplikací jazyk C# a umožňuje zpracování kódu na všech .NET zařízení, ať už po převedení kódu do vhodné podoby nebo jeho přímé zpracování. Obrácený postup však platí pro grafickou stránku aplikací a přístupu ke grafickým prvkům.

Vzhledem k omezené velikosti mobilních obrazovek a omezení vstupu v podobě dotykového displeje nebo tlačítek, je nativní mobilní grafické zobrazení nenahraditelné a v rámci multiplatformního vývoje se od něj odvíjí vývoj grafické stránky aplikace. Na rozdíl od aplikační logiky tady však desktopové zařízení a systémy strádají. Mobilní zobrazení nebylo v rámci operačních systémů Windows, Linux nebo macOS důležité, pokud se nejednalo přímo o mobilní verzi systému. Z toho důvodu nejsou grafické prvky desktopových operačních systémů obohaceny o mobilní podobu. Pro tvorbu grafické části aplikací je potřeba použít minimální počet prvků, které mají plnou podporu v mobilních i desktopových zařízení, nebo zkusit zcela nový způsob. Prostředí Uno Platform řeší tento problém pomocí vykreslovací plochy a prostředí .NET MAUI umožňuje zobrazit webovou aplikaci nativně. Obě tato řešení vykazují v rámci desktopových a mobilních aplikací nejkvalitnější výsledky z pohledu grafiky.

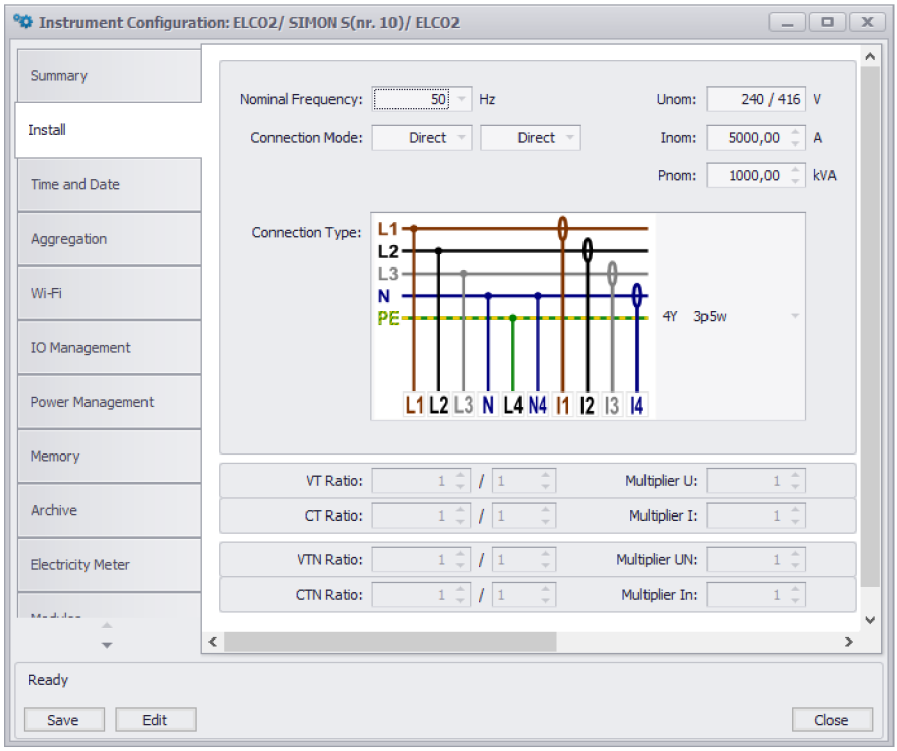
# 3. Vytvořené aplikace

Výstupem této práce jsou tři ukázkové aplikace, včetně knihovny pro práci s daty. Každá aplikace se zaměřuje na své unikátní vlastnosti a podporované platformy. Mezi tyto vlastnosti patří vykreslování uživatelského rozhrání, nativní podoba aplikace v různých prostředích, nebo možnost použít několik systémů pro vyobrazení potřebných informací. V rámci testování byly použity platformy Windows, Linux a Android. Pro platformy od společnosti Apple je zapotřebí kompilovat data na zařízeních s platformou macOS. Struktura a kód těchto aplikací je však použitelná pro všechny podporované platformy a neliší se od výsledných aplikací.

Aplikace používají prostředí, která jsou obsažena v kapitole 2. Tato prostředí jsou vázána s technologií .NET, avšak existují i mnohá další prostředí, která lze použít pro tvorbu multiplatformních aplikací mimo technologii .NET. Mezi tato prostředí se řadí například React Native nebo Flutter.

## 3.1. Předloha aplikací

Výstupní aplikace mají za úkol přiblížit se a napodobit konfigurační nastavení měřících přístrojů v programu Envis, jenž je předlohou pro tyto aplikace. Tento nástroj analyzuje a pracuje s daty z energetických měřících přístrojů a také zajišťuje jejich konfiguraci (viz Obrázek 2). [20]




**Obrázek 2:** Konfigurační nástroj programu ENVIS

### 3.1.1. Uživatelské prostředí

Uživatelské prostředí pro konfiguraci v aplikaci ENVIS obsahuje strukturu v podobě záložek, kde každá záložka obsahuje relevantní informace pro danou kategorii. Obsahem takové záložky je řada kontrolních prvků, jako jsou například výběrové nástroje, vstupní pole pro číselné datové tipy a tlačítka pro jednotlivé operace. Ve spodní části aplikace se poté nachází možnost uložit změněnou konfiguraci včetně možnosti editace.

Toto prostředí umožňuje vhodně zpracovat data v prostředí počítače, ale v rámci multiplatformní tvorby se jedná o rozhraní, které je nutné přepracovat do podoby vhodné pro dotykové obrazovky. Výstupní aplikace nahrazují toto zobrazení za účelem

vytvořit vhodnější podobu konfiguračního nástroje. Většina funkcionalit a nastavení však zůstává nadále v aplikacích, nebo jsou nahrazeny alternativní podobou.

### 3.1.2. Vstupní data

ENVIS používá datový formát cea, jenž je speciálně vyvinut pro tento program. Jedná se o kompresi archivu, která uchovává mnoho různých souborů, mezi které patří i konfigurační soubor. Tento soubor obsahuje veškeré informace o konfiguraci zařízení. Jedná se například o datum a čas uložení konfigurace nebo číselné hodnoty vyjadřující napětí, frekvenci, a to jak v celých číslech, tak desetinných. Mimo tyto formáty obsahuje tento soubor také složitěji uchovávaná data jako je IP adresa, časová zóna nebo číselné rozpětí.

Vzhledem k obrovskému počtu informací pracují výstupní multiplatformní aplikace pouze s omezeným počtem dat. Tato data reprezentují originalitu jednotlivých prvků a datových typů. Reprezentovaná data zároveň zamezují mnohonásobnému zpracování stejné informace. Předání těchto informací v podobě cea souboru je v tomto případě nevhodné, vzhledem k rozsáhlé kapacitě nepoužitých dat. Z tohoto důvodu jsou data z cea souboru přepsána do podoby souboru json, který spolupracuje s technologií .NET a obsahuje jednodušší a čitelnější strukturu, která umožňuje uložená data použít pro další účely.

## 3.2. Knihovna pro práci s daty

Výstupní aplikace sdílí jednu vlastnost a tou je použití technologie .NET. Jelikož tyto aplikace zpracovávají jednotný systém, který se liší pouze uživatelským rozhraním a počtem platforem, je vhodné oddělit identický kód. V případě ukázkových aplikací se jedná o externí .NET 6 knihovnu tříd. Knihovna je od aplikací oddělena a lze v ní pracovat nezávisle na aplikaci nebo prostředí, ve kterém je použita. Při sestavování aplikací se tato knihovna přidá do balíčku v podobě dll souboru. V případě ukázkových aplikací knihovna obstarává datovou a business vrstvu aplikací neboli práci s daty a jejich načítání.

### 3.2.1. Přístup k datům

Knihovna si uchovává veškeré informace o konfiguraci v třídě `JsonData`. Třída obsahuje veškeré proměnné, které korespondují s daty souboru json. Proměnné třídy se uchovávají v podobě datových formátů integer, string, boolean a double v závislosti na jejich kontextu. Všechny proměnné jsou volně přístupné vzhledem k použití přístupu public, avšak neobsahují žádnou přímou referenci v rámci prezentační vrstvy v aplikacích. I přestože se může zdát přímé připojení k datům z prezenční vrstvy jako vhodný způsob předání informací v proměnné, způsobuje toto řešení vážný problém v rámci generování výsledného souboru json. Aplikace vyžadují použití statických proměnných pro jakýkoliv zásah do externího prostředí, což znemožňuje generování json podoby souboru. Vhodným způsobem, jak předat třídu s daty do aplikace, je použití instance třídy v rámci aplikace, kde aplikace s touto instancí pracuje. Toto řešení však nelze použít pro některá prostředí vzhledem ke struktuře, kde je kód izolován a nelze vytvořit jednotnou instanci pro celou aplikaci. Aby se tento problém obešel, vytvoří se instance této třídy přímo v prostředí knihovny v třídě pro načítání, která poté slouží jako prostředník pro přístup k této třídě. Kód této třídy se nachází v příloze, viz Zdrojový kód 1.

### 3.2.2. Načítání souboru json

Načítání a sestavení json podoby dat obstarává druhá třída knihovny zvaná `LoadData`, která obstarává generování a předávání souboru do aplikace. I přestože tato třída zastupuje datovou vrstvu všech aplikací, nemá kompletní kontrolu nad vstupním a

výstupním souborem a pouze generuje a čte string. Výstupní aplikace pracují se stejnou technologií o stejném základu, ale vzhledem k mnoha limitací jednotlivých prostředí nedokážou poskytnout plnou kontrolu nad souborovým systémem zařízení. Toto omezení znemožňuje použít samostatnou datovou vrstvu pro všechna prostředí, natož pro mnoho různých cílových platforem. Z toho důvodu slouží tato třída pouze jako prostředník pro přenos instance třídy `JsonData` společně s funkcemi zajišťujícími převod formátu json do aplikace.

