Pedagogická fakulta
Faculty of Education

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice


**Jihočeská univerzita v Českých Budějovicích**

Pedagogická fakulta

Katedra informatiky


# Vytváření multiplatformních aplikací prostřednictvím Uno Platform

# Building Cross-Platform Applications via Uno Platform

Bakalářská práce


**Vypracoval:** Dominik Kutil

**Vedoucí práce:** RNDr. Hana Havelková



České Budějovice 2021

# Zadání bakalářské práce

## JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

Pedagogická fakulta

Akademický rok: 2019/2020

## ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

| | |
|---|---|
| Jméno a příjmení: | **Dominik KUTIL** |
| Osobní číslo: | **P18480** |
| Studijní program: | **B7507 Specializace v pedagogice** |
| Studijní obor: | **Informační technologie a e-learning** |
| Téma práce: | **Vytváření multiplatformních aplikací prostřednictvím Uno Platform** |
| Zadávající katedra: | **Katedra informatiky** |

### Zásady pro vypracování

Cílem práce je zmapovat aktuální situaci v oblasti tvorby multiplatformních aplikací, stručně popsat dostupné systémy a technologie a především představit v této oblasti poměrně nový framework, a to Uno Platform. Autor vytvoří tutoriál – uživatelsky srozumitelnou a názornou příručku, v níž na jednoduchých příkladech postupně objasní základní principy vytváření aplikací mobilních resp. webových. Funkcionalita příkladů (triviálních aplikací) bude otestována na platformách iOS, Android a na webu (ve vybraných prohlížečích), autor též porovná případné rozdíly v zobrazování na jednotlivých platformách. Součástí práce bude vlastní souvislý příklad (aplikace), na kterém budou předvedeny základní funkce frameworku Uno Platform. Práce by mohla sloužit jako studijní materiál pro práci s frameworkem Uno Platform – půjde o unikátní příručku, která momentálně v ČR chybí.

| | |
|---|---|
| Rozsah pracovní zprávy: | **40** |
| Rozsah grafických prací: | **CD ROM** |
| Forma zpracování bakalářské práce: | **tištěná** |

Seznam doporučené literatury:


1. Uno Platform Documentation. [Online] Dostupné zhttps://platform.uno/docs/articles
2. Price, M., J. C# 8.0 and .NET Core 3.0 – Modern Cross-Platform Development 4th Edition. Birmingham: Packt Publishing, 2019. ISBN:978-1788478120
3. Microsoft. Dokumentace k nástrojů XAML [Online] Dostupné z: https://docs.microsoft.com/cs-cz/visualstudio/xaml-tools
4. Microsoft. Dokumentace k jazyku C# [Online] Dostupné z: https://docs.microsoft.com/cs-cz/dotnet/csharp/
5. Microsoft. Xamarin. [Online] Dostupné z: https://dotnet.microsoft.com/apps/xamarin


Vedoucí bakalářské práce: **RNDr. Hana Havelková**
Katedra informatiky

Datum zadání bakalářské práce: **9. dubna 2020**
Termín odevzdání bakalářské práce: **30. dubna 2021**

L.S.

**doc. RNDr. Helena Koldová, Ph.D.**
děkanka

**doc. PaedDr. Jiří Vaníček, Ph.D.**
vedoucí katedry

V Českých Budějovicích dne 9. dubna 2020

# Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne .....

.....

Dominik Kutil

# Anotace

Cílem práce je zmapovat aktuální situaci v oblasti tvorby multiplatformních aplikací, stručně popsat dostupné systémy a technologie a především představit v této oblasti poměrně nový framework, a to Uno Platform. Autor vytvoří tutoriál - uživatelsky srozumitelnou a názornou příručku, v níž na jednoduchých příkladech postupně objasní základní principy vytváření aplikací mobilních resp. webových. Funkcionalita příkladů (triviálních aplikací) bude otestována na platformách iOS, Android a na webu (ve vybraných prohlížečích), autor též porovná případné rozdíly v zobrazování na jednotlivých platformách. Součástí práce bude vlastní souvislý příklad (aplikace), na kterém budou předvedeny základní funkce frameworku Uno Platform. Práce by mohla sloužit jako studijní materiál pro práci s frameworkem Uno Platform - půjde o unikátní příručku, která momentálně v ČR chybí.

# Klíčová slova

Uno Platform, C#, XAML, UWP, WASM, Windows, iOS, Android, macOS

# Abstract

The aim of the theses is to map the current situation in the area of multiplatform application creation, to briefly describe the available systems and technologies and, above all, to present a relatively new framework in this area, namely the Uno Platform. The author creates a tutorial - a user-friendly and graphic manual in which he gradually clarifies the basic principles of creating mobile and web applications on simple examples. The functionality of examples (trivial applications) were tested on iOS platforms, Android and on the web (in selected browsers), the author also compares possible differences in viewing on individual platforms. The work will include its own continuous example (application) on which the basic functions of the Uno Platform framework are demonstrated. The work could serve as a study material for working with the Uno Platform framework - it is be a unique manual, which is currently missing in the Czech Republic.



# Keywords

Uno Platform, C#, XAML, UWP, WASM, Windows, iOS, Android, macOS

# Poděkování

Tímto bych rád poděkoval paní RNDr. Haně Havelkové za odborné rady a trpělivost při psaní této bakalářské práce. Dále bych rád poděkoval Martinu Zikmundovi a komunitě Discord serveru UWP Community za rady vztahující se k Uno Platform.

# Obsah

**1 Úvod** **12**

1.1 Východiska práce . . . . . 12

1.2 Cíle práce . . . . . 13

1.3 Metodika práce . . . . . 14

**2 Multiplatformní aplikace** **15**

2.1 Pojem multiplatformní aplikace . . . . . 15

2.2 Výhody multiplatformních aplikací . . . . . 15

2.3 Nevýhody multiplatformního vývoje . . . . . 16

2.4 Metody multiplatformního vývoje . . . . . 16

2.4.1 Nativní vývoj . . . . . 17

2.4.2 Vývoj webových aplikací . . . . . 17

2.4.3 Vývoj hybridních aplikací . . . . . 18

2.4.4 Vývoj napříč platformami . . . . . 19

**3 Platformy** **20**

3.1 Android . . . . . 20

3.1.1 Architektura . . . . . 20

3.1.1.1 Jádro . . . . . 21

3.1.1.2 Hardwarová abstrakční vrstva . . . . . 21

3.1.1.3 Android Runtime . . . . . 21

3.1.1.4 Knihovny . . . . . 21

3.1.1.5 Java API framework . . . . . 22

3.1.1.6 Systémové aplikace . . . . . 22

3.2 iOS a Mac OS X . . . . . 22

3.2.1 Architektura iOS . . . . . 23

3.2.1.1 Cocoa Touch . . . . . 24

3.2.1.2 Media Layer . . . . . 24

3.2.1.3 Core Services . . . . . 25

3.2.1.4 Core OS . . . . . 25
3.2.2 Architektura Mac OS . . . . . 26
3.2.2.1 Cocoa Layer . . . . . 26
3.2.2.2 Media Layer . . . . . 27
3.2.2.3 Core Services Layer . . . . . 27
3.2.2.4 Core OS Layer . . . . . 28
3.2.2.5 Kernel and Device Drivers Layer . . . . . 28
3.3 Microsoft Windows . . . . . 29
3.3.1 Architektura Windows . . . . . 29
3.3.1.1 Režim uživatele . . . . . 30
3.3.1.2 Režim jádra . . . . . 31
3.3.2 Moderní webové prohlížeče . . . . . 31
3.3.2.1 Google Chrome . . . . . 31
3.3.2.2 Firefox . . . . . 32
3.3.2.3 Microsoft Edge . . . . . 32
3.3.2.4 Safari . . . . . 32
3.4 Vývoj aplikací . . . . . 32
3.4.1 Xamarin . . . . . 34
3.4.2 Blazor WebAssembly . . . . . 34
3.4.3 WinUi / UWP . . . . . 35

**4 Uno Platform** **36**
4.1 Struktura platformy Uno . . . . . 37
4.2 Uno vs. jiné frameworky . . . . . 38
4.3 Vývojové prostředí . . . . . 39
4.3.1 Uno Playground . . . . . 39
4.3.2 Visual Studio . . . . . 40
4.3.3 Další možnosti . . . . . 42

**5 Práce v Uno Platform** **43**
5.1 Založení projektu . . . . . 43

5.2 Spouštění aplikace . . . . . 45
5.3 Tvorba . . . . . 46
5.3.1 Schéma XAML . . . . . 46
5.3.2 Ovládací prvky . . . . . 47
5.3.2.1 Prvky pro layout . . . . . 49
5.3.2.2 Základní vstupy . . . . . 56
5.3.2.3 Navigace . . . . . 61
5.3.2.4 Animace . . . . . 64
5.4 Rozdíly v zobrazení . . . . . 66

**6 Aplikace** **68**
6.1 Komplikace . . . . . 68
6.2 Průchod a funkcionalita aplikace . . . . . 70
6.3 Testování aplikace . . . . . 73

**7 Závěr** **76**

**Seznam použité literatury a zdrojů** **83**

**Seznam obrázků** **85**

**Seznam zdrojových kódu** **86**

**Rejstřík** **87**

**A Příloha** **87**

**B Obrázky zobrazení příkladu Prvky pro layout** **88**

**C Obrázky zobrazení příkladu Vstupní prvky** **90**

**D Obrázky zobrazení příkladu Navigace** **92**

**E Obrázky zobrazení příkladu Animace** **94**

F Obrázky sekce Všech míst z aplikace TuristGo 96

# 1 Úvod

Tato bakalářská práce se zabývá problematikou tvorby multiplatformních aplikací prostřednictvím open-source frameworku Uno Platform za pomoci jazyku C# a XAML. Uno Platform je platforma s otevřeným zdrojovým kódem, která umožnuje vývojářům Windows Presentation Foundation, dále jen WPF, a Universal Windows Platform, dále jen UWP, použít své znalosti C# a XAML na více platforem.

Teoretická část je zaměřená na multiplatformovost jako takovou. Když vyvíjíme nějakou aplikaci a máme vybranou metodu vývoje, musíme také vědět, na jakou platformu či platformy cílit. Jednotlivé platformy, na které má Uno Platform výstup, jsou rozebrány dle architektury. U každé z nich používáme k vývoji jinou technologii. Současné metody vývoje budou rozebrány pro každou platformu, a nakonec samotná Uno Platforma, která má výstup na všechny zmíněné platformy.

Praktická část se zabývá již zmíněnou platformou Uno. Tato část je zaměřena na instalaci Uno platformy a uvádí postup instalace potřebných balíčku do prostředí Visual Studia. Uno platforma má výstupy na mnoho platforem, a proto je zde uvedeno nastavení výstupu prostřednictvím emulátoru. Formou malých triviálních příkladů jsou předvedeny metody tvorby aplikací. V mnou vytvořeném souvislém příkladu jsou následně aplikovány jednotlivé metody tvorby aplikací a ukázány základní funkce, které nabízí framework Uno Platform.

## 1.1 Východiska práce

V dnešní době existuje mnoho technologií pro přístup k jednotlivým platformám. Pro vývoj každé platformy je potřeba vybrat správný nástroj a chceme-li cílit třeba na tři platformy najednou, je zapotřebí pro každou platformu speciální nástroj. Multiplatformní technologie nám umožnují cílit na více platforem současně, ale i tak nelze zacílit na všechny zároveň.

Možností, jak zacílit na více platforem zároveň, je například Uno Platform od kanadských vývojářů ze společnosti Nventive. Uno platforma má skrze jednu kódovou základnu výstup na iOS, macOS, Windows, Android, moderní webové prohlížeče pomocí WebAssembly, dále jen WASM a ke konci roku 2020 se přidává také Linux. Kódová základna je tvořena pomocí XAML a C#.

Existuje několik návodů v angličtině od vývojářů Uno platformy, které mohou být složité na pochopení. V češtině zatím nebyla sepsána obsáhlejší příručka, která by popisovala samotný framework Uno Platform a metody tvorby aplikací jeho prostřednictvím.

## 1.2 Cíle práce

Cílem mé bakalářské práce je představení poměrně nového open-source frameworku pro tvorbu multiplatformních aplikací, a to Uno Platform. Práce si klade za cíl představit framework, jeho charakteristiku, architekturu a co může nabídnout. Výstižně a srozumitelně popíšu možnosti tvorby multiplatformních aplikací s výstupy na platformy Android, iOS, Windows a WASM.

V teoretické části se pokusím o zmapování situace v oblasti tvorby multiplatformních aplikací, přičemž také nastíním, co jsou to multiplatformní aplikace, proč se tvoří a jaké metody jsou používané k jejich vývoji. Těmi jsou například nativní, webový nebo hybridní vývoj aplikací. Vyložím platformy, na které má Uno Platform výstup, a to z hlediska jejich architektury a využívaných technologií, na které spoléhají jednotlivé platformy, jimiž jsou například Xamarin, WASM a WinUI / UWP.

Praktická část stručně popíše instalaci Uno Platform a instalace emulátorů pro výstupy aplikací. Dále se vytvoří příručka, v níž se objasní základní principy a metody vývoje aplikací formou triviálních příkladů, a také popíšu rozdíly v zobrazení. Výstupem této bakalářské práce bude

aplikace, ve které budou aplikovány mnou vyzkoušené postupy, způsoby práce a dovednosti, které budu popisovat ve své práci. Z toho vyplývá, že by má práce mohla být v budoucnu využita jako stručná příručka pomáhající jejím čtenářům zorientovat se a pochopit práci v Uno Platform.

## 1.3 Metodika práce

Než se budu zaobírat samotnou Uno platformou, je potřebné představit multiplatformovost a popsat jednotlivé platformy.

Úvod práce bude věnován multiplatformním aplikacím. Multiplatformovost bude analyzována z pohledů, významu pojmu multiplatformní aplikace, porovnání výhod a nevýhod multiplatformního vývoje a vývoj samotný. Rozebrány budou jednotlivé metody vývoje(nativní, webový, hybridní nebo multiplatformní vývoj).

Dále budou analyzovány jednotlivé platformy, se kterými pracuje Uno Platform. U každé z nich bude prozkoumána architektura a metoda vývoje pro cílení aplikace na konkrétní platformu. Poté bude popsána samotná platforma Uno, konkrétně instalace, možnosti výstupu a celkově její specifikace.