Pro převedení json struktury v podobě datového typu `string` používá tato třída .NET API referenci zvanou `System.Text.Json`. Pomocí tohoto API je možné formátovat objekt do json podoby a také zpětně přeformátovat do objektu. V případě třídy `LoadData` je toto API použito pro dvě funkce. První funkcí je funkce `CreateJson`, která je volána aplikací v případě, že uživatel zvolí možnost uložit si nastavené hodnoty. Tato funkce vygeneruje json podobu instance třídy `JsonData` do datového typu string, který nadále poskytuje aplikaci. Aplikace převzatý string uloží dle svých možností jako samostatně vygenerovaný soubor. Druhou funkcí je `UploadJson`, která zajišťuje zpětné přeformátování. Tato funkce obdrží vstupní soubor json v podobě proměnné string, který poté převede do instance třídy JsonData, odkud čerpá aplikace data. [21]

Třída `LoadData` obsahuje pouze statické proměnné a funkce, aby bylo možné k nim přistupovat bez nutnosti tvorby dalších instancí. Krom funkcí pro úpravu json podoby dat obsahuje tato třída také proměnnou typu boolean, jenž označuje, zda jsou data načtena. Proměnná boolean je v aplikacích použita jako kontrola, zda se může veškerý obsah aplikace zobrazit a editovat. Kód této třídy je obsažen v příloze, viz Zdrojový kód 2.

## 3.3. Aplikace Blazor

Blazor aplikace zastupuje multiplatformní tvorbu aplikací v podobě webového prostředí. Tvoří webovou aplikaci v podobě WebAssembly, která se spouští ve webovém prohlížeči. Aby byla aplikace kompatibilní s co největším množstvím zařízení a zároveň se zmíněnou knihovnou, používá podobu aplikace typu .NET 6. Webovou aplikaci lze hostovat na zařízeních s operačním systémem Windows, macOS a Linux.

Pro zabezpečení používá aplikace konfiguraci pro HTTPS protokol. Konfigurace je generována prostředím Visual Studio, kde lze také změnit cílovou technologii v případě problému s kompatibilitou. V případě aplikace Blazor lze změnit technologie .NET 6 na technologii .NET Core 3.1. Nástroj Blazor také umožňuje převést WebAssembly aplikaci do podoby Progresivní Webové Aplikace (PWA). Tato podoba umožňuje nainstalovat balíček aplikace jako desktopovou aplikaci, která lze spustit bez nutnosti prohlížeče. Výstupní aplikace tuto podobu neobsahuje, ale umožňuje.

### 3.3.1. Struktura aplikace

Aplikace udržuje základní Blazor strukturu a odděluje kód knihovny. Izolace kódu zde omezuje rizika v podobě chyb, které se mohou přenášet přes celou aplikaci. Stejně jako kód jsou izolovány i soubory zprostředkující webové zobrazení. Aplikace používá základní webovou strukturu html odděleně od aplikace, vzhledem k tomu, že veškerý obsah aplikace je do této struktury vyobrazen, ale není jeho součástí. Z toho důvodu nejde obsah aplikace staticky vyobrazit a je potřeba aplikaci sestavit.

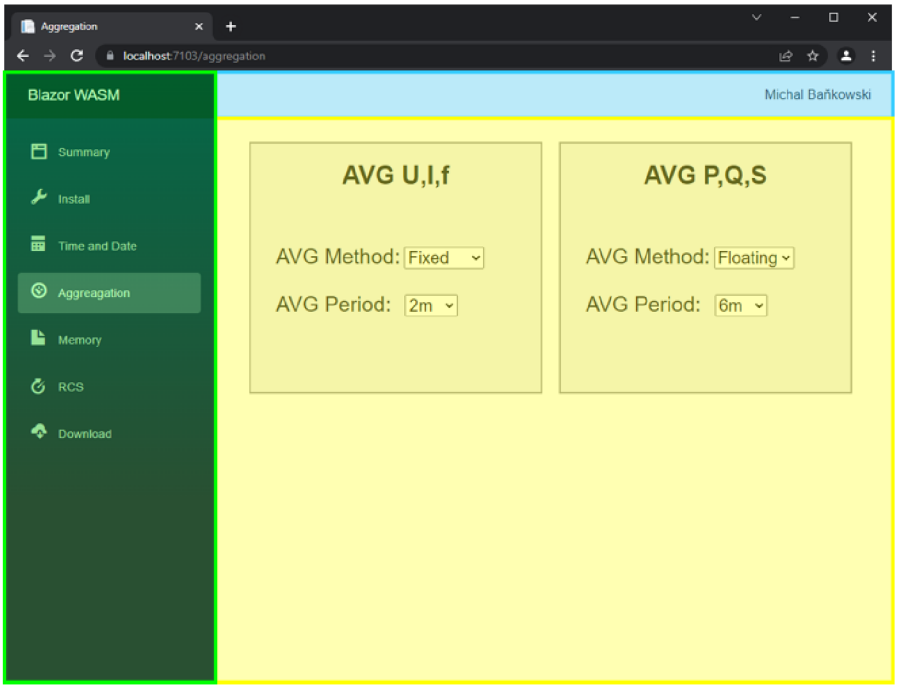
Externí .NET 6 knihovna je v aplikaci odkázána jako existující projekt mimo aplikaci. Veškeré změny, které se provedou v této knihovně, budou platné pro vytvořené aplikace nehledě na právě zvolené prostředí. Knihovna je k aplikaci připojena pomocí takzvaných závislostí. Toto spojení zajišťuje, že aplikace vždy tuto knihovnu vyhledá a bude schopna z ní čerpat a sestavit ji do podoby dll souboru, který se stane součástí balíčku.

Kořenem celé aplikace je soubor `Program.cs`, který zajišťuje základní rozložení aplikace. Mimo rozložení také přidává služby pro aplikaci, v případě výstupní aplikace se jedná o službu `HttpClient`, jenž je součástí webového API technologie ASP.NET Core. Obsah veškerého grafického rozhraní a chování aplikace se nachází ve složkách

Shared, kde se nachází rozvržení aplikace a sdílené komponenty, a ve složce Pages, která obsahuje všechny komponenty neboli jednotlivé stránky aplikace. Tyto soubory jsou formátu razor a obsahují kombinaci html, css a C# kódu.

### 3.3.2. Uživatelské rozhrání

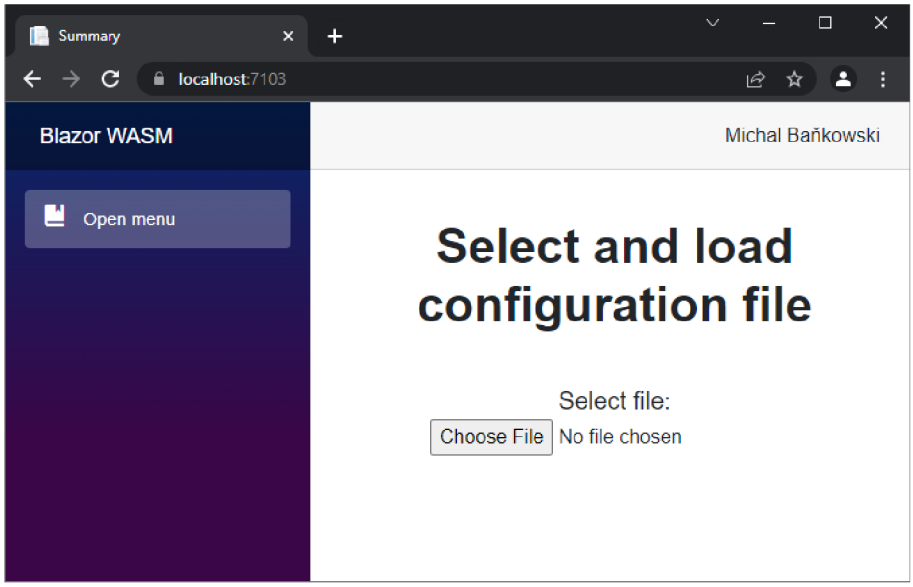
Základní rozložení aplikace je obsaženo ve sdíleném souboru MainLayout. Toto rozložení rozděluje stránky na tři sekce. První sekcí je navigační menu, tato sekce má vlastní komponentu, jenž je detailně popsána níže. Druhou sekcí je jednoduché záhlaví, které je použito pro označení autora aplikace a třetí sekce slouží pro zobrazení stránek. Toto rozložení je aplikováno na všechny stránky aplikace a všechna zařízení nehledě na rozlišení a poměru stran. Pro responzivní zobrazení jsou tyto sekce řízeny kaskádovými styly z prostředí Blazor. Rozložení aplikace je vyobrazeno na následujícím obrázku (viz Obrázek 3).




**Obrázek 3:** Graficky znázorněné rozložení aplikace Blazor

Navigační menu slouží pro orientaci a pohyb v rámci aplikace. Rozpoložení jednotlivých stránek napodobuje předlohu, kde každá záložka obsahovala vlastní část konfigurace. Na rozdíl od běžných webových aplikací je obsah měněn dynamicky. Pro omezení přístupu je v aplikaci Blazor použito zakrytí komponent v menu. Prostředí Blazor umožňuje vykreslovat html elementy v závislosti na podmínkách, jenž se dají

naprogramovat v rámci jednotlivých komponent, v tomto případě komponenty NavMenu. Aby uživatel mohl s aplikací pracovat, je vyzván k načtení konfiguračního souboru json. V případě, že uživatel soubor načte, odemkne se mu menu umožňující zobrazení všech komponent (viz Obrázek 4). Pro kontrolu načtení je použita proměnná boolean v externí knihovně, kde se provádí načítání vstupního textového řetězce ze souboru do instance objektu třídy. Zobrazení kompletního navigačního menu vyžaduje komponentu obnovit a načíst v druhé podobě. Jelikož Blazor neumožňuje dynamicky kontrolovat stav proměnné a obnovit vzhled menu, je uživatel nucen kliknout na menu, jenž spustí funkci, která upozorní aplikaci na změnu stavu.



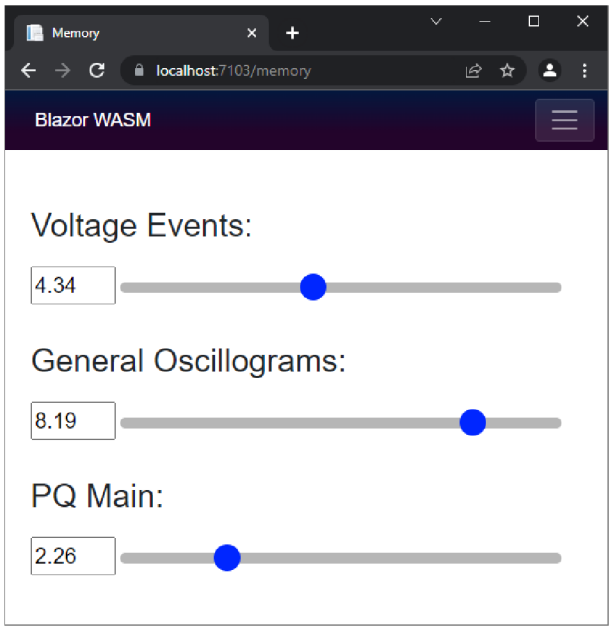

**Obrázek 4:** Skryté menu aplikace Blazor

Výstupní aplikace Blazor cílí na všechna zařízení s podporou technologie .NET a zajišťuje vhodné uživatelské prostředí nejen pro vstupní prvky jako klávesnice a myš, ale také pro dotykové displeje. Veškeré prvky aplikace jsou zvoleny tak, aby bylo jejich ovládání možné provádět na menších mobilních zařízení s dobrou čitelností. Všechny tyto prvky jsou tvořeny pomocí html elementů s podporou HTML 5, včetně kaskádových stylů, jenž jsou izolovány pro každou komponentu zvlášť.

Číselná data jsou v aplikaci řešena pomocí elementu input. V rámci celých čísel je zvolen typ `number`, kde je nastavena maximální a minimální hodnota čísla. Tento element je ideálním řešení pro veškeré vstupní periferie, kde je dotykový vstup řešen virtuální číselnou klávesnicí a v případě jednoduché změny umožňuje použití šipek, jež jsou umístěny vedle hodnoty. Element input nepovoluje vložení jiné než číselné

hodnoty, čímž ošetřuje vstupní hodnotu a zároveň ji i zaokrouhluje v případě snahy o vložení desetinného čísla. Některé hodnoty v konfiguračním souboru však desetinná čísla obsahují a v rámci aplikace tato čísla lze nastavit dvěma způsoby.

Prvním způsobem je použití již zmíněného inputu v podobě číselného inputu, kde lze nastavit počet desetinných míst pro vstup pomocí atributu `step`, jenž určuje minimální rozdíl mezi čísly. Druhou možností je použití inputu typu `slider`. Toto řešení koresponduje s předlohou aplikace, kde byl použit posuvník zároveň s číselným vstupem. Prostředí Blazor dokáže takové chování také zajistit a je součástí aplikace (viz Obrázek 5). Oba číselné vstupy jsou na sobě závislé a při změně hodnoty v jednom vstupu se změní i hodnota v druhém.

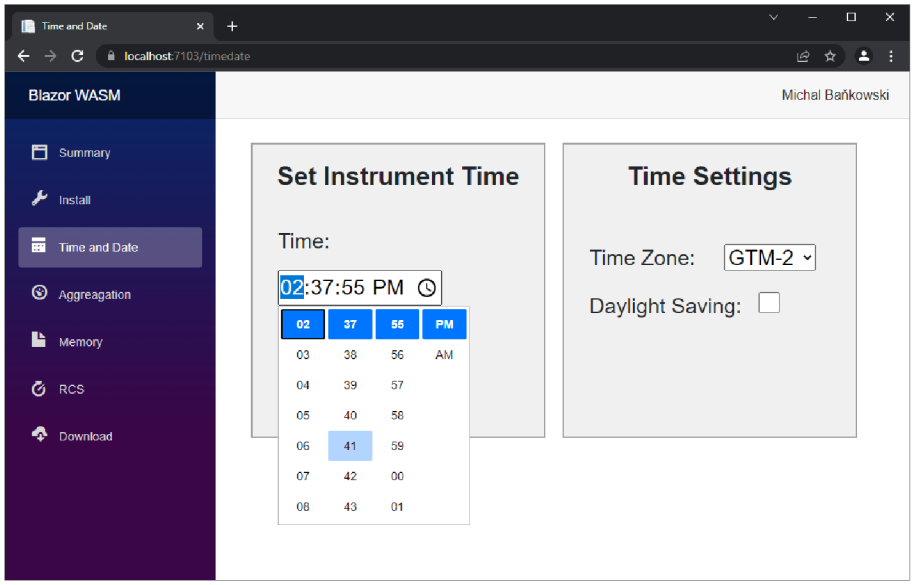



**Obrázek 5:** Číselné vstupy v aplikaci Blazor

Písemné vstupní hodnoty, které jsou uloženy v podobě proměnných datového typu string, jsou v aplikaci obsaženy v podobě výběru. Možnost vložení libovolného textu je pro konfigurační soubor risk v případě zadání neplatné hodnoty. Místo vkládání textu umožňuje aplikace vybrat si daný text nebo danou hodnotu z rozbalovací nabídky. Tato nabídka je řešena pomocí elementu select, kde jednotlivé možnosti zastupují textový řetězec, který bude uložen ve výsledné konfiguraci.

Poslední kategorií dat jsou data, jenž se nedají snadno vypsat nebo vybrat ze seznamu. Ve výstupní aplikaci se jedná především o určení času a datumu. Pro uložení datumu a času jsou použity elementy input typu time a date. Tyto elementy se přizpůsobují

prohlížeči a nabízí uživateli přehlednou nabídku v podobě hodin nebo kalendáře. Velkou výhodou je přizpůsobivost tohoto elementu v závislosti na operačním systému a prohlížeči. Datum i čas se zobrazují ve formátu, jenž odpovídá zařízení, ale výsledná hodnota se ukládá nezávisle. Čas lze obdobně jako číselné vstupy nastavit pomocí atributu step na detailnější jednotky, v případě aplikace se jedná o vteřiny (viz Obrázek 6).



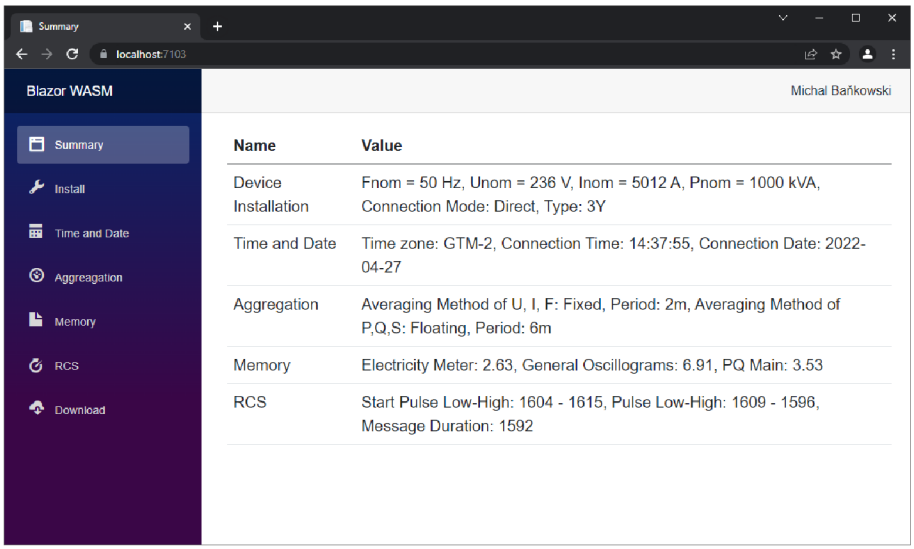

**Obrázek 6:** Nastavení času v aplikaci Blazor

### 3.3.3. Zobrazení a úprava dat

Veškeré hodnoty elementů jsou převzaty z externí knihovny. Aplikace umožňuje nejen tato data zobrazit, ale také změnit pomocí vázání dat. Toto vázání zajišťuje dynamické prezentování dat přímo v prostředí html kódu. Struktura Blazor je zaměřena na izolaci kódu, kde jsou data i html elementy součástí jednoho razor souboru, ale v případě výstupní aplikace jsou data čerpána z externí knihovny a veškerá další funkcionalita je řešena přímo v souborech razor, jak vývojáři původně zamýšleli.