Následně bude objasněna syntaxe platformy Uno za pomoci C# a XAML prostřednictvím malých triviálních příkladů. Tyto příklady nám objasní tvorbu prvků, jako jsou tlačítka, různé druhy textových boxů a podobné prvky. Výsledkem bude souvislý příklad v podobě aplikace, která předvede a objasní tvorbu aplikace v platformě Uno.

# 2 Multiplatformní aplikace

Většina lidí si chce své aplikace spustit na všech svých zařízeních a očekávají, že se aplikace spustí, ať má jejich zařízení jakýkoliv operační systém. Před multiplatformním vývojem aplikací bylo nutné vyvíjet aplikace pro konkrétní systémy (například Android, iOS nebo desktopy) zvlášť, což vyústilo potřebu větších vývojářských týmů a financí pro vývoj.

## 2.1 Pojem multiplatformní aplikace

Multiplatformní aplikace je označení pro software, který pracuje na více platformách, kterými se myslí zařízení či operační systémy. Mezi tyto platformy můžeme zařadit Windows, macOS, Android či iOS. Tyto platformy popíšu později v kapitole Platformy. [1]

## 2.2 Výhody multiplatformních aplikací

Na výhody multiplatformního vývoje softwaru se lze dívat z mnoha pohledů.

Z pohledu zákazníka, který má více zařízení jako jsou tablet, notebook, desktop nebo telefon. Tato zařízení však povětšinou nemají stejný operační systém. A zde lze využít multiplatformní vývoj softwaru, jenž umožní použití uvedeného softwaru na více zařízeních s odlišným operačním systémem. [2]

Z podnikatelského pohledu jsou jasnou výhodou nižší náklady. Není potřeba mít pro každou platformu vývojářský tým a technologii. Dále tu máme větší uplatnění na trhu, kde můžeme aplikace publikovat ve více obchodech, což může zajistit, že se aplikace rychleji uchytí a osloví více potenciálních zákazníků. Jedním z dalších důvodů je minimalizace počátečních nákladů: prostřednictvím multiplatformního vývoje se rychleji připraví první verze aplikace pro vyzkoušení toho, jak bude vnímána na trhu. S tím určitě souvisí analýza trhu, která nám odhalí, kam by měla aplikace směřovat a tím můžeme aplikaci rychleji přizpůsobit. [2, 3]

V potaz musíme vzít i pohled vývojáře. Napsání zdrojového kódu pouze jednou je velké plus nejen pro firmu, ale i pro vývojáře samotného. Z tohoto plusu vyplývá opakovaně použitelný kód, a to až 80 %, který zapříčiní větší efektivitu a produktivitu, a tak může být kód použit i při jiném vývoji aplikací. Díky zvýšené rychlosti vývoje se nám zrychlí i prototypování. Prototypování zajištuje, že jsme s produktem "na dobré cestě". Rychlost vývoje se zvyšuje kvůli konkurenci a potřebám zákazníků. To přivodí zvyšování tlaku na vývojáře, kteří musí dodržovat termíny. Dále máme snadnější údržbu kódu, která vyžaduje opravy chyb, vylepšení aplikace, ale také testovaní aplikace, které opět bude rychlejší kvůli testování pouze jednoho společného kódu. [2, 3]

## 2.3 Nevýhody multiplatformního vývoje

Oproti všem kladům vývoje napříč platformami se můžeme setkat s nevýhodou: při použití pokročilejších funkcí může mít aplikace větší spotřebu baterie zařízení a zdrojů oproti nativnímu vývoji. [3]

Dále můžeme narazit na nedostatek hardwarové síly potřebné k plynulosti animací HTML5. Na rozdíl od nativního vývoje můžeme zaznamenat mnohem pomalejší výkon. Také se můžeme setkat s omezeným přístupem k funkcím kamery nebo mikrofonu. Multiplatformní aplikace nedokážou využívat výhody nativních komponentů User Experience[1](UX), uživatel se v aplikaci poté nemusí úplně vyznat kvůli zbytečným informacím a bude ztrácet čas. [3, 4, 5]

## 2.4 Metody multiplatformního vývoje

Vzhledem k mnoha možnostem přístupu pro vytváření aplikací je velice důležité uvážit, jakou metodu vybrat pro náš projekt. Jeden z aspektu, který musíme do úvahy zahrnout, je složitost aplikace. Záleží na tom, s čím bude aplikace

[1]User Experience - uživatelské použití aplikace vycházející z uživatelových reakcí

pracovat – s daty ze sítě, nebo hardwaru zařízení. Zřetel bude kladen také na časovou náročnost vývoje či na náklady s vývojem spojené. [4]

### 2.4.1 Nativní vývoj

Nativní vývoj je jednou ze čtyř důležitých metod přístupu a tato metoda patří ke starším metodám vývoje aplikací. Charakteristický rys pro tento přístup je vyvíjení kódu pro každou platformu zvlášť, k čemu se využívají zvláštní nástroje a programovací jazyky. Pokud budeme chtít zacílit na web, Android, iOS, Windows nebo macOS, budeme vyvíjet až pět řešení, z čehož vyplývá, že náročnost vývoje zapříčiní zvýšenou časovou a finanční náročnost, nebo pro každou zvolenou platformu bude potřeba speciální vývojářský tým, který má zkušenosti s konkrétní platformou. Tot musíme mít na paměti, i když vezmeme v potaz testování i skutečnost, že časem budeme opravovat nebo aktualizovat každou platformu zvlášť.

Nativní vývoj je vhodné použít pro práci s velkým objemem dat a přístupem k hardwaru. Tato metoda nabízí ze všech možností vývoje aplikací nejvyšší výkon. [6, 7]

### 2.4.2 Vývoj webových aplikací

Z pohledu náročnosti vývoje, ale i nákladů jsou nejsnazší metodou webové aplikace. Webové aplikace jsou vyvíjeny prostřednictvím HTML, CSS nebo JavaScript technologií a realizovány prostřednictvím prohlížeče, na kterém závisí výkon aplikace. Webové aplikace máme dvojího typu.

Prvním typem jsou jednostránkové webové aplikace, dále jen SPA. Aplikace tohoto druhu zachycuje svůj obsah na jedné stránce. Když se zapne aplikace, stáhnou se kompletní data, která jsou lokálně uložena a dynamicky načítána, oproti jiným aplikacím, které jednotlivá data stahují ze serveru, až když se s těmito daty má pracovat. Tento typ webových aplikací se může například použít u formulářů, jako jsou například evidence. [6, 7, 8]

Druhým typem jsou progresivní webové aplikace, dále jen PWA. Tyto webové aplikace se svým chováním a vzhledem snaží imitovat mobilní aplikace. Zásadní výhodou PWA je lepší přístupnost k hardwaru zařízení v podobě notifikací nebo tak práce bez sítě. [9]

Tento typ vývoje je vhodné zvolit kvůli nižším nárokům na výkon, menší náročnosti vývoje či přístupu k aplikaci z libovolného operačního systému. Určitě bychom neměli opomenout, že webová aplikace se nemusí instalovat a také není třeba používat obchod s aplikacemi jako je Google Play. Aplikace tohoto druhu je vhodné použít pro vývoj menších a nenáročných aplikací, které nepotřebují pracovat s velkým objemem dat a přístupy k hardwaru. [6, 7]

### 2.4.3 Vývoj hybridních aplikací

Vývoj hybridních aplikací sdružuje všechny složky nativní a webové varianty vývoje. Hlavní část stejně jako u webových aplikací používá HTML, CSS nebo JavaScript. Prostřednictvím JavaScript frameworku (jako například NativeScript, Xamarin nebo React Native) má takováto aplikace přístup téměř ke všemu hardwaru zařízení. V zařízeních se aplikace přizpůsobuje každému přístroji zvlášť, a tím se chová jako nativní, ačkoliv je vyvíjena jako webová aplikace.

Podobně, jako tomu je u vývoje napříč platformami, je vyvíjena pouze jedna kódová základna a ta je dále uzpůsobena pro různá zařízení a operační systémy. Velkou předností „hybridního“ vývoje je jeho rychlost. Během vývoje se ušetří hodně času, protože se pracuje pouze s jednou kódovou základnou. Mezi další výhody tohoto typu vývoje můžeme zařadit multiplatformovost, kdy je aplikace tvořena, aby fungovala na více platformách, nižší náklady díky nutnosti pouze jednoho vývojářského týmu nebo možnostmi distribuce skrze obchody. [6, 7, 10]

### 2.4.4 Vývoj napříč platformami

Tento typ je obdobný metodě hybridního vývoje, který také umožnuje konstruovat aplikace napříč platformami. Aplikace jsou překládány do nativního kódu, tudíž nejsou spouštěny jako hybridní aplikace skrze webové prohlížeče, což umožní spouštět aplikace s nativními funkcemi a výkonem.

Apliakce tohoto typu mají pouze jeden sdílený kód, který nám sníží náklady potřebné na vývoj aplikace. Tento kód je možné opětovně použít, což ušetří opakování při vytváření, čas i prostředky. Když je vyvíjena na více platforem pouze jedna aplikace, je snazší její údržba, ale tak aktualizace, která se synchronizuje na všech platformách a zařízeních. Díky jednomu kódu pro všechny platformy máme pro všechny i jednotný design. [7, 11]

Oproti všem kladům vývoje napříč platformami se můžeme setkat s nevýhodou – při použití pokročilejších funkcí může mít aplikace větší spotřebu baterie zařízení a zdrojů oproti nativní. Dále se můžeme setkat nedostatkem hardwarové síly potřebné k plynulosti animací HTML5. [12]

# 3 Platformy

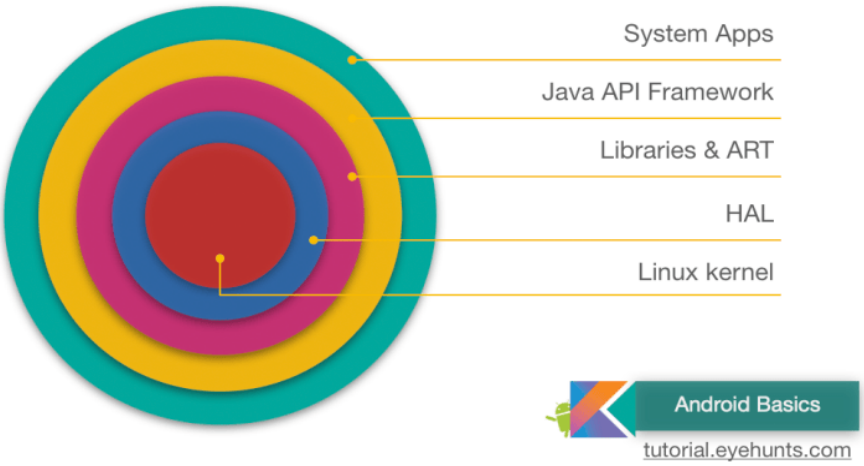
## 3.1 Android

Mobilní operační systém Android, který má linuxové jádro, byl spuštěn společností Android Inc. Ovšem v roce 2005 ho převzal Google a učinil tak konkurenci společnostem jako jsou Nokia nebo Apple. [13]

Pod vedením společnosti Google je poskytován vývojářům jako open-source pod licencí Apache, což umožnuje stáhnou zdrojový kód. Vytváření aplikací je díky dostupnému zdrojovému kódu přívětivější a aplikace lze distribuovat pomocí obchodu Google Play. [14]

### 3.1.1 Architektura

Android architektura je rozdělena do šesti základních sekcí. Každá z nich má svou funkci a přístup (komunikaci nebo adaptaci) závislé na požadavcích vývoje. [15]




Obrázek 1: Android architektura (převzato z [15])

#### 3.1.1.1 Jádro

Podstatou Android platformy je linuxové jádro, které se nachází v základu architektury Android. Důvodem výběru tohoto druhu jádra jsou spolehlivé základní funkce, které jsou vhodné pro vývoj Androidu. Linuxové jádro zpracovává zabezpečení mezi aplikací a systémem, síťovou komunikaci, spravuje paměť, řízení procesů a správu napájení systému. Vzhledem k udržovanému a předpřipravenému jádru operačního systému poskytuje Android dobrou základnu vývojářům. [13, 15]

#### 3.1.1.2 Hardwarová abstrakční vrstva

Standardní rozhraní nám poskytuje hardwarová abstrakční vrstva, dále jen HAL. Rozhraní pracují s možnostmi hardwaru vyšší úrovně Java rámce API[2]. HAL nám umožnuje skrze několik modulů a knihoven přistupovat ke konkrétním rozhraním (například Bluetooth, kamera zařízení, senzory) a použít je do svých aplikací. [15]

#### 3.1.1.3 Android Runtime

Od verze 5.0 a vyšší běží každá aplikace ve vlastním prostředí s vlastní instancí Android Runtime, dále jen ART. Před verzí 5.0 byl jako virtuální stroj používán Dalvik. ART je virtuální stroj, který vytvoří prostředí pro běh aplikací. Aplikace jsou ukládány do paměti RAM a spouští se v krátkém čase. Jedná se tedy o virtuální stroj pro každou spuštěnou aplikaci. ART obsahuje sady knihoven, které jsou pro vývoj aplikací na Android pomocí jazyka Java nezbytné.[13, 15]

#### 3.1.1.4 Knihovny

Tato vrstva obsahuje knihovny C / C++ používané komponentami Androidu. Pro přístup ke knihovnám nativní platformy se používá sada Android NDK. Mezi funkce knihovny patří například ukládání dat pomocí databází,

[2]API - rozhraní pro programování aplikací

procházení webů, správa a vykreslování ploch aplikací, frameworky s kodeky pro audio a video, knihovny pro práci s 3D a 2D vykreslování ... [13, 15]

#### 3.1.1.5 Java API framework

Jde o kolekci API napsaných v jazyce Java, které poskytují vývojářům přístup k celé sadě funkcí systému Android. Pomocí zjednodušení opakovaného použití hlavních, modulárních systémových vlastností a služeb takováto API tvoří stavební bloky potřebné k tvoření aplikací pro Android. Tato vrstva umožňuje například správu polohy zařízení, oken a kreslících ploch, balíčků aplikací, oznámení, dat aplikací nebo uživatelského rozhraní. [13, 15]

#### 3.1.1.6 Systémové aplikace

Android má ve svém systému zabudované výchozí a základní aplikace. Mezi ně můžeme zařadit prohlížeč internetu, mapy, kontakty, kalendáře, emaily, zasílání zpráv a mnoho dalších. Základní aplikace jsou uživatelsky dostupné ke stažení z obchodu Google Play Store.

Android má vestavěné funkce jako softwarový zásobník, což umožnuje plynulý chod systému na jakémkoli zařízení. Z pohledu vývojáře není třeba vytvářet funkcionalitu, například pro posílání SMS, ale stačí zavolat libovolnou aplikaci na zasílání SMS, která je v zařízení nainstalována. [13, 15]

## 3.2 iOS a Mac OS X

Mobilní operační systém iOS, dříve známý jako iPhone OS, je vyvíjený společností Apple. iOS je unixový systém[3] odvozený od Mac OS X. Tento mobilní systém může být na rozdíl od Androidu nainstalován pouze na zařízení společnosti Apple. Podobná situace je i v případě instalací aplikací, které musí být výhradně z obchodu App Store. O prolomení těchto omezení

[3]Unix systém - operační systém firmy AT&T

v oblasti instalace aplikací z jiného zdroje než z obchodu App Store, se „snaží“ softwarové nástroje JailBreak. [16, 17]

Při vydání první verze systému iOS v roce 2007 nesla označení iPhone OS 1, avšak v roce 2010 byla přejmenována z původního označení na dnes již známé iOS. Jednotlivé verze jsou číselně označovány od 1.0 až po 14.4, která je momentálně nejnovější verzí. [30]

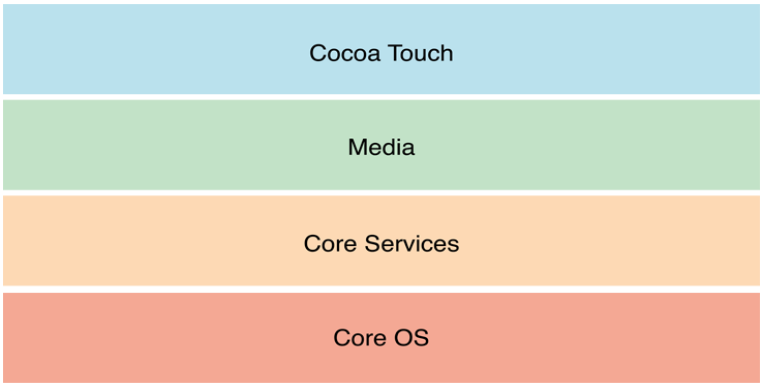
Operační systém Mac OS X je také jako iOS od společnosti Apple a je rovněž založen na Unixu. Unixový základ Darwin pro systém Mac má open-source kód a je samostatně dostupný jako fungující operační systém. [18]

Mac OS X určený pro řadu počítačů Macintosh je navržen aby fungoval na zařízeních vyrobený společností Apple. Předchůdcem systému Mac OS X byl Mac OS. Systém Mac byl spuštěn již v období systémového standardu MS-DOS. Funkce a služby, které může poskytnout, se podobají systémům, jako jsou OS Windows a Linux. [19]

Tento typ je až druhou řadou tohoto systému. První řada nesla označení Mac OS a patřili do ní například Macintosh System Software nebo Mac OS 8 a 9. Do řady Mac OS X lze zařadit verze Mac OS X 10.0 až po současnou verzi Mac OS X 11.0 s označením Big Sur. [31]

### 3.2.1 Architektura iOS

iOS a Mac OS mají velmi podobnou architekturu. Znatelný rozdíl je na nejvyšší úrovni tedy v rozhraní. V ostatních vrstvách jsou menší rozdíly. iOS vytyčený pro mobilní zařízení, poskytuje hlavní komunikaci se zařízením za pomoci dotykové obrazovky. Oproti tomu se Mac OS zaměřuje na desktopové a přenosové počítače, pro které je primární komunikace realizovaná pomocí přes klávesnice a myši. [20]




Obrázek 2: iOS architektura (převzato z [21])

#### 3.2.1.1 Cocoa Touch

Vrstva Cocoa Touch, která je vrcholem iOS architektury, vymezuje aplikační infrastrukturu nebo podporu klíčových vlastností jako multitasking, dotykový vstup nebo notifikace. Nejvýznamnější framework této vrstvy, UIKit, poskytuje zázemí pro aplikace grafického typu nebo zajišťuje aspekty jako jsou multitasking, přístupnost, notifikace, správa aplikací, podpora grafiky a oken nebo podpora tisku.

Ostatní frameworky této vrstvy zastávají funkce pro emailové zprávy, práci s kontakty, mapové rozhraní nebo práci s kalendářem. Mezi obecné funkce této vrstvy patří například poskytování lokálních notifikací pro aplikace, tiskové úlohy a navrhování uživatelského rozhraní. [21]

#### 3.2.1.2 Media Layer

Mediální vrstva, jak vystihuje název, obsahuje multimediální prvky, například grafické, video a zvukové technologie. V mediální vrstvě se nacházejí klíčové technologie, jako jsou nástroje pro 2D a 3D kreslení, zprostředkovávání prostorového zvuku pro aplikace nebo podporu nativních zvuků. Tato vrstva také poskytuje funkce pro práce s textem a fonty nebo také poskytuje knihovny pro iTunes[4]. V mediální vrstvě jsou též obsaženy

[4]iTunes - aplikace pro správu a přehrávání multimédií

frameworky, které nám umožnují například přehrávání audia, vykreslování grafiky nebo filtrování obrázků. [21]

#### 3.2.1.3 Core Services

Vrstva Core Services má na starosti spravování základních služeb systému důležitých pro všechny aplikace. Tato vrstva zprostředkovává přístup k řadě potřebných funkcí, které mohou poskytovat šifrování uživatelských dat, možnost sdílení uživatelských dat přes iTunes, práci s databází, přístup cloudového uložiště a mnoho dalších funkcí.

Hlavním přínosem této vrstvy je Automatic Reference Counting, dále jen ARC. ARC je druh kompilátoru, usnadňující proces správy životnosti objektů, což značně zjednodušuje správu paměti a snižuje potřebný čas. Mezi služby poskytované frameworky, které vrstva Core Services umožňuje použít, patří pracování s uživatelskými účty a poskytování modelu těchto účtů, zpřístupnění reklamních účelů, správu datového modelu ... [21]

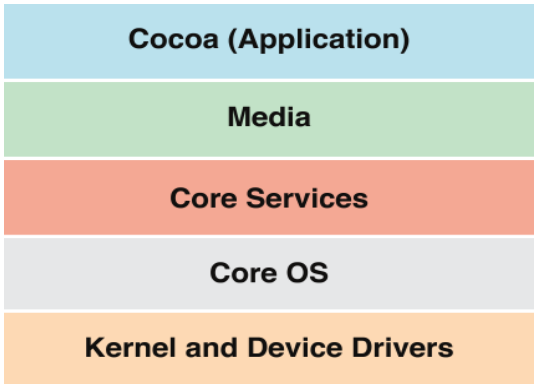
#### 3.2.1.4 Core OS

Zmíněné platformy – jak iOS, tak i macOS – sdílejí stejné jádro založené na Unixu zvané Darwin. Vrstva Core OS má v sobě zabalené jádro z bezpečnostních důvodů. Tato vrstva umožnuje pomocí funkcí nízké úrovně poskytovat funkcionality funkcí zprostředkovávané z vyšších úrovní.

Frameworky zprostředkovávané touto vrstvou mohou být přímo využívány námi napsanými aplikacemi. Mezi funkce těchto frameworků patří provádění výpočtů lineární algebry, zpracovávání obrazu, Bluetooth, komunikace s externími zařízeními, bezpečností zařízení, bezpečnost dat a prostředí jádra, ovladače a nízkoúrovňová rozhraní systému Unix. [21]

### 3.2.2 Architektura Mac OS

Architektura tohoto systému se skládá z mnoha vrstev. Nižší vrstvy poskytují základní služby, na které spoléhá většinu softwaru. Mac OS a iOS si jsou velmi podobné, jak jsem již popsal v kapitole Architektura iOS.




Obrázek 3: MacOS architektura (převzato z [22])

#### 3.2.2.1 Cocoa Layer

Aplikační vrstva Cocoa je důležitá především pro vzhled aplikací a reakce na uživatelovy akce.

Mezi funkce aplikační vrstvy patří notifikace aplikací, možnosti sledování herního žebříčku a porovnávání výsledků s jinými hráči, sdílení obsahu napříč mnoha služeb, zapínání aplikací po opětovném zapnutí zařízení, zobrazení vyskakujících oken, ...

Klíčovým frameworkem pro tuto vrstvu je AppKit, který implementuje uživatelské rozhraní (UI) aplikace obsahující okna, dialogy, ovládací prvky, nabídky a události. Tento framework obsahuje také funkce jako dvojitý dotyk pro přiblížení, uspořádávání oken a jejich přepínání, sdílení jednoho zařízení mezi více uživateli. Další frameworky této vrstvy poskytují například doplňky pro uživatelské rozhraní a bezpečnostní prvky, například autorizace, implementace spořiče obrazovky ... [22]

#### 3.2.2.2 Media Layer

Mediální vrstva nám umožňuje do aplikací začlenit 2D a 3D grafiku, animace, zvuk, různé obrazové efekty a další obrazové funkce. Mac OS podporuje více než 100 typů medií. Tuto vrstvu lze rozdělit na tři části. grafickou, zvukovou a video.

Grafická část je velmi obsáhlá, obsahuje například technologie pro tvorbu grafiky a kreslení, kreslení jednoduchých tvarů jako jsou čáry nebo obdélníky, 2D nativní vykreslování, typografické služby, zpracovávání obrázků prostřednictvím filtrů, správu barev, tisku ...

Zvuková část se skládá ze tří frameworků rozdělených do úrovní. V nejvyšší úrovni se nachází funkce pracující s přehráváním, úpravou, analýzou a nahráváním zvuku. Střední úroveň obsahuje framework pro implementaci multiplatformního standardu API pro 3D zvuk a nakonec nižší úroveň je složena z více frameworků, které zprostředkovávají zvukové služby jako nahrávání, přehrávání, synchronizaci, zpracování signálu, prostorový zvuk nebo převod formátu.

Video část obsahuje technologie poskytující funkce jako přehrávání vizuálního obsahu v aplikacích, přístup k hardwaru médií, model kanálu pro digitální video a technologie napomáhající přehrávání, nahrávání, čtení, kódování, psaní nebo úpravě audiovizuálních médií. [22]

#### 3.2.2.3 Core Services Layer

Tato vrstva obsahuje technologie poskytující základní služby aplikacím, ale nemá vliv na rozhraní aplikací. Technologie z této vrstvy jsou závislé na technologiích a frameworcích ze dvou nižších vrstev.

Klíčové technologie obsažené v této vrstvě zprostředkovávají základní informace o účtech sociálních médií, jako je například Facebook, cloudové úložiště, funkce pro vyhledávání síťových zařízení, přístup k databázím SQL, rozpoznávání a reakce na mluvený jazyk ...

Základní frameworky této vrstvy shodné s iOS jsou uživatelské účty, kontakty nebo datový model. Dalšími příklady jsou kalendář a jeho události nebo připomenutí, zobrazování miniatur obrázků a náhledů dokumentů, funkce pro platby za nákupy v Mac App Store, zobrazování stránek HTML ... [22]

#### 3.2.2.4 Core OS Layer

Technologie a frameworky obsažené v této vrstvě zprostředkovávají služby související s hardwarem a sítěmi. Mezi funkce této vrstvy řadíme blokování instalací softwaru majícího jiný původ než z Mac Apple Store nebo vhodně certifikovaných aplikací, pomyslné poslední zabezpečení před poškozením, smazáním nebo odcizením dat, detekci zásahu do kódu aplikací ...

Frameworky, které jsou v rámci této vrstvy, umožňují například upozornění aplikacím při připojení či odpojení lokálního nebo externího svazku včetně informací o nich, poskytování výpočtů pro grafický procesor, dále jen GPU, a centrální procesovou jednotku, dále jen CPU, poskytování dat, která mohou být uložená v místních nebo síťových databázích, funkce pro uspořádávání dostupných sítí dle dosažitelnosti a možnosti připojení nebo poskytovávání API pro urychlení složitých operací prostřednictvím vektorové jednotky. [22]

#### 3.2.2.5 Kernel and Device Drivers Layer

Oproti iOS architektuře se tato vrstva uvádí zvlášť. V iOS jsou jádro a ovladače součástí vrstvy Core OS.

Jádro Darwin vyvinuté na základně Unixu s prostředím Mach je nejdůležitější částí 64bitového jádra. Mach zprostředkovává nejdůležitější funkce operačního systému, například chránění paměti, preemptivní multitasking, pokročilejší virtuální paměť a podpora reálného času. Součástí Darwinu je také funkce pro rozšíření síťových jader nebo framework I/O Kit umožňující vývoj ovladačů zařízení.

Berkeley Software Distribution, dále jen BSD, je integrovaná spolu s Darwinem. BSD slouží jako základna pro sítové rozhraní a souborové systémy. Mezi hlavní vlastnosti BSD patří sdílená paměť, fronty jádra, události, technologie pro souborové systémy a svazky, zabezpečení systému, podpora standardních síťových protokolů, technologií a diagnostik sítě ... [22]

## 3.3 Microsoft Windows

Série Windows jsou operační systémy od společnosti Microsoft. Novodobé verze operačního systému Windows obsahují grafické rozhraní, dále jen GUI, ve kterém může uživatel pracovat se složkami a soubory. Systém Windows je vhodný pro použití jak v domácnosti, tak i pro profesionální účely. [32]

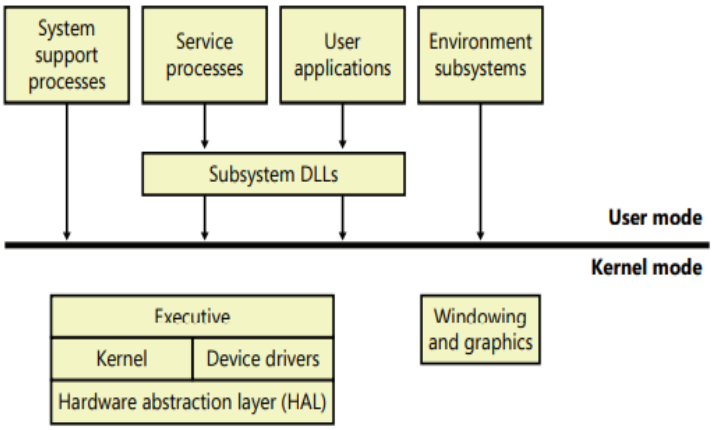
Prvními verzemi systému Windows byly Windows 1.0 až 3.1. Vydáním poslední zmíněné verze se Windows uchytil na trhu jako hojně užívaný operační systém. Následovaly verze Windows 9x, které byly grafickou nadstavbou MS DOS[5]. Nejnovějším typem systému Windows jsou verze Windows NT - verzí Windows NT 3.1 počínaje a nejnovější verzí Windows 10 konče. [28]

### 3.3.1 Architektura Windows

Přestože je struktura tohoto systému velice obsáhlá, s ohledem na zaměření této publikace bude architektura představena pouze stručně.