Pro zobrazení těchto dat obsahuje html část razor souborů vsuvky v podobě zavináčů, kde následuje jméno dané proměnné včetně cesty k ní. Symbol zavináče je unikátní pro Blazor aplikace, jelikož odlišuje html kód od kódu C#. Díky této syntaxi lze vytvořit podmínky v rámci HTML dokumentu a je možné zobrazit veškeré systémové informace, ke kterým má technologie .NET přístup v rámci zařízení. Příklad této syntaxe je vyobrazen v příloze, viz Zdrojový kód 3, kde je zobrazena část kódu pro

komponentu Summary. Tato komponenta po zadání souboru vyobrazuje veškerá data z knihovny v podobě tabulky (viz Obrázek 7)




**Obrázek 7:** Výpis dat z externí knihovny v aplikaci Blazor

Úprava dat probíhá v aplikaci podobně jako jejich zobrazení. Veškeré vstupní prvky v html kódu obsahují atribut value. Tento atribut slouží k zobrazení základní hodnoty, která poté může být změněna. V rámci výstupní aplikace je tento atribut nahrazen syntaxí `@bind`. Tato syntaxe přiřazuje atributu value hodnotu proměnné, ke které je vázána. Proměnná musí mít vhodný datový typ, aby bylo možné ji vázat k prvku a musí splňovat nastavené náležitosti, jako jsou například maximální nebo minimální možná hodnota. Při změně hodnoty prvku ze strany uživatele se změní i její hodnota v instanci třídy externí knihovny. Data se mění při dokončení výběru nebo dokončení vypisování hodnoty. V případě, že je data potřeba měnit v reálném čase, lze tuto změnu specifikovat přímo v elementu. Ukázkou takové funkce je posuvník, jehož hodnota se mění identicky s číselným vstupem (viz Obrázek 5).

### 3.3.4. Načítání a ukládání souboru

Při spuštění aplikace je uživatel vyzván, aby vložil konfigurační soubor do aplikace. Tento soubor lze vložit pomocí html elementu InputFile, který otevírá systémový soubor zařízení, kde poté může uživatel vybrat vstupní soubor. Při výběru je použit základní průzkumník souborů dle operačního systému. Po vložení souboru si tento element uchová informace jako je název a kapacita souboru včetně jeho obsahu. Vložení souboru spustí úlohu, jenž data ze souboru převede do textového řetězce a

následně se uživateli zobrazí tlačítko Upload, které posílá zpracovaný řetězec do knihovny a obnoví stránku s novými parametry.

Načítání souboru probíhá pomocí takzvaného asynchronního programování, jenž je jednou z mnoha součástí technologie .NET. Pro načtení souboru je vytvořena asynchronní úloha `LoadFile`, jenž je součástí razor komponenty Summary. Vstupním parametrem úlohy je argument předávající informace o souboru. Pro ukládání dat ze souboru je vygenerován MemoryStream, který poté bude data ze souboru uchovávat.

Asynchronní programování odlučuje aktuálně pracující kód s kódem nahrávání souboru, kde kopírování dat souboru vyčkává díky syntaxi await na přísun dat. Pro tvorbu textového řetězce je MemoryStream, jenž obsahuje data ze souboru, převeden na pole pomocí metody `ToArray`. Následně je toto pole převedeno na textový řetězec pomocí metody `GetString`, který je uložen do proměnné v rámci komponenty razor. Při stisknutí tlačítka upload je zavolána funkce, jež posílá tuto proměnou do externí knihovny, kde je její obsah převeden do třídy proměnných v rámci knihovny, viz Zdrojový kód 3.

Ukládání souboru je v aplikaci řešeno značně složitěji. Prostředí Blazor neumožňuje ukládat soubor, jenž není součástí sestavené aplikace, což v případě vygenerovaného souboru nelze. Aplikace však výsledný soubor uložit dokáže, a to díky chytrému použití kódu JavaScript mimo technologii .NET a prostředí Blazor, ze strany základní webové aplikace.

Javascript kód je v aplikaci uložen samostatně ve složce wwwroot, jenž zajišťuje základní náležitosti webové stránky. Tento skript je přidán do základní stránky index.html a stává se tak součástí aplikace mimo strukturu prostředí Blazor. Skript obsahuje funkci `BlazorDownloadFile`, která obstarává nikoli uložení souboru, ale odkazování na něj. V případě webového prostředí je v rámci html kódu jakýkoliv odkazovaný soubor možné přímo stáhnout. Tento způsob stahování souborů je běžně používán ve webových aplikacích i mimo výstupní aplikaci.

Funkce obsahuje tři vstupní parametry. Prvním parametrem je název souboru, který bude vygenerovaný soubor obsahovat. Název souboru je zapsán i s příponou, ke které koresponduje druhý parametr, typ souboru. Poslední parametr obsahuje řetězec, jenž slouží pro vložení obsahu do vygenerovaného souboru. Po načtení vstupních

proměnných funkce vygeneruje soubor obsahující již zmíněné parametry. Pro soubor je vytvořen URL odkaz pomocí metody `createObjectURL`. Funkce poté vygeneruje odkazovací html element `a`, jenž vloží do stránky index.html. Tomuto elementu se přidělí atributy, jenž k němu připojí vygenerovanou URL společně se zacílením sama na sebe. Poté je vygenerováno kliknutí, jenž stahování souboru zahájí a následně je URL odkaz vypuštěn.

Pro zavolání funkce komunikuje webová aplikace ze strany prostředí Blazor přímo s vnitřní webovou strukturou. Tato komunikace je zajištěna pomocí IJSRuntime, jenž je součástí .NET API. Výstupní aplikace obsahuje toto API v asynchronní úloze pro stažení souboru, kde v asynchronní podobě volá funkci pro stažení souboru se vstupními parametry, které určují podobu souboru json.

## 3.4. Aplikace .NET MAUI

.NET MAUI aplikace cílí na nativní prostředí zařízení, jenž je dáno jeho operačním systémem. Pro přetvoření aplikace do vhodné podoby používá aplikace balíčky SDK, jenž jsou součástí šesté verze technologie .NET. Tyto balíčky aplikaci kompilují do podoby, jenž lze zprovoznit nebo nainstalovat v operačním systému daného prostředí.

Aplikace je cílena na platformy Windows, Android, iOS a MacCatalyst. Prostředí operačních systémů od značky Apple jsou omezena a lze pro ně aplikaci kompilovat pouze na jejich vlastních zařízeních. Z toho důvodu je aplikace testována a poskytnuta pro systémy Windows a Android. Prostředí .NET MAUI je stále ve vývoji a neustále obdržuje nové aktualizace. Z toho důvodu může být řešení aplikace, včetně její struktury, zastaralé a může vykazovat chyby.

### 3.4.1. Použití technologie Blazor

Prostředí .NET MAUI poskytuje tvorbu aplikace dvěma způsoby. Prvním způsobem je použití XAML a CS struktury, jenž umožňuje vložit omezené množství prvků, které jsou kompatibilní se všemi cílovými platformami. Druhým způsobem je použití technologie Blazor v podobě WebView. Tento způsob zobrazení Blazor aplikace lze použít uvnitř prostředí .NET MAUI a je obsažen ve výsledné aplikaci. Použití XAML a CS obsahuje minimální počet prvků a vykazuje anomálie v desktopové podobě Windows. Struktura Blazor v rámci aplikace tyto anomálie obsahuje v minimálním množství a umožňuje tvořit jednotné grafické rozhrání, jenž umožňuje více grafických úprav a podporuje detailnější tvorbu aplikace. Výstupní aplikace používá Blazor strukturu následujícím způsobem.

Nastavení aplikace je obsaženo v souboru MainPage.xaml, který obsahuje základní vlastnosti .NET MAUI aplikace, včetně prvku ContentPage. Tento prvek označuje tělo aplikace a obsahuje vlastnosti jako je jméno aplikace a zdroj schématu, který aplikace používá. V tomto prvku je vnořena komponenta BlazorWebView, jenž zajišťuje zobrazení struktury Blazor v aplikaci. BlazorWebView obsahuje informaci oznamující cestu k základnímu HTML souboru index.html, včetně hlavního razor souboru aplikace.

### 3.4.2. Struktura aplikace

Aplikace je rozdělena do tří individuálních částí. První částí jsou soubory pro tvorbu v prostředí .NET MAUI. Tyto soubory obstarávají základní potřeby pro spuštění aplikace na cílených platformách. Druhou částí je struktura Blazor aplikace, která je téměř identická s již zmíněnou výstupní aplikací Blazor, obsaženou v kapitole 2.1.2. Třetí část tvoří externí .NET 6 knihovna, jenž je v aplikaci odkazována.

.NET MAUI struktura aplikace je obsažena v aplikaci v následující podobě. Soubory, jako je hlavní stránka aplikace a třída pro sestavení WebView, jsou obsaženy přímo v projektu jako základ aplikace. Nejdůležitější součástí této struktury jsou projekty cílené na každou platformu zvlášť. Projekty poskytují informace o aplikaci v podobě cílené platformy. Mezi tyto informace patří název aplikace, název instalačního balíčku a schémata, která platforma používá.