Zjednodušená architektura operačního systému Windows, kterou jsem zvolil, je rozdělená na dvě části uživatelskou část a část pro jádro. [23]

[5]MS DOS - operační systém firmy Microsoft




Obrázek 4: Architektura systému Windows (převzato z [23])

#### 3.3.1.1 Režim uživatele

Tato úroveň obsahuje čtyři základní typy procesů.

První proces je System support processes. Tato skupina procesů pro podporu systému potřebují operace. Mezi tyto procesy řadíme tvorbu a správu relací, správu navázaných uživatelských relací, bezpečnostní zásady jako uživatelská a skupinová práva nebo oblast hesel ...Dále tento proces umožňuje přihlašování a odhlašování uživatele ve spolupráci se zabezpečením, úlohy jako inicializace infrastruktury Windows, vytvoření správce podpory služeb a procesy lokálního zabezpečení nebo spouštění, zastavení a komunikace se službami.

Mezi další procesy uživatelské úrovně patří Service processes, mezi které řadíme hostující služby, jako je plánovač úloh a řadič tisku, User application. Dále sem řadíme Environment subsystem, skládající se z více částí majících na starosti vytváření a mazání procesů, zachycování vstupů z periferních zařízení, poskytování podpory pro konzolové aplikace a funkce Windows, které jsou implementovány jako součást jádra.

Na konec máme Subsystem DLLs, jehož hlavním úkolem je přeložit funkci do správného volání systémové služby. Uživatelské aplikace nevolají přímo nativní služby Windows, ale procházejí více knihovnami DLL. [23]

#### 3.3.1.2 Režim jádra

Vrstva jádra se skládá z několika částí. První z nich je částí je Windows Executive, který zprostředkovává základní služby, jako jsou řízení paměti a vláken, zabezpečení, I/O, síťování a mezi procesovou komunikaci.

Jádro operačního systému Windows je složeno z nízkoúrovňových funkcí. Mezi ně patří plánování vláken, odesílání a přerušení výjimek a víceprocesová synchronizace. Dále zprostředkovávají sady prostředků a rutin pro konstrukce vyšší úrovně.

Do režimu jádra zařadíme také ovladače. Device Drivers obsahují dva hardwarové ovladače. První z nich pracuje s funkcí I/O pro požadavky uživatele a ovladače hardwarových zařízení jako například systémové soubory a síťové ovladače.

Další část vrstvy jádra je HAL. HAL obklopuje jádro, ovladače a ostatní části Windows Executive. Tato abstrakční vrstva umožní poskytnutí přímého rozhraní s hardwarem, na kterým je spuštěný systém Windows.

Poslední částí je Windowing and graphics system implementující funkce GUI jako jsou práce s okny, malování a ovládací prvky UI. [23]

### 3.3.2 Moderní webové prohlížeče

#### 3.3.2.1 Google Chrome

Prohlížeč Chrome vydaný v roce 2008 byl založen na projektu s otevřeným zdrojovým kódem Chromium. Tento prohlížeč od společnosti Google je dostupný pro operační systémy Windows, Mac OS X, Linux, Android a iOS. Chrome se stal prvním uvedeným webovým prohlížečem obsahujícím adresový řádek a vyhledávací pole. Každý otevřený web v prohlížeči se spouští jako vlastní proces, což zvyšuje bezpečnost. Webové standardy jako kaskádové styly (CSS) a HTML5 jsou samozřejmostí. Na vzhledu Chromu byl založen cloudový operační systém Chrome OS zaměřený na webové aplikace. [24]

#### 3.3.2.2 Firefox

Webový prohlížeč Firefox byl vydán oficiálně v roce 2004 a stal se velice populární alternativou Microsoft Internet Exploreru, a to pro lepší bezpečnost. Od roku 2017 se stal čtvrtým nejpoužívanějším prohlížečem a v tento rok dostal novou technickou základnu Firefox Quantum. Tato základna dvakrát zrychlila chod prohlížeče. Prohlížeč má stejnou dostupnost pro operační systémy jako prohlížeč Chrome. [25]

#### 3.3.2.3 Microsoft Edge

Microsoft Edge je nástupce zastaralého prohlížeče Microsoft Internet Explorer, jenž byl součástí operačních systému Windows přes 20 let. Microsoft Edge má stejně jako prohlížeč Chrome zdrojový kód Chromium, a je dostupný pro operační systémy jako Chrome, vyjma Linuxu. [26]

#### 3.3.2.4 Safari

Prohlížeč Safari vydaný společností Apple v roce 2003 je výchozí prohlížeč pro zařízení typu iPhone, iPad, macOS a krátce byl dostupný pro systém Windows. Safari, vesměs jako každý jiný moderní prohlížeč, podporuje HTML5, umožnuje otvírání více webových stránek do karet prohlížeče a procházet webové stránky. Safari je výchozí prohlížeč v systému Mac OS, kde obsahuje funkci zabraňující ve sledování procházení mezi weby. [27]

## 3.4 Vývoj aplikací

Pro vývoj aplikací zaměřených na mnou již zmíněné platformy existuje mnoho způsobů, jak přistupovat k vývoji. Mnou zmíněné přístupy jsou metody, pomocí kterých bychom mohli vyvíjet aplikace s cílem multiplatformovosti mimo framework Uno. Mají sjednocenou kódovou základnou C# související se zkoumaným frameworkem Uno Platform.

Při vývoji nativních aplikací pro více platforem zároveň musíme pro každou z nich použít specifickou kódovou základnu a taktéž zvláštní přístup pro uživatelská rozhraní. Pro vývoj webových aplikací lze použít kódovou základnu s jazykem JavaScript a uživatelské rozhraní vytvářet prostřednictvím HTML5 a CSS3. Nativní aplikace vyvíjené pro Android mohou mít kódovou základnu vyvíjené například pomocí jazyka Kotlin a uživatelské rozhraní stylizované prostřednictvím Google Material Design. Vývoj aplikací určených pro operační systém Windows má kódovou základnu C# a uživatelské rozhraní psané v XAMLu. Pro vývoj aplikací specializujících se na systémy společnosti Apple je vhodnou kódovou základnou jazyk Swift a uživatelské rozhraní zprostředkované Swift UI.

Když chceme vyvíjet aplikace cílené na více platforem, může to být při tomto nativním vývoji aplikací složité, neboť každá platforma musí mít jak rozdílnou kódovou základnu, tak rozdílné přístupy pro uživatelské rozhraní. Vývoj takových aplikací je velice náročný jednak z hlediska potřeby počtu vývojářských týmů (pro každou platformu zvláštní tým), tak z ekonomického úhlu pohledu nebo testování a opravy chyb.

Následujícím krokem je sjednotit kódovou základnu pro všechny platformy. Pro sjednocenou kódovou základnu můžeme použít například C# a uživatelské rozhraní za pomocí některého z dostupných frameworků. Pro vývoj webových aplikací bychom použili Blazor WebAssembly, vývoj na Mac OS, iOS a Android bychom řešili pomocí frameworku Xamarin a vývoj pro Windows bychom realizovali prostřednictvím WinUI / UWP. [37]

Z těchto přístupů vyplývá, že máme Windows a platformy spojené s frameworkem Xamarin spojené s kódovou základnou C# a rozhraní tvořené za pomocí XAMLu, ale webové aplikace nikoliv. V tento moment lze použít Uno Platfrom. [37]

### 3.4.1 Xamarin

Open source platforma Xamarin je vhodná pro tvorbu novodobých aplikací pro iOS, Android, macOS a další platformy s využitím C# a .NET[6]. Xamarin umožnuje vyvíjet propracované aplikace dosahující nativního výkonu, vzhledu a chování na podporovaných platformách. Xamarin spravuje komunikaci mezi sdíleným kódem a základním kódem platforem, čímž nám umožňuje vytvářet uživatelské rozhraní pro každou platformu a napsat sdílenou logiku v jazyce C# napříč platformami.

Aplikace lze psát na počítačích s operačním systémem Windows a aplikací Visual Studio pro Windows, ale pro nasazení aplikací do zařízení od společnosti Apple je zapotřebí počítač s macOS s jeho verzí Visual Studia. [33]

### 3.4.2 Blazor WebAssembly

Blazor je open source multiplatformní rozhraní webového uživatelského rozhraní, použitelné především pro vytváření jednostránkových aplikací za pomoci .NET a C# místo jazyka JavaScript. Výkonný a flexibilní model, na kterém je Blazor založen, umožňuje vytvářet uživatelské rozhraní pomocí implementace .NET kódu v kombinací s HTML a C#.

Blazor umožnuje stavět klientské webové aplikace dvojím způsobem. První způsob je Blazor Server, ve kterém běží aplikace na logickém serveru prostřednictvím .NET Core[7]. Druhý podporovaný způsob je Blazor WebAssembly provádějící hostování komponentů na straně klienta v prohlížeči v prostředí .NET postaveného na WASM bez nutnosti dalšího pluginu a s podporou moderních prohlížečů. [34]

[6].NET - označení pro soubor technologií tvořící celou platformu

[7].NET Core - označení pro otevřený a bezplatný softwarový framework

### 3.4.3 WinUi / UWP

UWP je metoda umožňující tvořit aplikace pro zařízení se systémem Windows 10. Takovéto aplikace lze programovat v jazycích C#, C++, Visual Basic nebo JavaScript a uživatelské rozhraní za pomoci XAML, HTML, DirectX[8] nebo WinUI. [35]

WinUI je knihovna uživatelského rozhraní pro aplikace systému Windows a UWP poskytující nativní uživatelské prostředí prostřednictvím moderních ovládacích prvků a stylů pro zařízení Windows 10, mezi které patří například stolní PC s Windows systémem, Xbox, Surface Hub[9] ... WinUI mají dvě verze, a to WinUI 2.x, která je spojená s Windows 10 SDK sadou a WinUI 3.x, která je oddělená od sady Windows 10 SDK a na rozdíl od starší verze bude fungovat se všemi verzemi aplikací pro systém Windows. [36]

[8] DirectX - knihovna poskytující aplikační rozhraní pro ovládání moderního hardwaru
[9] Surface Hub - označení interaktivní tabule od společnosti Microsoft

# 4 Uno Platform

Platforma Uno je open source pod licencí Apache 2.0. Jedná se o univerzální implementaci UWP API s výstupem napříč platformami Windows, Android, iOS, macOS, WASM a od konce roku 2020 má Uno výstup i na Linux.

Vývoj může být velmi složitý proces, zejména pokud se jedná o uživatelské rozhraní pro zařízení Windows, iOS, Android a WASM. Každá z těchto platforem přistupuje k dynamickému rozvržení jinak a to může být obtížné kvůli potřebnému množství konkrétních znalostí pro každou platformu, přičemž by se musela aplikace vyvíjet vícekrát a pokaždé by to vyžadovalo nový vývojový cyklus. Uno napodobuje Windows přístup vývoje uživatelského rozhraní XAML za pomoci Uno.UI, díky čemuž se nemusíme učit přístupy pro uživatelské rozhraní každé platformy. To nám umožní sdílet styly, rozložení a datové vazby při komínování stylu XAML a nativního rozložení.


Obrázek 5: Oficiální logo Uno Platform (převzdato z [45])

Uno je univerzální platforma Windows Bridge, umožňující spouštění kódu UWP na iOS, Android a WASM. Umožňuje nám použití Windows nástrojů UWP prostřednictvím Visual Studia, a to XAML Edit and Continue[10] a C# Edit and Continue, pro vývoj aplikací, které jsou psané na Windows, ale běží na iOS, Android a WASM. Uživatelské rozhraní psané v XAML umožnuje použít stejný soubor s kódem napříč platformami. V rámci práce s Uno Platformou lze

[10]Edit and Continue - metoda umožnující upravu kódu při pozastavení aplikace

také použít vzor MVVM[11] nebo Windows Community ToolKit[12] podporovaný na všech platformách a pomocí něho lze použít funkce vazeb, stylingu, kontroly a šablony dat. [39]

Uno platforma by mohla poskytovat pro uživatele stabilní prostředí napříč platformami, pro vývojáře poskytne schopný vývoj za pomocí nástrojů Microsoft a pro designéry poskytne pixelové vzory a bohaté uživatelské rozhraní prostřednictvím XAML. [39]

## 4.1 Struktura platformy Uno

Uno platforma se v souvislosti s iOS, Android a macOS spoléhá na nativní zásobník Xamarin, který je součástí .NET 5 / .NET 6. Uno je sice podobné Xamarin.Forms, protože oba používají Xamarin „klasik“, avšak Uno Platform nevychází z Xamarin.Forms.