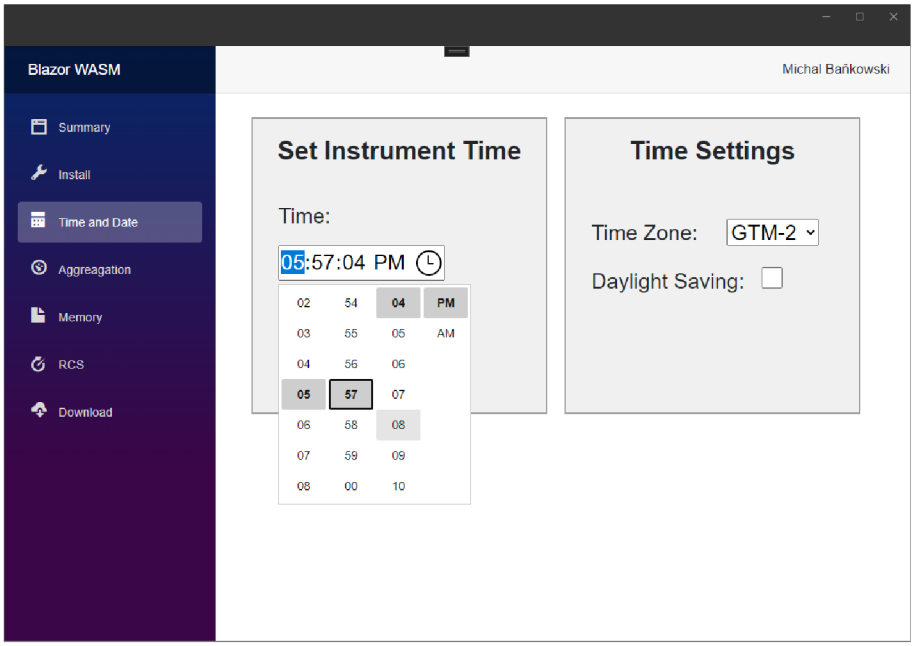
Blazor struktura je v prostředí aplikace .NET MAUI vnořena ve vlastní části aplikace. Komponenty jsou uloženy ve složkách Shared a Pages v závislosti na obsahu. Soubory webové podoby aplikace jsou uloženy ve složce wwwroot. Tato složka obsahuje krom základního souboru index.html také obrázky, jenž aplikace zobrazuje. Pro multimediální soubory jako jsou obrázky, fonty nebo zvukové stopy je určena v rámci .NET MAUI aplikací vlastní složka, která však ze strany Blazor aplikace není viditelná, a proto nemá ve výsledné aplikaci využití a není součástí výsledné aplikace.

Knihovna je v aplikaci přidána pomocí reference. Obsah knihovny je v aplikaci zobrazen, avšak není její fyzickou součástí a je umístěn samostatně mimo aplikace .NET MAUI. Pro tuto externí knihovnu je vytvořena závislost, která zajišťuje sestavení aplikace společně s knihovnou v podobě souboru dll.

### 3.4.3. Prostředí Windows

Uživatelské rozhrání je identické s rozhráním webové aplikace Blazor. Rozdílem oproti aplikaci Blazor je chování a zobrazení těchto prvků. Webová aplikace je řízena prohlížečem, který určuje vzhled aplikace a obsahuje podporu pro veškeré platformy. V rámci nativního zobrazení závisí výsledná aplikace na balíčcích technologie .NET 6, které přizpůsobují webové prvky aplikace do nativní podoby.

V rámci .NET MAUI aplikace pro Windows je největší viditelnou změnou prostředí okna Windows. Okno aplikace lze velikostně upravit, včetně poměru stran. Úprava velikosti stran koresponduje s responzivním zobrazením a aplikují se úpravy v rámci kaskádových stylů. Horní lišta okna umožňuje základní chování, jako je minimalizovat, maximalizovat nebo zavřít aplikaci. Prostředí Windows aplikace také reaguje na klávesové zkratky.

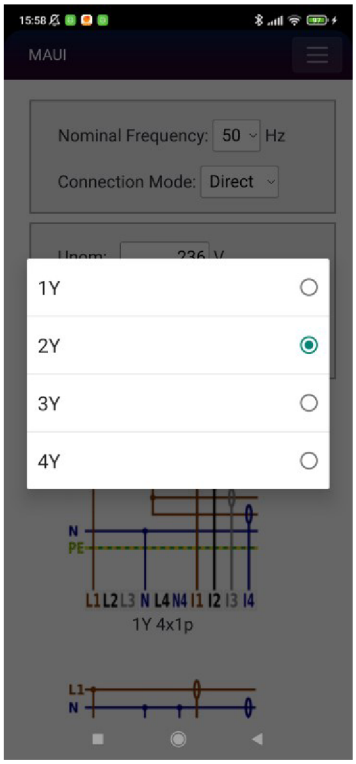



**Obrázek 8:** Nastavení času v aplikaci .NET MAUI

V rámci ovládání prvků aplikace simuluje chování webového prostředí. Výběr ze vstupních prvků je s webovou aplikací identický, liší se pouze barva pozadí, kde je místo modré použita šedá barva (viz Obrázek 6 a Obrázek 8). Výstupní aplikace však obsahuje několik chyb, jež budou s vysokou pravděpodobností v budoucnu opraveny. Nefunkčním prvkem je posuvník, jenž je obsažen v komponentě Memory, viz Obrázek 5. Tento prvek lze posouvat pouze v řádu jednotek i přestože obsahuje atribut s nastavením pohybu v řádu setin. Problém s tímto prvkem je v aplikaci nalezen pouze pro platformu Windows. Další nedostatek, který aplikace vlastní, je omezení notifikací. Pro vkládání souboru je použit systémový průzkumník, avšak ukládání dat probíhá bez jakékoliv notifikace. Stahování dat řeší webové prohlížeče několika způsoby, kde se odkaz na stažený soubor nachází na horní nebo spodní liště.

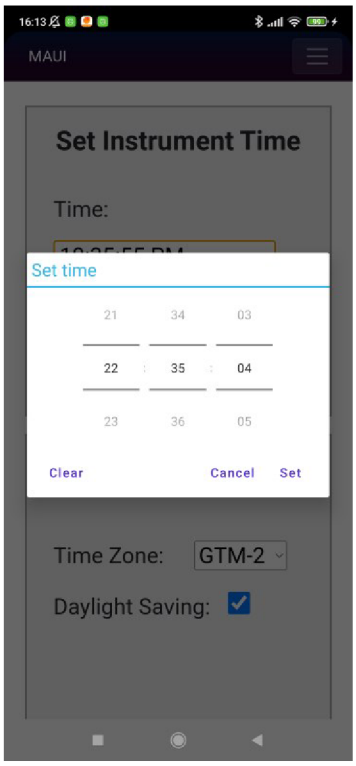
### 3.4.4. Prostředí Android

Aplikace je v prostředí operačního systému Android znatelně rozdílná oproti podobě v operačním systému Windows. Není obsažena v okně, neobsahuje rámeček ani lištu a nevykazuje negativní náznaky multiplatformní tvorby. Při vkládání souboru je využit systémový průzkumník a vkládání souboru probíhá bez problémů. Grafické prvky, které jsou součástí technologie Bootstrap, se uvnitř aplikace přizpůsobují systému a upraví grafické detaily aplikace. Mezi tyto změny patří například zaostření tlačítek nebo rozsvícení okraje prvku při dotyku.




**Obrázek 9:** Výběr hodnoty v mobilním prostředí aplikace .NET MAUI

Jelikož operační systém Android slouží především pro dotykové displeje, jsou prvky aplikace pro tato zařízení přizpůsobeny. Jakýkoliv výběr z elementu select je řešen vyskakovacím oknem uvnitř aplikace (viz Obrázek 9). Vstupní číselné pole neobsahuje šipky, ale při dotyku lze hodnotu editovat pomocí virtuální číselné klávesnice. Pro výběr času a datumu jsou použity systémové nástroje, u kterých je grafické omezení nativních prvků nejvíce znatelné (viz Obrázek 10).




**Obrázek 10:** Nastavení hodin v mobilním prostředí aplikace .NET MAUI

Bohužel však prostředí .NET MAUI není vytvořeno za účelem spravovat data a aplikace má v prostředí Android fatální chybu. I přestože soubor lze načíst a upravit, není možné jej z aplikace stáhnout. Toto prostředí neumožňuje ukládání dat a ani s použitím funkce v kódu JavaScript aplikace nedokáže, v aktuální verzi pro operační systém Android, vygenerovaný soubor uložit.

## 3.5. Aplikace Uno Platform

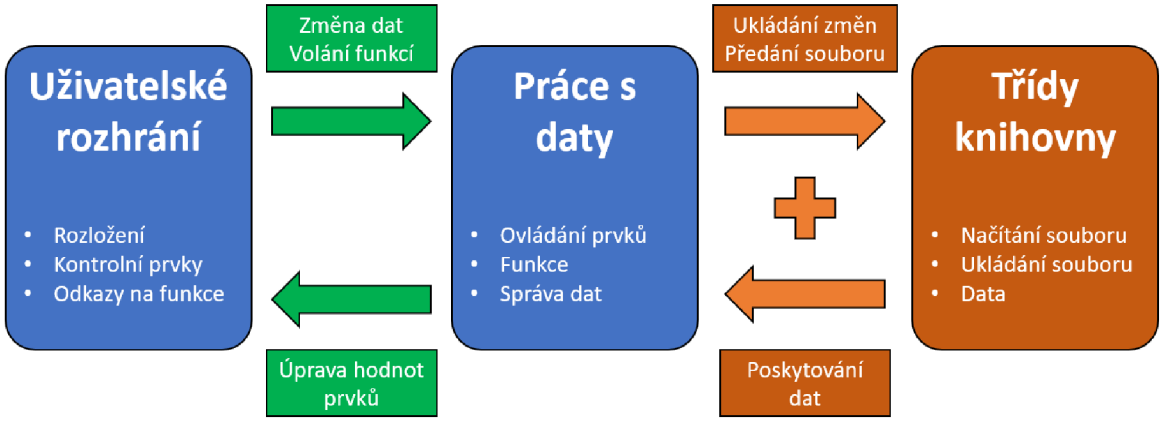
Aplikace Uno Platform, dále pouze Uno, reprezentuje multiplatformní tvorbu aplikací z podoby nástroje třetí strany. Výstupní aplikace používá a testuje mnoho výhod, které toto prostředí vyzdvihuje. Nejzásadnější vlastností je počet cílených platforem a jednotné uživatelské prostředí pro všechny tyto platformy. Schopnosti tohoto prostředí jsou součástí výsledné aplikace, která napodobuje předlohu a zároveň ukazuje, jak se tyto výhody projevují v tvorbě.

Uno je vytvořeno v podobě .NET 6 aplikace a cílí na platformy Windows, Linux, Android, iOS, MacCatalyst, MacOS a webové prostředí. Aplikace používá dvě metody, jak zacílit tento velký počet různých prostředí. První metodou je poskytování aplikace v podobě již funkčního nástroje, ke kterému je přidán sdílený obsah. Druhá metoda používá vykreslování aplikace pomocí knihovny SKIA, kde obsah aplikace generuje. Oba tyto způsoby jsou v rámci výstupní aplikace použity.

### 3.5.1. Nahrazení knihovny

Aplikace Uno nepoužívá externí .NET 6 knihovnu. Struktura aplikace neumožňuje přidat existující projekt, jenž je obsažen mimo hlavní složku aplikace. Toto omezení znemožňuje pracovat s veškerými externími zdroji, které nejsou poskytnuty jinak než v podobě balíčků cílené pro každou platformu zvlášť.

I přestože prostředí Uno Platform neumožňuje přidání knihovny, jsou data z knihovny v aplikaci obsažena. Externí knihovna obsahuje dvě třídy, které komunikují pouze navzájem bez použití dalších náležitostí. Identické kopie těchto dvou tříd jsou obsaženy v aplikaci Uno. Použití těchto tříd splňuje použití jednotného kódu, ale za cenu náročnosti aplikace. V aplikaci je kód rozdělen do podoby uživatelského rozhraní a zdrojového kódu, jenž uživatelské rozhraní používá. Uno obsahuje navíc třetí skupinu, třídy z knihovny. Uživatelské rozhrání nelze s třídami svázat a vzniká problém, kdy je nutné data z knihoven do aplikace přesouvat manuálně (viz Obrázek 11).




**Obrázek 11:** Rozdělení kódu aplikace Uno

Jak je na předchozím obrázku znázorněno, přidané třídy aplikaci nezjednodušují, jak je původně zamýšleno, ale naopak, jsou v aplikaci navíc. Změna libovolného prvku je zaznamenána ve funkci, která novou hodnotu vloží do knihovny. Při načítání je použit opačný postup, kde při načtení libovolné stránky aplikace je spuštěna funkce, která do prvků v uživatelském rozhraní načte hodnoty z knihovny. V obou případech se jedná o neefektivní způsob práce s daty, který však splňuje požadavek jednotného kódu externí knihovny pro všechny aplikace.

### 3.5.2. Struktura aplikace

Aplikace je rozdělena do dvou částí. V první části jsou obsaženy všechny cílené platformy a jejich vlastní projekt. Tato část je pro tvorbu aplikace nepodstatná a je vygenerována prostředím Uno Platform. Veškerý kód aplikace, včetně souborů, je obsažen v druhé části, jenž je pro všechny projekty sdílena. Sdílený kód je součástí projektu Shared, který není spustitelný a ani neobsahuje balíčky. Oddělení těchto dvou částí znatelně zjednodušuje tvorbu aplikace a omezuje tvorbu chyb.

Projekty obsaženy v první části aplikace nejsou určeny pouze pro jednotlivé platformy, ale používají také jiné nástroje pro vytvoření aplikace. Základním cílem projektů je jediné, sestavit aplikaci ze sdílených souborů. Pro sestavení aplikace obsahuje každý projekt vlastní sadu balíčků, konfigurační soubory a multimediální soubory v podobě fontů. V rámci webové aplikace obsahuje webový projekt navíc kaskádové styly společně se soubory JavaScript. Každá platforma se svým projektem silně odlišuje v závislosti na cílové platformě a převzaté struktuře.

Druhá část obsahuje sdílený projekt, jenž obsahuje veškerý obsah aplikace. Přidané soubory, jež aplikace používá, jsou uloženy ve složce Assets. Tato složka je určena pro multimediální soubory a v rámci výstupní aplikace Uno obsahuje obrázky použité v aplikaci. Třídy jsou obsaženy ve složce Data, která tímto způsobem odděluje třídy z knihovny od zbytku souborů. Hlavní a nejdůležitější součástí sdíleného projektu jsou soubory XAML a CS. Soubory XAML obsahují grafickou reprezentaci celé aplikace společně se soubory CS, které obstarávají aplikační logiku. V rámci výstupní aplikace Uno obsahuje sdílený projekt soubory XAML a CS pro každou stránku aplikace samostatně.

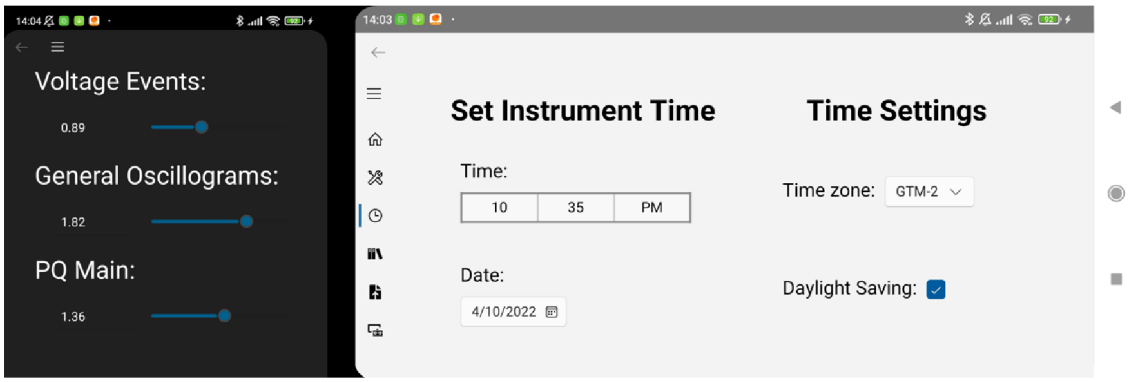
### 3.5.3. Uživatelské rozhrání

Grafická podoba aplikace je vytvořena pomocí kontrolních prvků značkovacího jazyku XAML. Veškeré vytvořené prvky vypadají identicky pro všechny cílové platformy a navzájem se neliší. Počet možných prvků je však omezen z důvodu použití vícero platforem. Rozvržení jednotlivých stránek je řešeno pomocí dvou kontrolních prvků, Grid a StackPanel. Grid je použit v případě, pokud jsou jednotlivé prvky aplikace rozděleny do skupinek, kde každá skupina prvků má určenou pozici na stránce. Srovnání prvků poté obstarává prvek StackPanel. StackPanel seřazuje prvky do řady, a to vodorovně nebo svisle v závislosti na jeho nastavení. Tyto panely umožňují vnoření, jenž je použito v případě, že stránka obsahuje mnoho na sebe navazujících prvků. Prvek Grid lze také použít pro tvorbu tabulek (viz Obrázek 12).


**Obrázek 12:** Tvorba tabulky pomocí prvku Grid v aplikaci Uno

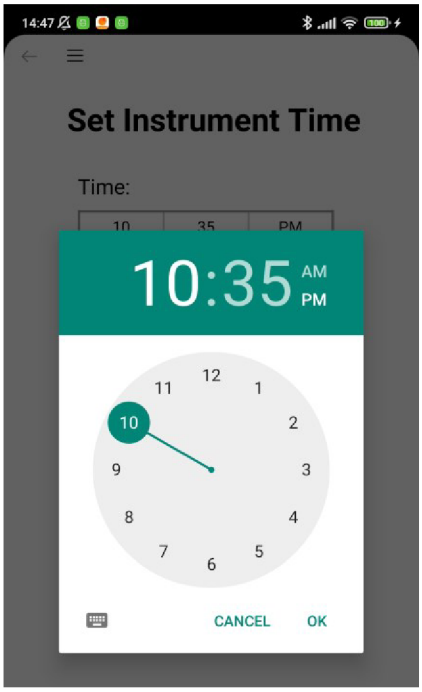
Unikátním prvkem prostředí Uno Platform je reakce na schéma zařízení. Aplikace se dokáže přizpůsobit barevnému schématu, jenž uživatel používá na svém zařízení.

Grafická změna prostředí se provádí nejen při spuštění aplikace, ale i v době používání. Tato vlastnost je velice užitečná pro uživatele, avšak přidává mnoho omezení pro grafickou tvorbu aplikace v prostředí, které již takto obsahuje malý počet prvků pro grafické úpravy. Výstupní aplikace používá tuto schopnost (viz Obrázek 13).




**Obrázek 13:** Tmavý a světlý režim v mobilním prostředí aplikace Uno

Aplikace Uno používá skrytí komponent pro omezení přístupu uživatele. Při spuštění aplikace je uživateli zobrazena žádost o vložení souboru a tlačítko. Po přidání souboru je uživateli zobrazeno navigační menu společně se stránkou Summary. Hlavní stránka je složena ze dvou částí. První částí hlavní stránky je StackPanel, kde je zobrazen text společně s tlačítkem. Druhá část je skryta a obsahuje prvek NavigationView. Tento prvek obsahuje položky, které jsou zobrazeny v navigačním menu a prvek ScrollViewer, obstarávající zobrazení stránky. ScrollViewer zobrazuje vedle navigačního menu stránku, jenž uživatel zvolí kliknutím do menu. Navigační menu lze minimalizovat a přizpůsobuje se šířce okna nebo displeje zařízení. Kód grafického rozpoložení je obsažen v příloze, viz Zdrojový kód 4.