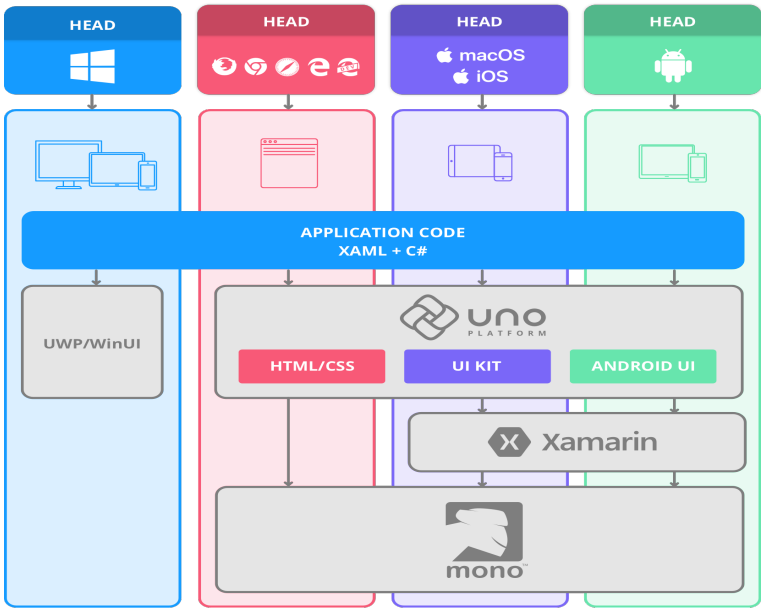
Webové aplikace jsou přetvářeny za pomocí Uno Web Boostrapper[13], který ke spouštění kódu C# používá modul Mono-WASM[14]. WASM je pro Uno platformu dobrou volbou, protože není potřeba spouštění kódu JavaScript, tím WASM umožní platformě Uno spouštět aplikace Windows v našich prohlížečích. [40]

[11] MVVM - vzor umožnující oddělení vývoje GUI od vývoje logické části

[12] Windows Community ToolKit - kolekce pomocných funkcí, vlastních ovládacích prvků a aplikačních služeb

[13] Uno Web Boostrapper - umožňuje spusti kód C# a .NET v prostředí prohlížeče

[14] Mono-WASM - technologie kombinující dva přístupy. Spouštění upravovaného kódu za běhu a statickou kompilaci do jednoho souboru, který spustí prohlížeč




Obrázek 6: Architektura Uno Platfrom(převzato z [47])

Windows aplikace jsou řešeny pomocí UWP, jak již bylo popsáno v kapitole 3.4.3, UWP. Některé prvky XAMLu používané při vytváření UWP ještě nejsou plně podporovány Uno Platformou, ale s každou nově vydanou verzí vývojáři Uno platformy přidávají podporu dalších prvků.

Uno platforma se skládá z mnoha NuGet balíčků (správce balíčků) jako například Uno.Core[15], Uno.UI[16] ... Tyto balíčky je nutné udržovat vždy na nejnovější verzi, abychom měli přístup k nejnovějším prvkům a možnostem, které nám Uno Platform může poskytnout. Na konci března 2021 vyšla nejnovější aktualizace Uno Platform 3.6 umožňující používat moderní funkce WinUI 3, jako je práce se soubory [48].

## 4.2 Uno vs. jiné frameworky

Zde se zaměříme pouze na multiplatformní frameworky pracující s C# a vynecháme ty JavaScriptové. Podíváme se na frameworky Xamarin.Forms a Avalonia UI a srovnáme odlišnosti od platformy Uno.

[15] Uno.Core - snaží se o vyplnění některých nedostatků ve stávajícím .NET

[16] Uno.UI - poskytuje sjednocené uživatelské rozhraní

Xamarin.Forms pracuje s vlastním dialektem XAML a systémem ovládacích prvků. Tento specifický dialekt XAML může být matoucí. Používání nativních ovládacích prvků Xamarin dosáhne pomocí renderovaných systémů. Funkcionality nabízející prvky v XAMLu jsou často jen průnikem toho, co nabízejí jiné platformy. Framework má velké množství knihoven a velmi silnou podporu prostřednictvím GitHubu.

Avalonia UI pracuje se svým dialektem XAML postaveným na dialektu WPF. Na rozdíl od Xamarinu používá „custom rendering“[17] tam, kde je to možné. U některých prvků chybí nativní vzhled.

Uno Platform má dialekt UWP XAML a snahu 100% kompatibility, takže když známe UWP, do velmi značné míry si vystačíme se znalostmi, které již máme. Vývojáři mohou vyvíjet UWP a plně využít nástroje poskytující skrze Visual Studio. Na pozadí použijeme nativní prvky pro danou platformu, což zajistí nativní chování. Cílem je poskytnou vzhledově jednotné uživatelské rozhraní. Uno v oblasti multiplatformosti bere v potaz i náročnou API specifickou pro platformu, jako je například práce s hardwarem, ovšem vývojář nemusí vyvíjet specifický kód, ale může použít „nativní“ Windows API jako je Geolocator, a na pozadí se pak zavolá nativní API pro cílenou platformu. [45]

## 4.3 Vývojové prostředí

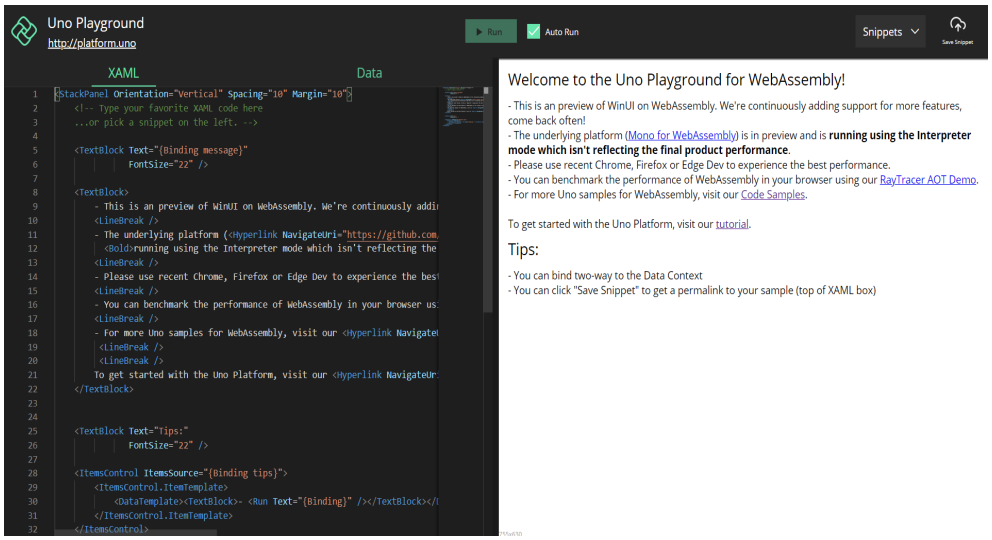
Uno platforma nám nabízí několik možností, jak si vyzkoušet nebo už psát kód a tvořit projekty.

### 4.3.1 Uno Playground

Vývojáři Uno platformy poskytují online editor prostřednictvím stránky `https://playground.platform.uno/#wasm-start`, který umožnuje vyzkoušet náhled WinUI na WASM. Editor nám umožnuje vyzkoušet

[17] Custom rendering - postup umožňující obejít nativní styly

existující ale i nové prvky XAML přímo v našem prohlížeči, uvidíme v něm i výstup našeho zkoušeného XAML kódu. V editoru máme možnost napsat i datový kontext, který můžeme navázat na náš kód XAML. Součástí Uno Playground jsou také malé příklady jednotlivých prvků XAML. [38]



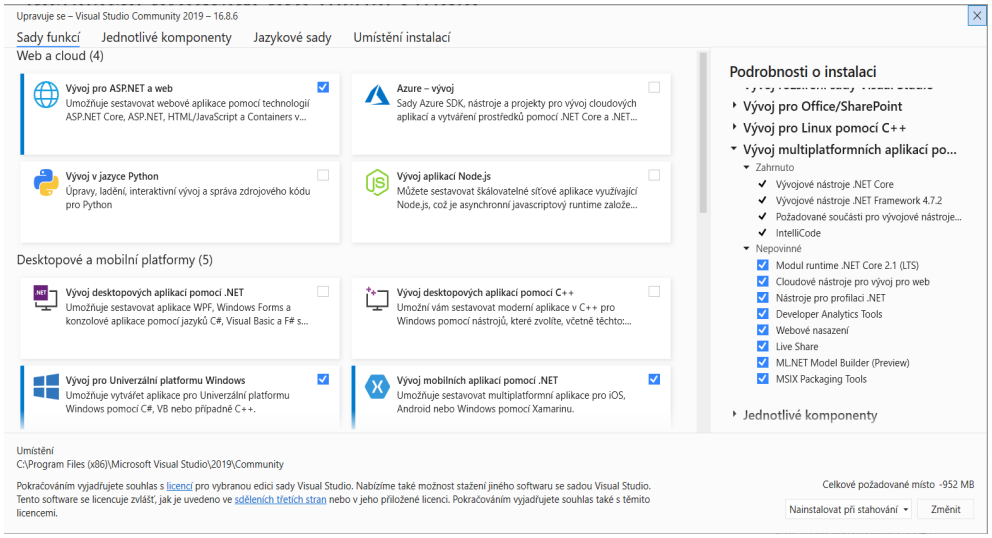

Obrázek 7: Uno Playground

### 4.3.2 Visual Studio

Další možností, jak vytvářet Uno aplikace je prostřednictvím Visual Studia. Uno podporuje Visual Studio pro Windows i macOS. Použití macOS verze nám bohužel neumožnuje vyvíjet aplikace pro systém Windows. Obdobný problém nastává s verzí Windows s aplikacemi pro iOS a macOS, ale u této verze lze tento problém eliminovat za pomocí vzdáleného přístupu k počítači s macOS, z čehož vyplývá, že jednodušší je pracovat v systému Windows.

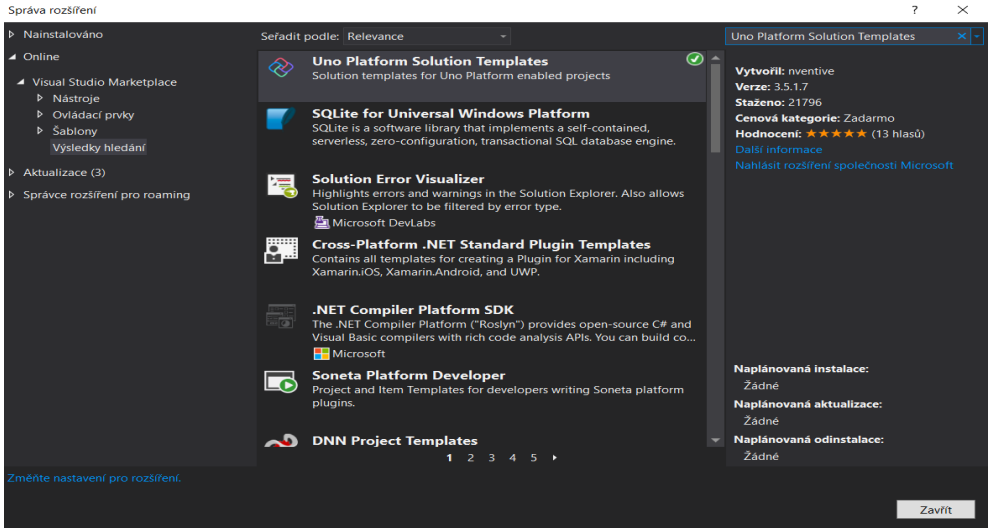
Abychom mohli vytvářet Uno aplikace ve Visual Studiu, potřebujeme verzi Visual Studio 2019 16.3 a vyšší. Pro Windows aplikace musíme mít nainstalované sady Vývoj pro Univerzální platformu Windows, Vývoj mobilních aplikací pomocí .NET pro macOS, iOS a Android s Emulátory, a nakonec Vývoj pro ASP.NET[18] a web s .NET Core 2.2 pro WASM. [41]

[18] ASP.NET - multiplatformní framework pro vytváření webových aplikací




Obrázek 8: Potřebné sady ve Visual Studiu

Ovšem abychom mohli pracovat s Uno platformou, je nutné do Visual Studia nainstalovat šablonu platformy Uno. Tu je možné stáhnout na Visual Studio Marketplace nebo si přímo ve Visual Studiu rozbalíme záložku s názvem Rozšíření a rozklikneme si správu rozšíření, ve kterém si vyhledáme Uno Platform Solution Templates a nainstalujeme. [41]




Obrázek 9: Správa Rozšíření

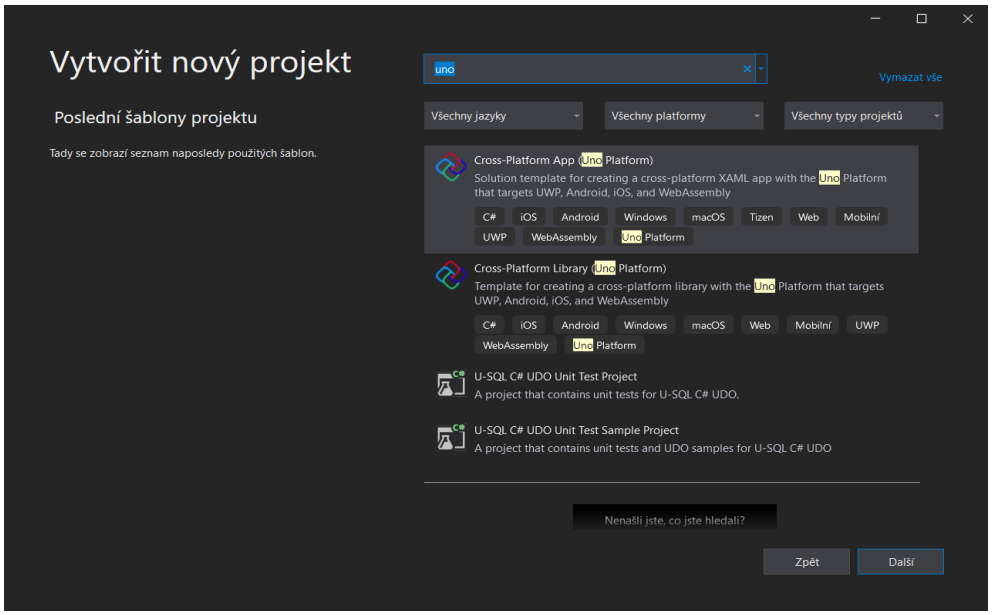
### 4.3.3 Další možnosti

Mimo Visual Studio lze také použít pracovní prostředí JetBrains Rider, do něhož je zapotřebí doinstalovat pluginy pro Xamarin, a protože ladění WASM ještě není plně podporováno, je nutné použít speciální ladicí program Chromium v prohlížeči. Také můžeme použít Visual Studio Code, který má bohužel podporu jen pro WASM. [42, 43]

# 5 Práce v Uno Platform

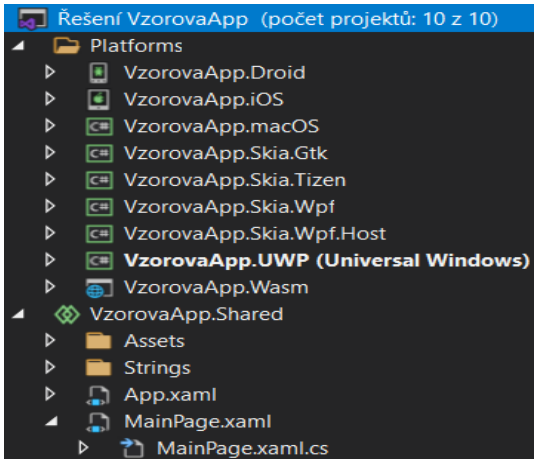
## 5.1 Založení projektu

Ve Visual Studiu zvolíme možnost založit nový projekt. Dále vyhledáme a vybereme šablonu Cross-Platform App (Uno Platform). [44]



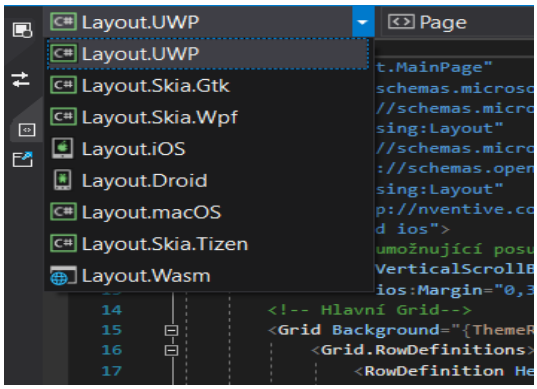

Obrázek 10: Založení Uno projektů

Založený projekt naší Uno aplikace obsahuje adresář Platforms obsahující projekt pro každou platformu a samostatný Shared projekt. Projekty jednotlivých platforem obsahují pouze nutné soubory pro funkcionalitu spuštění aplikace. [45]




Obrázek 11: Rozložení projektů

Veškerou práci provádíme ve sdíleném projektu Shared, ve kterém se nachází dva hlavní soubory XAML. Soubor App.xaml zahrnuje reference o aplikaci a jeho podsoubor App.xaml.cs obsahuje funkcionality pro zavedení dat pro platformu, na kterou sestavujeme aplikaci. Pracovní soubor MainPage.xaml spolu s jeho podsouborem MainPage.xaml.cs, jsou prvními soubory pro psaní uživatelského rozhraní a aplikační logiky. Pokud by naše aplikace neměla být jednostránková, můžeme přidat prázdnou XAML stránku, která už bude obsahovat automaticky i podsoubor pro logiku aplikace.




Obrázek 12: Nastavení zobrazení kódu pro UWP

Důležité je mít nastavené zobrazení kódu XAML své aplikace pro UWP kvůli kompilátoru, aby se nám nezobrazovali chybové hlášky chyb, které vlastně ani neexistují. Ostatní možnosti zobrazení bychom zvolili při nativním vývoji, který je velice obsáhlý, a proto jsem ho do své práce nezahrnul. Hlavní soubor MainPage.xaml obsahuje úvodní výpis „Hello Word“, tudíž můžeme sestavit a spustit aplikaci.

## 5.2 Spouštění aplikace

Spuštění a sestavení aplikace pro Windows umožnuje simulátor, vzdálený počítač nebo přímo naše zařízení s řešením x86. Při sestavování aplikace pro WASM využijeme IIS Express[19] za pomoci nainstalovaného prohlížeče (Google Chrom, Mozilla, Microsoft Edge nebo Safari) v našem počítači.

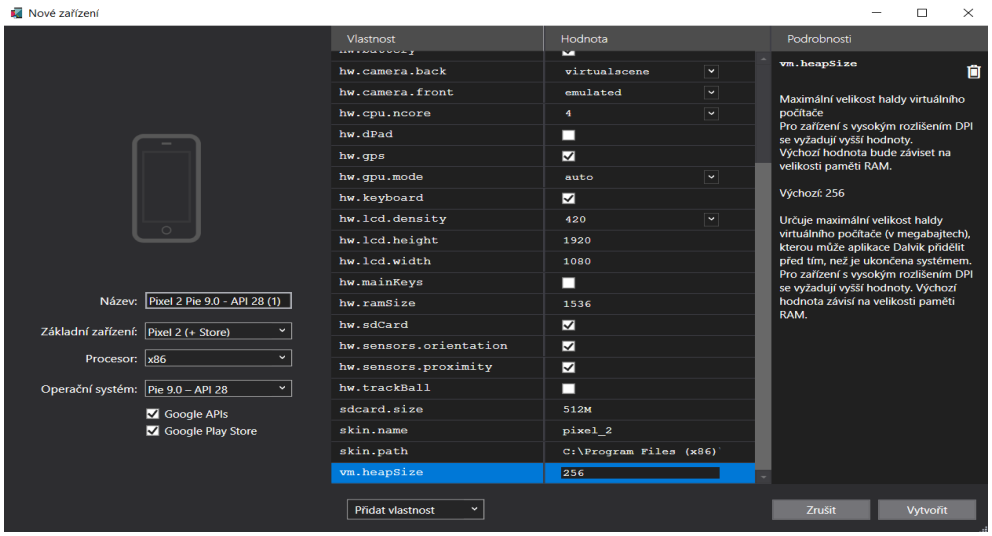
Zaměření na iOS a macOS může být problém, pokud pracujeme v systému Windows. První možností je připojit vzdálené zařízení se systémem macOS, který ovšem nemusíme mít. Druhou možností máme připojení virtuálního stroje s macOS za pomocí funkce Spárovat s počítačem Mac. Obě varianty potřebují mít v systému nainstalované Visual Studio pro Mac a korektní verzi Xcode Studia[20]. Emulátor pro otestování aplikací iOS jsem zvolil iPhone 11 Pro s iOS 14.4. Použití virtuálního počítače vyžaduje značný výkon a může zabírat velmi znatelné místo na disku našeho počítače

Pro sestavení aplikace na Android potřebujeme mít vytvořený Google emulátor.

Emulátor vytvoříme za pomocí funkce Správa zařízení s Androidem. Pro emulátor zvolíme základní zařízení, procesor a operační systém (verzi Androidu). Pro rychlost emulátoru můžeme upravit velikost RAM paměti, jak nám náš počítač dovolí a zbytek nastavit dle našeho uvážení. Pro práci a testování jsem zvolil Android 9.0 s API 28.

[19]ISS Express - je softwarový webový server pro hostování webových aplikací

[20]Xcode Studio - je vývojové prostředí od společnosti Apple




Obrázek 13: Nastavení emulátoru Android

## 5.3 Tvorba

Uno je univerzální implementace UWP API, jak již bylo zmíněno v kapitole 4, tudíž kód píšeme za pomocí XAML a C#. Většina ovládacích prvků se implementuje stejně jako v UWP, tudíž nebudu uvádět všechny ovládací prvky, ale prostřednictvím následujících mnou vybraných ovládacích prvků, které předvedu v několika malých triviálních příkladech, nastíním tvorbu v platformě Uno. Vzhledem k tomu, že Uno Platfrom staví na základech UWP, tak je vhodné přistupovat k prvkům souvisejícím s uživatelským rozhraním pomocí XAML a procedurální kód používat co nejméně a tím vytvářet „zdravý“ kód.

### 5.3.1 Schéma XAML

XAML je odvezené od XML, tedy musíme postupovat podle schématu XML. Schéma je jakási dohoda mezi producentem a spotřebitelem XML. Námi vytvořený projekt obsahuje soubor MainPage.xaml. Tento soubor má již před vygenerované jmenné prostory. [46]

```
<Page
    x:Class="VzorovaApp.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:VzorovaApp"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">
</Page>
```

Příklad 1: Schéma XAML

Pokud bychom chtěli přidat jmenný prostor pro na mapování zdroje dat pro naše uživatelské rozhraní, například pro prvek DataTemplate, můžeme narazit na problém, že náš zdroj dat není vidět nebo neexistuje v oboru názvů. Tuto chybu lze vyřešit zavřením všech souboru v projektu, poté zavřít i samotný projekt a vymazáním skryté složky .vs v souborech našeho projektu.

### 5.3.2 Ovládací prvky

Mnou vybrané ovládací prvky jsem rozdělil do skupin a pro každou z nich vytvořím jeden triviální příklad, na kterém budou prvky demonstrovány.

První skupina „Prvky pro layout“ obsahuje ovládací prvky jako GridView, Grid, StackPanel, ScrollViewer, Border a RelativePanel. Kvůli přehlednosti zobrazení v Gridu a StackPanelu použiji další prvky jako Rectangle a TextBlock. Druhá skupina „Základní vstupy“ bude obsahovat prvky typu Button, CheckBox, ComboBox, RadioButton, Slider, ToggleSwitch, ProgressRing, ProgressBar a Image. Třetí skupina „Navigace“ bude obsahovat prvek NavigationView, který známe z UWP. Jedná se o základní navigaci, kterou můžeme znát ze systému Windows nebo emailu Outlook. Čtvrtá skupina „Animace“ bude obsahovat pár příkladů animací. Kompletní pohledy jednotlivých příkladů pro každou platformu uvedu do příloh práce.

Jak jsem již zmínil, Uno Platform se zakládá UWP, a v důsledku toho je ovládacích prvků opravdu mnoho. Mezi základní ovládací prvky můžeme zařadit například:

- TextBox,
- Button,
- RadioButton,
- CheckBox,
- ComboBox,
- Slider,
- ToggleSwitch,
- ProgressBar,
- GridView,
- ListView,
- Image,
- NavigationView.

Výše uvedené ovládací prvky mají společné a specifické vlastnosti. Mezi společné vlastnosti můžeme zařadit například:

- Name – jméno vkládacího prvku,
- Width – nastavení šířky,
- Height – nastavení výšky,
- FontSize – velikost textu ovládacího prvku,
- IsEnabled – povolí nebo zakáže interakci s prvkem,

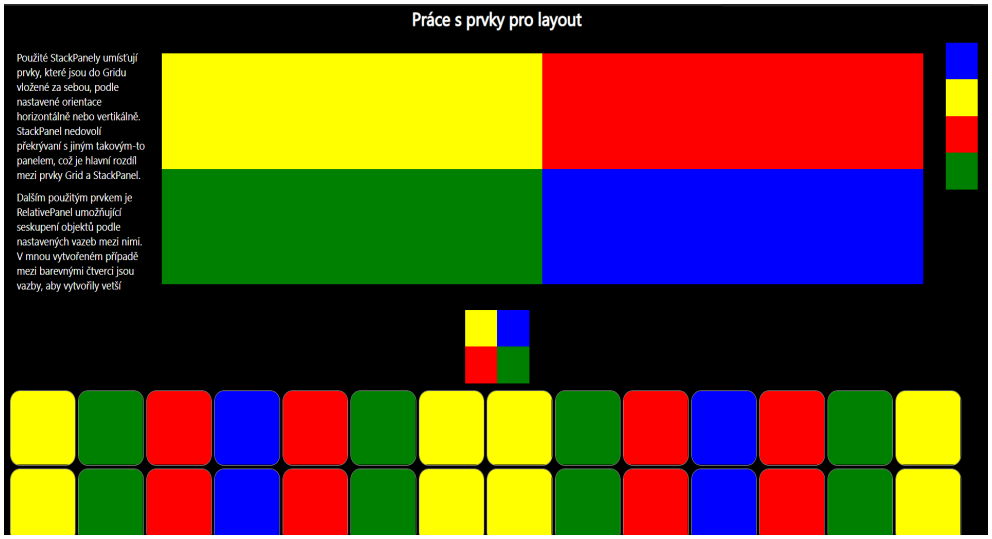
- Margin – nastaví vlastnosti okraje prvku,
- VerticalAlignment – vertikální zarovnání,
- HorizontalAlignment – horizontální zarovnání,
- Visibility – nastaví viditelnost prvku,
- Opacity – nastaví neprůhlednost prvku,
- MaxWidth / MaxHeight – nastavení maximální výšky nebo šířky,
- MinWidth / MinHeight – nastavení minimální výšky nebo šířky,
- Padding – definuje prostor mezi vnitřním ohraničením prvku,
- atd.

Pro získání bližších informací Vám doporučuji navštívit Microsoft UWP dokumentaci dostupnou z adresy `https://docs.microsoft.com/en-us/uwp/api/windows.ui.xaml.controls?view=winrt-19041`.

První skupina „Prvky pro layout“ obsahuje ovládací prvky jako GridView, Grid, StackPanel, ScrollViewer, Border a RelativePanel. Kvůli přehlednosti zobrazení v Gridu a StackPanelu použiji další prvky jako Rectangle a TextBlock. Druhá skupina „Základní vstupy“ bude obsahovat prvky typu Button, CheckBox, ComboBox, RadioButton, Slider, ToggleSwitch, ProgressRing, ProgressBar a Image. Třetí skupina „Navigace“ bude obsahovat prvek NavigationView, který známe z UWP. Jedná se o základní navigaci, kterou můžeme znát ze systému Windows nebo emailu Outlook. Čtvrtá skupina „Animace“ bude obsahovat pár příkladů animací. Kompletní pohledy jednotlivých příkladů pro každou platformu uvedu do příloh práce.

#### 5.3.2.1 Prvky pro layout

Abychom mohli vhodně umístit jednotlivé ovládací prvky v naší aplikaci, potřebujeme se seznámit s mřížkami a panely, do kterých budeme usazovat jednotlivé prvky.