**Obrázek 14:** Nastavení hodin v mobilním prostředí aplikace Uno

Stránky aplikace jsou tvořeny pomocí základních XAML prvků. Textové zobrazení je řešeno pomocí prvků TextBlock, kde je textový obsah dán manuálně v případě, že se jedná o statický text nebo upraven dynamicky dle načtených dat. Pro výběr hodnot je v aplikaci použit prvek ComboBox, který otevírá nabídku s možností výběru. Zadávání číselných hodnot obstarává prvek NumberBox. Tento prvek automaticky neumožňuje vložit jinou než číselnou hodnotu a s přidáním minimální a maximální hodnoty zamezuje vložení neplatného vstupu. Aplikace obsahuje také posuvník, jenž rozšiřuje možnosti zadání číselného vstupu. Speciální hodnoty, jako jsou čas a datum, jsou zprostředkovány pomocí prvků TimePicker a CalendarDatePicker. Oba dva prvky jsou graficky totožné pro všechna testovaná zařízení, avšak Time picker se chová rozdílně ve způsobu zadávání hodnot. V rámci Android zařízení je vstupní hodnota zadána pomocí systémové nabídky hodin, jenž se liší od ostatních platforem (viz Obrázek 14). Vstup pomocí kalendáře je však identický. Lze očekávat, že vstupní prvky se mohou výjimečně lišit i v prostředích, kde nebyla aplikace testována.

### 3.5.4. Načítání a úprava dat

Hodnoty prvků se v aplikaci přizpůsobují v závislosti na vstupním souboru a vstupu uživatele. Při spuštění aplikace žádný prvek, který je použit pro změnu dat, neobsahuje hodnotu, ale pouze pojmenování, jenž slouží k zacílení prvku v prostředí pro práci

s daty. Z pohledu uživatele obsahují všechny prvky hodnoty v momentu, kdy k nim dostane přístup. V rámci aplikace jsou použity dva přístupy pro práci s daty, jeden pro načtení hodnot a druhý pro jejich uložení.

Hlavní stránka aplikace MainPage.xaml obsahuje základní rozvržení aplikace v podobě prvku NavigaionView, jenž obsahuje prvek ScrollViewer (viz Zdrojový kód 4). Tento prvek reprezentuje prostor v aplikaci, který, jak již naznačuje název, umožňuje posouvání v tomto prostoru, ale také zobrazit v něm mnohé další prvky. Výstupní aplikace používá v tomto prostoru prvek zvaný Frame, jenž umožňuje zobrazit obsah libovolné stránky z aplikace ve vlastním prostoru. V případě aplikace Uno je obsahem hlavní stránky a společně s navigačním oknem umožňuje libovolně přepínat její obsah. Při každém přesměrování na jinou stránku ze strany uživatele načte Frame instanci vybrané stránky a spolu s ní se nově načtou všechna data stránky.

Každá stránka v aplikaci obsahuje základní prvky Grid nebo StackPanel, které slouží pro srovnání a přiřazení prostoru vnořeným prvkům. V aplikaci každý základní Grid a StackPanel obsahuje atribut Loaded, který obsahuje název funkce. Tato funkce se zavolá pokaždé, když bude element, včetně jeho vnořených prvků, načten a očekávat další příkazy. Výstupní aplikace obsahuje jednu tuto funkci pro každou stránku a používá ji pro načítání hodnot. Hodnoty prvků jsou prázdné při spuštění aplikace, ale při načítání jednotlivých stránek zajišťuje Loaded funkce přiřazení hodnot ze tříd knihovny přímo do elementů stránky. Aplikace načítá hodnoty při jakémkoliv načtení a data jsou díky tomuto vždy aktuální, avšak za cenu dlouhé odezvy, kterou lze pozorovat při načítání obsáhlejších stránek.

Pro ukládání hodnot jsou zvoleny dva způsoby. Prvním způsobem je použití funkce, jenž lze aktivovat interakcí s prvkem například pomocí kliknutí nebo zaškrtnutí ikonky. Tyto prvky spouští pomocí atributu Click přiřazené funkce. Volaná funkce není pro element vázána jinak nežli pomocí syntaxe Bind. Druhou možnost vyžadují prvky, jenž slouží pro výběr dat, nebo vložení hodnoty. Pro tyto elementy je v aplikaci použit atribut změny, který se liší v závislosti na elementu. Tento atribut také volá funkci uvnitř kódu s daty, avšak tentokrát je tato funkce vázána k prvku, ze kterého vychází. Každá tato funkce vyžaduje událost použitého prvku pomocí upravené třídy EventArgs. Obsah těchto funkcí poté uloží hodnotu z daného prvku do tříd z knihovny. Volání těchto funkcí probíhá při změně vybrané nebo zadané hodnoty prvku.

### 3.5.5. Práce se souborem

Aplikace pracuje s konfiguračním souborem json, který je totožný pro všechny aplikace. Soubor je vyžadován pro zobrazení menu a další navigaci v rámci aplikace. Obsah souboru je převeden do třídy z knihovny a soubor není nadále používán. Pro načítání a ukládání souboru jsou použity asynchronní úlohy obdobně jako v aplikaci Blazor, viz kapitola 3.3.4. Prostředí Uno Platform používá pro práci se soubory API `Windows.Storage`. Toto API poskytuje jednoduché prostředky pro načtení a ukládání souborů, včetně vlastní nabídky. V rámci aplikace je však toto API zásadním problémem a znemožňuje použití aplikace v podobě webové WASM aplikace. I přestože umožňuje jednotný způsob pro práci se soubory, obsahuje několik podob, které nejsou kompatibilní se všemi platformami. Webová aplikace nedokáže použít toto API pro načtení a ukládání souboru, čímž ztrácí pro výslednou aplikaci smysl.

Načítání souboru probíhá v jedné funkci, která se zavolá při kliknutí na tlačítko na úvodní obrazovce. Ve funkci se vytvoří proměnná, jenž zastupuje instanci třídy `FileOpenPicker`. Z pohledu uživatele se při zavolání funkce zobrazí vyhledávací okno s průzkumníkem souborů v prostředí operačního systému Windows, kde jsou zobrazeny pouze soubory typu json. V mobilním prostředí se otevře základní aplikace pro správu souborů. Soubory, jenž neobsahují příponu json, jsou zašedlé a nelze je zvolit. Zvolit lze pouze jeden soubor a při volbě souboru se navigační okno zavře. Soubor je ze strany aplikace převeden do podoby proměnné typu MemoryStream, jenž je poté převedena do textového řetězce a poslána třídám knihovny pro zpracování.

Ukládání souboru je zahájeno pomocí nabídky v navigačním menu. Při kliknutí na položku pro stažení souboru je zavolána funkce pro ukládání. Funkce vytvoří proměnnou, jenž zastupuje instanci třídy `FileSavePicker`. Tato instance obsahuje navíc doporučený název a typu souboru. Ve výstupní aplikaci je pro tyto hodnoty nastaven název config a dva různé typy souboru. Soubor lze tak uložit v podobě json anebo v podobě textového souboru txt. Název a formát si může uživatel změnit a vybrat v prostředí okna souborového systému zařízení.

# 4. Závěr

V teoretické části práce jsou zkoumána vývojová prostředí, jenž umožňuji multiplatformní tvorbu aplikací za pomoci technologie .NET. Zkoumaná prostředí používají stejné principy multiplatformní tvorby pomocí sdíleného aplikačního kódu, avšak každé v jiné podobě.

Výstupem práce jsou tři aplikace zaměřené na výhody jednotlivých prostředí. Umožňují práci s daty v podobě načítání, úpravy a generování souboru a je na ně kladen důraz v rámci grafické jednoduchosti a přenositelnosti mezi podporované zařízení. Součástí aplikací je externí knihovna, obsažena v kapitole 3.2, která omezuje redundanci kódu.

Webovou podobu poskytují platformy Blazor a Uno Platform. Výsledná aplikace tvořená v Uno Platform v rámci multiplatformní tvorby neumožňuje načítat data jednotným způsobem s ostatními platformami a je pro tuto úlohu nepoužitelná. Naopak aplikace Blazor, viz kapitola 3.3, splňuje veškeré požadavky práce, a i přestože je obsažena pouze ve webové podobě umožňuje také poskytovat nativní podobu. Tato aplikace je plně použitelná a snadno rozšiřitelná pro další účely.

Desktop verzi aplikace poskytují nástroje .NET MAUI a Uno Platform. Obě tyto platformy lze spustit v prostředí operačního systému Windows (v případě Uno Platform i Linux emulátoru) a splňují základní funkcionalitu. V rámci aplikace .NET MAUI je použito promítání webového prostředí, viz kapitola 3.4.1, a lze postřehnout nedostatky v rámci chování uživatelského rozhraní, se kterým se toto prostředí potýká. Uno Platform tento problém neobsahuje a vykazuje nejlepší výsledky pro desktop podobu aplikace, avšak navzdory omezení externí knihovny a špatné responzivity. Obě aplikace jsou funkční, avšak vyžadují další úpravy pro plné nasazení.

Mobilní verze aplikací neumožňují plnou funkcionalitu a není vhodné je prozatím dál rozvíjet. I přestože je lze použít, neumožňují ukládání souboru a slouží pouze pro prohlížení obsahu. Tento problém je zaviněn balíčkem .NET, který pro mobilní zařízení neobsahuje možnost zasahovat do souborového systému zařízení a soubor uložit. Mimo tento problém obsahují mobilní aplikace .NET MAUI a Uno Platform veškerou funkcionalitu a jediným jejich nedostatkem je již zmíněná responzivita pro aplikaci Uno Platform.