Obrázek 14: Ukázka příkladu Prvky pro layout v prohlížeči Google Chrome

Šablonu aplikace tvoří ScrollViewer a v něm vnořený Grid o čtyřech řádcích. Tento vnořený Grid obsahuje TextBlock, další Grid, RelativePanel a GridView. Celý příklad jsem zaobalil do prvku ScrollViewer, abychom měli možnost posouvání obsahu vygenerovaného v prvku GridView, které by například při zobrazení na mobilním zařízení nebyly vidět.

```
<ScrollViewer VerticalScrollBarVisibility="Visible"
              ios:Margin="0,30">
    <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>
        <TextBlock Text="Prace s Grid a StackPanel"
                   FontSize="25"
                   Height="50"
                   FontWeight="Bold"
                   HorizontalAlignment="Center"
                   VerticalAlignment="Center"/>
        <Grid Margin="10,0,0,20"
              Grid.Row="1">
            ...
        </Grid>
        <RelativePanel Grid.Row="2"
                       HorizontalAlignment="Center">
            ...
        </RelativePanel>
        <GridView ItemsSource="{x:Bind ListData}"
                  Margin="10"
                  Grid.Row="3">
            ...
        </GridView>
    </Grid>
</ScrollViewer>
```

Příklad 2: Schéma příkladu Prvky pro layout

V tomto konkrétním příkladu použitý vnořený Grid s barevnými čtverci nám ukazuje, že budeme-li chtít Rectangle umístit na první pozici sloupce i řádku, nemusíme tyto parametry definovat, protože jsou při implementaci prvku automaticky nastaveny na nulu, tedy první sloupec a řádek. Pro nastavení velikosti řádků a sloupců můžeme použít jako parametry „Auto“, které nám přizpůsobí velikost automaticky podle vloženého prvku nebo můžeme použít „*“, což vyplní zbytek dostupného místa. Popřípadě můžeme nastavit pevnou velikost v pixelech.

```
<Grid Margin="10,0,0,20"
      Grid.Row="1">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="Auto"/>
        <ColumnDefinition Width="*"/>
        <ColumnDefinition Width="Auto"/>
    </Grid.ColumnDefinitions>
    <StackPanel  Orientation="Vertical">
        <TextBlock TextWrapping="Wrap"
                   Margin="10"
                   Width="200">
            ...
        </TextBlock>
    </StackPanel>
    <Grid Grid.Column="1"
          Margin="15">
        <Grid.RowDefinitions>
            <RowDefinition Height="*"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>
        <Rectangle Fill="Yellow" />
        <Rectangle Fill="Red" Grid.Column="1"/>
        <Rectangle Fill="Green" Grid.Row="1"/>
        <Rectangle Fill="Blue" Grid.Row="1" Grid.Column="1"/>
    </Grid>
    <StackPanel Orientation="Vertical"
                Margin="20,0,20,0"
                Grid.Column="2" >
        <Rectangle Fill="Blue" Height="50" Width="50"/>
        <Rectangle Fill="Yellow" Height="50" Width="50"/>
        <Rectangle Fill="Red" Height="50" Width="50"/>
        <Rectangle Fill="Green" Height="50" Width="50"/>
    </StackPanel>
</Grid>
```

Příklad 3: Vnořený Grid

Použité StackPanely umísťují prvky, které jsou do Gridu vložené za sebou, podle nastavené orientace horizontálně nebo vertikálně. StackPanel nedovolí

překrývání s jiným takovýmto panelem, což je hlavní rozdíl mezi prvky Grid a StackPanel.

Dalším použitým prvkem je RelativePanel umožňující seskupení objektů podle nastavených vazeb mezi nimi. V mnou vytvořeném případě mezi barevnými čtverci jsou vazby, aby vytvořily větší čtverec.

```
<RelativePanel Grid.Row="2"
               HorizontalAlignment="Center">
    <Rectangle x:Name="blue"
               Fill="Blue"
               RelativePanel.RightOf="yellow"
               MinHeight="50"
               MinWidth="50"/>
    <Rectangle x:Name="yellow"
               Fill="Yellow"
               MinHeight="50"
               MinWidth="50"/>
    <Rectangle x:Name="red"
               Fill="Red"
               RelativePanel.Below="yellow"
               MinWidth="50"
               MinHeight="50"/>
    <Rectangle x:Name="green"
               Fill="Green"
               RelativePanel.RightOf="red"
               RelativePanel.Below="blue"
               MinHeight="50"
               MinWidth="50"/>
</RelativePanel>
```

Příklad 4: RelativePanel

Na konec příkladu jsem použil GridView umožňující řazení vytvářených prvků v tomto případě barevných čtverců do řádků a sloupců. Tyto čtverce jsou načítány do prvku DataTemplate za pomocí dat z vytvořeného listu v C# kódu a jsou vnořeny do prvku Border, pomocí kterého mají čtverce zvýrazněný okraj a zakulacené rohy. GridView sám přizpůsobuje počet námi vygenerovaných čtverců v jedné řadě podle šířky okna aplikace. Namísto prvku GridView lze využít i prvek ListView. Rozdíl mezi těmito prvky je v zobrazení. ListView řadí prvky do jednoho sloupce, ale prvek GridView je řadí do řádků.

```
<Page
   ...
      xmlns:data1="using:Layout">
   ...
    <GridView ItemsSource="{x:Bind ListData}"
            Margin="10"
            Grid.Row="2">
        <GridView.ItemTemplate>
            <DataTemplate x:DataType="data1:Data">
                <Border CornerRadius="15"
                        BorderBrush="DarkGray"
                        BorderThickness="1">
                    <Rectangle Fill="{x:Bind Barva}"
                               Width="100"
                               Height="100"/>
                </Border>
            </DataTemplate>
        </GridView.ItemTemplate>
    </GridView>
    ...
</Page>
```

Příklad 5: GridView

Zmíněný DataTemplate přebírá data z jmenného prostoru, který jsme definovali. Jmenný prostor nám určuje zdroj dat přiřazený do tohoto prvku, což v našem případě je vytvořená třída Data. Tato třída obsahuje dvě proměnné a metodu vracející list s daty.

```
namespace Layout
{
    public class Data
    {
        // deklarace dvou promennych Id a Barvu pro ctverce
        public int Id { get; set; }
        public string Barva { get; set; }
    }
    // metoda na vraceni listu s daty
    public class DataManager
    {
        public static List<Data> GetData()
        {
            var data = new List<Data>
            {
                new Data { Id = 1, Barva = "Yellow"},
                new Data { Id = 2, Barva = "Green" },
                new Data { Id = 3, Barva = "Red" },
                new Data { Id = 4, Barva = "Blue" },
                ...
                new Data { Id = 30, Barva = "Green" }
            };
            return data;
        }
    }
}
```

Příklad 6: Třída Data

Ve třídě Data jsme si připravili další třídu DataManager obsahující metodou GetData() vracející seznam čtverců. Nyní musíme v souboru MainPage.xaml.cs, definovat list a poté nad ním zavolat naši metodu GetData.

```
namespace Layout
{
    public sealed partial class MainPage : Page
    {
        // deklarace listu
        public List<Data> ListData;

        public MainPage()
        {
            //naplneni listu za pomoci metody GetData() z tridy DataManager
            ListData = DataManager.GetData();
            this.InitializeComponent();
        }
    }
}
```

Příklad 7: Definice dat z třídy Data

V neposlední řadě musíme vyřešit problém se zobrazením na iOS kvůli výřezu displeje. Přidáme jmenný prostor pro iOS a poté do ScrollVieweru přidáme Margin cílený pro iOS. Nesmíme zapomenout přidat iOS i do seznamu ignorovaných položek.

```
<Page
     ...
    xmlns:ios="http://nventive.com/ios"
    mc:Ignorable="d ios">
    ...
<\Page>
```

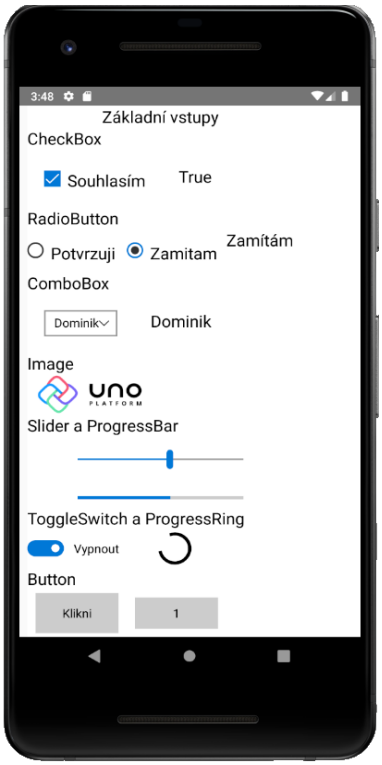
Příklad 8: Jmený prostor pro iOS

#### 5.3.2.2 Základní vstupy

Mezi základní ovládací prvky by mohl být zahrnut větší počet ovládacích prvků, avšak v rámci této práce myslím, že stačí předvést pár následujících prvků, které by mohly být prioritně používány nejčastěji.

První prvek, který jsem použil, je CheckBox. Tento prvek podle „zaškrtnutí“ přenáší svoji hodnotu do TextBoxu. CheckBox má základní vlastnosti jako

název, velikost či zarovnání. Na rozdíl od tlačítka, které má vlastnost Click, CheckBox pracuje s vlastností Tapped.




Obrázek 15: Ukázka příkladu Základní vstupy na systému Android

```
...
<CheckBox x:Name="checkBox"
          Content="Souhlasim"
          FontSize="20"
          Tapped="ClickCheck"
          Margin="20"/>
    <TextBlock Text="Vysledek"
               x:Name="vysledek"
               FontSize="20"
               Margin="20"/>
    ...
public void ClickCheck(object sender, TappedRoutedEventArgs e)
    {
        vysledek.Text = checkBox.IsChecked.ToString();
    }
```

Příklad 9: Prvek CheckBox

Stejné atributy demonstruje i RadioButton, avšak tento prvek pracuje se specifickou vlastností Checked.

```
...
 <RadioButton Content="Potvrzuji"
              x:Name="Po"
              GroupName="radio"
              Checked="Checke"
              FontSize="20"
              Margin="0,10,0,10"/>
    <RadioButton Content="Zamitam"
                 x:Name="Za"
                 GroupName="radio"
                 Checked="Checke"
                 FontSize="20"/>
        <TextBlock x:Name="radioVysledek"
                   .../>
    ...
public void Checke(object sender, RoutedEventArgs e)
    {
        radioVysledek.Text = (bool)Po.IsChecked ? "Potvrzuji" : "Zamitam";
    }
```

Příklad 10: Prvek RadioButton a jeho metoda

Mezi další použité prvky patří Image a ComboBox. Image má jako ostatní prvky základní vlastnosti plus vlastnost Source. Do této vlastnosti přiřazujeme hodnotu zdroje obrázku, což může být složka Assets. Také nesmíme zapomenout na vlastnost Stretch určující, jak se obrázek zobrazí. ComboBox má nastavenou vlastnost SelectionChanged, ke které je přiřazená metoda měnící hodnotu popisu TextBoxu podle vybraného prvku.

```
<Image Source="/Assets/Uno.png"
       Stretch="Fill"
       Width="150"
       Height="50"/>
     ...
<ComboBox x:Name="comboB"
          SelectionChanged="ComboCheck"
          Margin="20">
    <ComboBoxItem Content="Jirka"/>
    <ComboBoxItem Content="Dominik"/>
    <ComboBoxItem Content="Josef"/>
</ComboBox>
    <TextBlock x:Name="textCombo"
               ... />
   ...
public void ComboCheck(object sender, SelectionChangedEventArgs e)
    {
        if (textCombo == null) return;
        var comboBox = (ComboBox)sender;
        var prvek = (ComboBoxItem)comboBox.SelectedItem;
        textCombo.Text = prvek.Content.ToString();
    }
```

Příklad 11: Ukázka Image a ComboBox

Prvky Slider a ProgressBar jsem úmyslně předvedl v jedné ukázce tím, že data ze Slideru jsou ihned po změně hodnoty přenášena do ProgressBaru prostřednictvím funkce x:Bind. U obou těchto prvků je vhodné nastavit minimální a maximální rozsah hodnot například 0–100 a velikost prvků jako takových.

```
<Slider Name="slider"
        Minimum="0"
        Maximum="100"
        Width="200"
        Margin="10"/>
<ProgressBar Value="{x:Bind slider.Value, Mode=OneWay}"
             Maximum="100"
             Width="200"
             Margin="10"/>
```

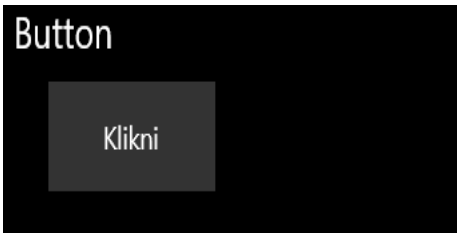
Příklad 12: Ukázka prvků Slider a ProgressBar

Abych na příkladu ukázal také generování prvku pomocí procedurálního kódu, zvolil jsem pro to prvek tlačítko. Po kliknutí na tlačítko se vytvoří další, které se přidá jako „potomek“ StackPanelu, do kterého tlačítka náleží.

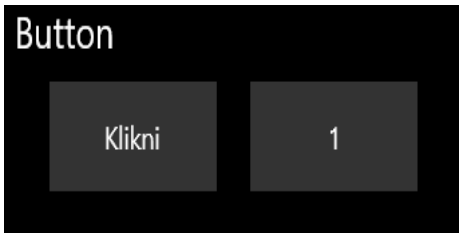
```
<StackPanel ...>
    <Button x:Name="btn"
            Content="Klikni"
            MinHeight="50"
            Click="Click_btn"
            MinWidth="100"
            Margin="10"/>
</StackPanel>
```

Příklad 13: Ukázka prvku Button

Metoda vytváří nový objekt typu Button. Dále přiřazuje našemu vytvořenému tlačítku parametry jako velikost nebo obsah a poté se přidá jako potomek do našeho StackPanelu.