# Použitá literatura


[1] What is .NET? Introduction and overview. Microsoft technical documentation [online]. Redmond: Microsoft, 2022 [cit. 2022-05-09]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/core/introduction

[2] ROTH, Daniel, Rick ANDERSON a Shaun LUTTIN. Overview to ASP.NET Core. Microsoft technical documentation [online]. Redmond: Microsoft, 2022 [cit. 2022-05-09]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-6.0

[3] ASP.NET Core Blazor hosting models. Microsoft technical documentation [online]. Redmond: Microsoft, 2022 [cit. 2022-05-09]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/blazor/hosting-models?view=aspnetcore-6.0

[4] ASP.NET Core Blazor. Microsoft technical documentation [online]. Redmond: Microsoft, 2022 [cit. 2022-05-09]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/blazor/?view=aspnetcore-6.0

[5] ASP.NET Core Razor components. Microsoft technical documentation [online]. Redmond: Microsoft, 2022 [cit. 2022-05-09]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/blazor/components/?view=aspnetcore-6.0

[6] .NET Multi-platform App UI (.NET MAUI). GitHub [online]. San Francisco: GitHub, c2022 [cit. 2022-05-09]. Dostupné z: https://github.com/dotnet/maui

[7] What is .NET MAUI?. Microsoft technical documentation [online]. Redmond: Microsoft, 2022 [cit. 2022-05-09]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/maui/what-is-maui

[8] REYES, Leomaris. What Is Single Project in .NET MAUI?. Telerik [online]. Sofia: Telerik, 2022 [cit. 2022-05-09]. Dostupné z: https://www.telerik.com/blogs/what-is-single-project-dotnet-maui

[9] Target multiple platforms from .NET MAUI single project. Microsoft technical documentation [online]. Redmond: Microsoft, 2022 [cit. 2022-05-09]. Dostupné z: https://docs.microsoft.com/en-ca/dotnet/maui/fundamentals/single-project

[10] CHARBENEAU, Edward. Blazor Hybrid Web Apps with .NET MAUI. Code: An EPS Company [online]. Houston: EPS Software, 2021 [cit. 2022-05-09]. Dostupné z: https://www.codemag.com/Article/2111092/Blazor-Hybrid-Web-Apps-with-.NET-MAUI

[11] Host a Blazor web app in a .NET MAUI app using BlazorWebView. Microsoft technical documentation [online]. Redmond: Microsoft, 2022 [cit. 2022-05-09]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/maui/user-interface/controls/blazorwebview

[12] ANAND, Vijay. .NET MAUI – Blazor – Getting Started: .NET MAUI and Blazor – Best of both modern technologies for Mobile, Desktop and Web solution from a single codebase. Developer Thoughts: Explore the world of .NET MAUI and Blazor [online]. Chennai: Developer Thoughts, 2021 [cit. 2022-05-09]. Dostupné z: https://egvijayanand.in/2021/11/13/net-maui-blazor/

[13] Uno Platform: Pixel-Perfect. Multi-Platform. C# & Windows XAML. Today. GitHub [online]. San Francisco: GitHub, c2022 [cit. 2022-05-09]. Dostupné z: https://github.com/unoplatform/uno

[14] Uno Platform Pixel Perfect: Pixel Perfect Rendering. Native Performance. Uno Platform [online]. Montréal: Uno Platform, 2022 [cit. 2022-05-09]. Dostupné z: https://platform.uno/pixel-perfect/

[15] Uno Platform app solution structure. Uno Platform [online]. Montréal: Uno Platform, 2022 [cit. 2022-05-09]. Dostupné z: https://platform.uno/docs/articles/uno-app-solution-structure.html

[16] What is Xamarin?. Microsoft technical documentation [online]. Redmond: Microsoft, 2021 [cit. 2022-05-09]. Dostupné z: https://docs.microsoft.com/en-us/xamarin/get-started/what-is-xamarin

[17] What is Xamarin.Forms?. Microsoft technical documentation [online]. Redmond: Microsoft, 2021 [cit. 2022-05-09]. Dostupné z: https://docs.microsoft.com/en-us/xamarin/get-started/what-is-xamarin-forms

[18] Xamarin.Essentials. Microsoft technical documentation [online]. Redmond: Microsoft, 2021 [cit. 2022-05-09]. Dostupné z: https://docs.microsoft.com/en-us/xamarin/essentials/

[19] Xamarin.Forms XAML Basics. Microsoft technical documentation [online]. Redmond: Microsoft, 2021 [cit. 2022-05-09]. Dostupné z: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/xaml/xaml-basics/

[20] Aplikace ENVIS v. 2.0: přehled funkcí a novinky [online]. Liberec: KMB systems, 2021, 9 [cit. 2022-05-09]. Dostupné z: https://www.kmb.cz/ke-stazeni/dokumentace/category/21-aplikacni-poznamky

[21] How to serialize and deserialize (marshal and unmarshal) JSON in .NET. In: Microsoft technical documentation [online]. Redmond: Microsoft, 2022, s. 9 [cit. 2022-05-09]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/standard/serialization/system-text-json-how-to?pivots=dotnet-6-0

# Příloha 1 – Ukázky kódu aplikací

```
namespace ClassLibraryExternal
{
    public class JsonData
    {
        // Install
        public int fnom { get; set; }
        public int unom { get; set; }
        public int inom { get; set; }
        public int pnom { get; set; }
        public int vtRatio1 { get; set; }
        public int vtRatio2 { get; set; }
        public int ctRatio1 { get; set; }
        public int ctRatio2 { get; set; }
        public int multiplierU { get; set; }
        public int multiplierI { get; set; }
        public string? connectionMode { get; set; }
        public string? connectionType { get; set; }

        // Time and Date
        public string? connectionTime { get; set; }
        public string? connectionDate { get; set; }
        public string? timeZone { get; set; }
        public bool daylightSaving { get; set; }

        // Aggregation
        public string? avg1method { get; set; }
        public string? avg1period { get; set; }
        public string? avg2method { get; set; }
        public string? avg2period { get; set; }

        // Memory
        public double voltageEvents { get; set; }
        public double generalOscillograms { get; set; }
        public double pqMain { get; set; }

        // RCS
        public int startPulseHigh { get; set; }
        public int startPulseLow { get; set; }
        public int pulseHigh { get; set; }
        public int pulseLow { get; set; }
        public int messageDuration { get; set; }
        public bool createEvent { get; set; }

    }
}
```

**Zdrojový kód 1:** Třída JsonData

```
using System.Text.Json;

namespace ClassLibraryExternal
{
    public class LoadData
    {
        public static bool isJsonLoaded = false;

        public static JsonData? jData;

        public static string CreateJson()
        {
            return JsonSerializer.Serialize(jData);
        }

        public static void UploadJson(string inputData)
        {
            jData = JsonSerializer.Deserialize<JsonData>(inputData);

            isJsonLoaded = true;
        }
    }
}
```

**Zdrojový kód 2:** Třída LoadData

```
@page "/"

<PageTitle>Summary</PageTitle>

@if (lData.isJsonLoaded.Equals(false))
{
    <div id=c1>
        <h1>Select and load configuration file</h1>
        <label>Select file:</label>
        <InputFile OnChange="@LoadFile" /><br />
        @if (isFileLoaded.Equals(true))
        {
            <button class="btn btn-primary"
                    @onclick="UploadData" >Upload data</button>
        }
    </div>
}
else
{
    <table class="table">
        . . .
    </table>
}

@code {

    string inputText;
    bool isFileLoaded = false;

    async Task LoadFile(InputFileChangeEventArgs e)
    {
        MemoryStream ms = new MemoryStream();
        await e.File.OpenReadStream().CopyToAsync(ms);
        var bytes = ms.ToArray();

        inputText = System.Text.Encoding.UTF8.GetString(bytes);
        isFileLoaded = true;
    }

    private void UploadData()
    {
        lData.UploadJson(inputText);
    }

}
```

**Zdrojový kód 3:** Razor komponenta Summary

```
<Page
  . . . >

  <Grid>

    <StackPanel x:Name="uploadPanel" VerticalAlignment="Center">
      <TextBlock Text="Please load configurationfile"
                 TextAlignment="Center"
                 TextWrapping="Wrap"
                 FontSize="50"
                 FontFamily="Calibri"
                 HorizontalAlignment="Center"
                 Padding="0 0 0 10"/>

      <Button Content="Upload file"
              Click="{x:Bind UploadData}"
              FontSize="40"
              FontFamily="Calibri"
              HorizontalAlignment="Center" />
    </StackPanel>

    <NavigationView x:Name="navView"
                    OpenPaneLength="200"
                    Loaded="NavigationView_Loaded"
                    SelectionChanged="NavigationView_SelectionChanged"
                    IsSettingsVisible="False"
                    Visibility="Collapsed">

       <NavigationView.MenuItems>
         <NavigationViewItem x:Name="itemSummary" Icon="Home"
                             Content="Summary" Tag="summary" />
         <NavigationViewItem x:Name="itemInstall" Icon="Repair"
                             Content="Install" Tag="install" />
         <NavigationViewItem x:Name="itemTimeDate" Icon="Clock"
                             Content="Time and Date" Tag="timedate" />
         <NavigationViewItem x:Name="itemAggregation" Icon="Library"
                             Content="Aggregation" Tag="aggregation" />
         <NavigationViewItem x:Name="itemMemory" Icon="OpenFile"
                             Content="Memory" Tag="memory" />
         <NavigationViewItem x:Name="itemRcs" Icon="Manage"
                             Content="Rcs" Tag="rcs" />
          <NavigationViewItem x:Name="itemDownload" Icon="Download"
                             Content="Downlnoad File" Tag="download" />
       </NavigationView.MenuItems>

       <ScrollViewer>
         <Frame x:Name="ContentFrame" />
       </ScrollViewer>

    </NavigationView>

  </Grid>
</Page>
```

**Zdrojový kód 4:** Hlavní stránka aplikace Uno

# Příloha 2 – CD

Součástí práce je CD, které obsahuje:

- Elektronickou podobu práce ve formátu PDF
- Vstupní konfigurační soubor json
- Spustitelné a instalační balíčky aplikací
- Zdrojové kódy aplikací v podobě projektů Visual Studio
- Textový návod pro orientaci a práci se soubory