Obrázek 16: Před přidáním nového tlačítka




Obrázek 17: Po přidání nového tlačítka

```
public void Click_btn(object sender, RoutedEventArgs e)
    {
        ...
         Button btn = new Button();
         btn.Width = 100;
         btn.Height = 50;
         btn.Margin = new Thickness(10);
         btn.Content = i;
         i++;
         btnStack.Children.Add(btn);
    ...
    }
```

Příklad 14: Ukázka metody prvku Button

Jako poslední dva prvky jsem zvolil ToggleSwitch a ProgressRing. ToggleSwitch v tomto příkladu zapíná a vypíná zmíněný ProgressRing, který známe z operačního systému Windows. Tento prvek pracuje s vlastnostmi OffContent a OnContent, které určují, co bude zobrazeno při zapnutí či vypnutí tohoto prvku. Poslední funkci, kterou jsem přiřadil, je Toggled, což je podobná vlastnost jako jsem popisoval u CheckBoxu.

```
<ToggleSwitch OffContent="Zapnout"
              OnContent="Vypnout"
              Toggled="ToggleSwitch"/>
<ProgressRing x:Name="progRing" Visibility="Collapsed"/>
```

Příklad 15: Ukázka ToggleSwitche a ProgressRing

```
public void ToggleSwitch(object sender, RoutedEventArgs e)
    {
        ToggleSwitch toggleSwitch = sender as ToggleSwitch;
        if (toggleSwitch != null)
            {
                if (toggleSwitch.IsOn == true)
                    {
                        progRing.IsActive = true;
                        progRing.Visibility = Visibility.Visible;
                    }
                else{
                        progRing.IsActive = false;
                        progRing.Visibility = Visibility.Collapsed;
                    }
            }
    }
```

Příklad 16: Ukázka metody ToggleSwitch

#### 5.3.2.3 Navigace

Pro ukázku navigace jsem zvolil prvek NavigationView. Tento prvek vytvoří vysouvací menu s „nadpisy“ pro přepínání stránek, které jsou směrovaný pro zobrazení do prvku Frame. Náš prvek pro navigaci obsahuje vlastnost SelectionChanged přijímající metodu NavigaceZmena(). Dalšími nastavenými vlastnostmi jsou titulek menu nabídky PaneTitle anebo Header, což je nadpis

nad prvkem Frame měnící se při načtení jiné stránky. Každý prvek v menu nabídce může obsahovat ikonu, popis a vlastnost Tag, pomocí kterého přistupujeme v naší navigaci k jednotlivým stránkám.

```
<NavigationView Header="Hlavni stranka"
                x:Name="nav"
                SelectionChanged="NavigaceZmena"
                IsBackButtonVisible="Collapsed"
                PaneTitle="Navigace">
    <!-- Jednotlive "talcitka" jako odkazy na nami vybranou stranku -->
    <NavigationView.MenuItems>
        <!-- naspis sekce -->
        <NavigationViewItemHeader Content="Sekce jedna"/>
        <NavigationViewItem Icon="Contact2"
                            Content="Prvni strana"
                            Tag="prvni"
                            IsSelected="True"/>
        <!-- oddelujicicara mezi sekcemi-->
        <NavigationViewItemSeparator/>
        <!-- nadpis sekce-->
        <NavigationViewItemHeader Content="Sekce dva"/>
        <NavigationViewItem Icon="Camera"
                            Content="Druha strana"
                            Tag="druha"/>
        <NavigationViewItem Icon="Pictures"
                            Content="Treti strana"
                            Tag="treti"/>
        </NavigationView.MenuItems>
    <!-- Do prvku frame jsou nacitany ostatni stranky-->
    <Frame x:Name="contentFrame"
           Margin="15,0,15,0"/>
</NavigationView>
```

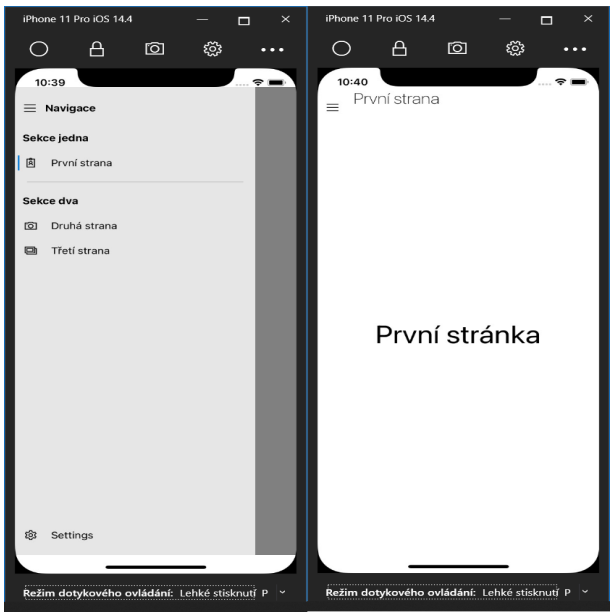
Příklad 17: XAML Kód UI navigace

Logická část ukázky navigace se týká metody pro ovládání navigace v celé aplikaci a to NavigaceZmena(). Tato metoda obsahuje podmínku, zda vybraná položka v menu je nastavení. Pokud tomu tak není, vybírá se možnost pomocí switche, ve kterém se rozhoduje, podle již zmiňovaného prvku Tag, co bude načteno do prvku Frame.

```
1  private void NavigaceZmena(NavigationView sender,
2      NavigationViewSelectionChangedEventArgs args)
3  {
4      if (args.IsSettingsSelected)
5          {
6              nav.Header = "Nastaveni";
7              contentFrame.Navigate(typeof(Nastaveni));
8          }
9          else
10         {
11             NavigationViewItem item = args.SelectedItem as NavigationViewItem;
12             switch (item.Tag.ToString())
13                 {
14                     case "prvni":
15                         nav.Header = "Prvni strana";
16                         contentFrame.Content = new Prvni();
17                         break;
18                     case "druha":
19                         contentFrame.Content = new Druha();
20                         nav.Header = "Druha strana";
21                         break;
22                     case "treti":
23                         nav.Header = "Treti strana";
24                         contentFrame.Content = new Treti();
25                         break;
26                 }
27         }
28 }
```

Příklad 18: Kód metody NavigaceZmena

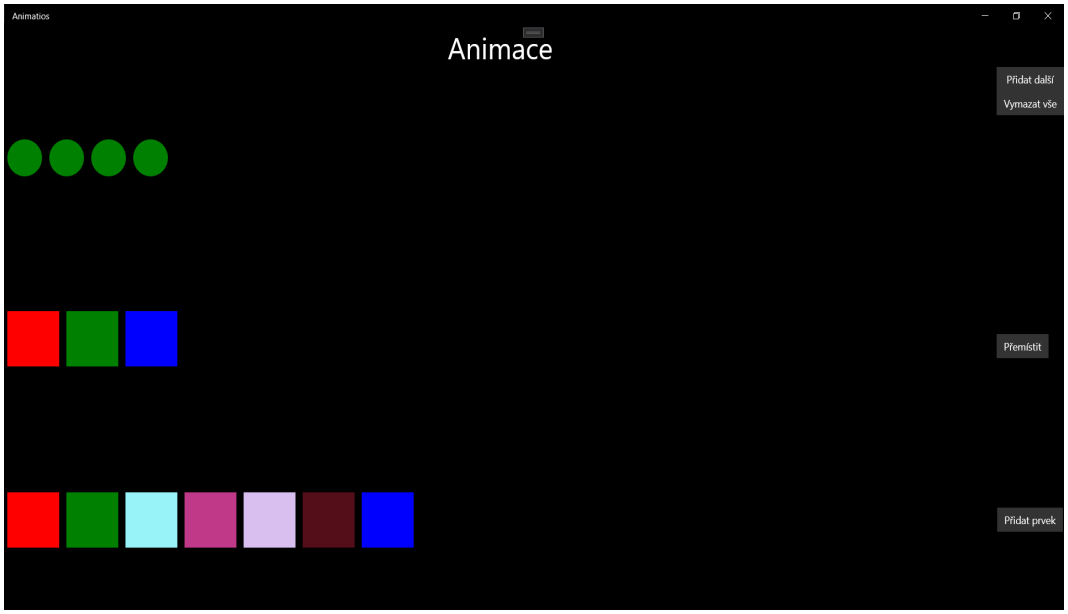
Pro ukázku výsledného zobrazení v tomto příkladu navigace jsem zvolil vzhled z iOS.




Obrázek 18: Ukázka příkladu Navigace na systému iOS

#### 5.3.2.4 Animace

Příklad animací jsem chtěl demonstrovat jako posun různých objektů podle os, avšak tyto animace mi fungovaly pouze na platformě Windows. Proto jsem byl nucen zvolit snazší pohyby objektů, jako jsou přechody. Mnou použité animace jsou inspirovány animacemi, které byly použité v ukázkové aplikaci od společnosti Microsoft, a to XAML Controls Gallery stažitelné jako kód z adresy `https://github.com/microsoft/Xaml-Controls-Gallery/tree/master/`.




Obrázek 19: Ukázka příkladu Animace na systému Windows

První příklad obsahuje animaci EntranceThemeTransition projevující se při prvním zobrazení nového objektu malým přechodem. Zajímavé na této animaci je, že na Windows se přechod projevil zespodu nahoru a na mobilních zařízeních zprava doleva. Tato animace funguje na Windows, Android a iOS, ale na WASM se mi její zobrazení nepodařilo.

```
<StackPanel ...>
    <StackPanel.ChildrenTransitions>
        <TransitionCollection>
            <EntranceThemeTransition IsStaggeringEnabled="True" />
        </TransitionCollection>
    </StackPanel.ChildrenTransitions>
    <Ellipse Width="50" Height="50" Margin="5" Fill="Green" />
</StackPanel>
```

Příklad 19: Ukázka zápisu EntranceThemeTransition

Druhý příklad je postaven na animaci RepositionThemeTransition reagující na pohyb objektů. Jeden z objektů, v tomto případě prostřední čtverec, schováme a zobrazujeme po kliknutí na tlačítko. Tato animace je jediná, která funguje na všech mnou testovaných platformách.

```
<Rectangle ... >
    <Rectangle.Transitions>
        <TransitionCollection>
            <RepositionThemeTransition />
        </TransitionCollection>
    </Rectangle.Transitions>
</Rectangle>
```

Příklad 20: Ukázka zápisu RepositionThemeTransition

Třetí příklad pracuje s animací ReorderThemeTransition, reaguje na změnu pořadí při přidání nového objektu. Tato animace viditelně funguje pouze na platformě Windows, na ostatních platformách se klasicky přidá nový objekt, ale bez přehrání animace přechodu.

```
<StackPanel Orientation="Horizontal">
    <StackPanel.ChildrenTransitions>
        <TransitionCollection>
            <ReorderThemeTransition/>
        </TransitionCollection>
    </StackPanel.ChildrenTransitions>
</StackPanel>
```

Příklad 21: Ukázka zápisu ReorderThemeTransition

Abych to shrnul, možnost tvorby animací v platformě Uno není zrovna nejlepší. Naštěstí Uno Platform pracuje na NuGet Balíčku Uno Platform Lottie, který by měl eliminovat nedostatky v animacích a pravděpodobně bude součástí šablony Uno Platform.

## 5.4 Rozdíly v zobrazení

Všechny ukázkové příklady jsem testoval za pomocí Android a iOS Emulátoru, Windows 10 a mnou vybraných webových prohlížečů jako jsou Google Chrom, Firefox Mozilla, Microsoft Edge. Při testování jsem zatím nenarazil na odlišnosti v zobrazení mezi jednotlivými prohlížeči, což dle mého názoru je oproti jiným technologiím velké plus. Obrázky jednotlivých pohledů příkladů ke každé platformě budou uloženy do příloh.

Během práce v Uno Platform jsem narazil na pár rozdílů v zobrazování. První rozdíl je zobrazení fontu textu. Ačkoli se nejedná o tak závažný prvek jako dát do aplikace specifický font, je velice obtížné najít jiný než základní font textu, který by se zobrazoval na všech platformách stejně, a ještě s českými znaky.

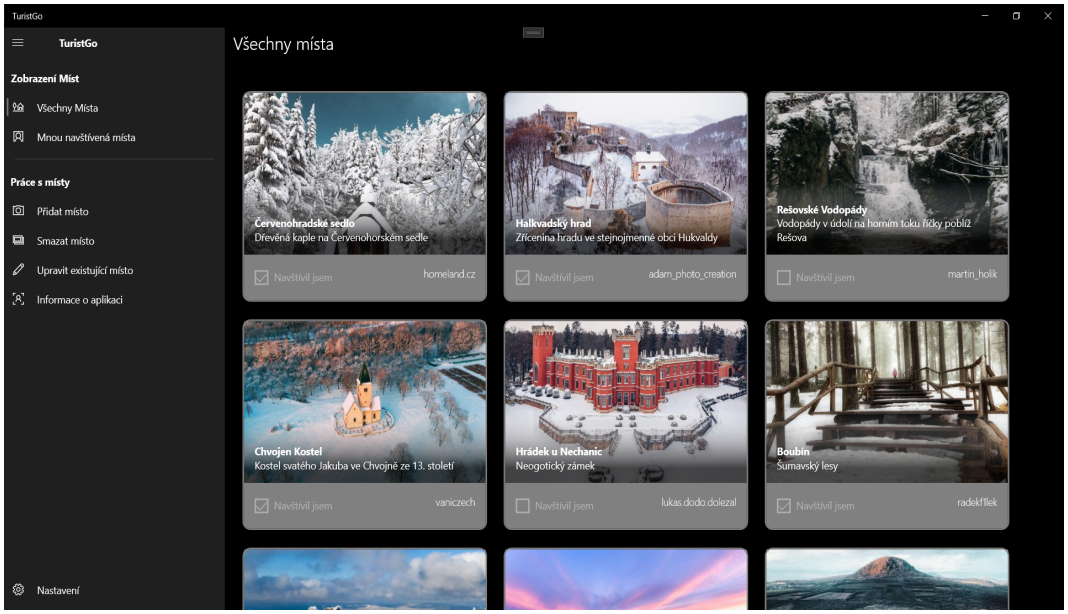
Další rozdíl v zobrazování nastane, když spustíme aplikaci na platformě iOS. Tato platforma mívá často výřez displeje a tomu se uživatelské rozhraní psané v platformě Uno nepřizpůsobuje. Tento problém však lze vyřešit přidáním specifického jmenného prostoru pro iOS a nastavením „marginu“ pro odsazení, když spouštíme aplikaci pro iOS, jak je ukázáno na příkladu v kapitole 5.3.2.1.

Další problém se zobrazováním je v animacích. Uno Platform pravděpodobně zatím plně nepodporuje animace, které zvládá UWP. Po tomto selhání Uno jsem se rozhodl zkusit animace podporované i WPF, avšak ani ty neprošly přes kompilátor mimo sestavení aplikace pro platformu Windows. Poté jsem se rozhodl zkusit animace, který by podle Uno dokumentace měly fungovat, a to přechody. Tyto přechody už sice prošly přes kompilátor, avšak na některých platformách se i přesto nezobrazovaly.

Abychom ale nebyli jen kritičtí, tak za pomoci XAML kódu můžeme přidat uživatelskému rozhraní motiv podle nastavení v zařízení, na kterém aplikaci spouštíme, což je za mě velká výhoda. Tento XAML kód je předveden na příkladu 2 v kapitole 5.3.2.1 pomocí nastavení specifické vlastnosti Background pro prvek Grid. Tím se dostávám k prvku Grid a jeho velké výhodě, který je nejvhodnější volbou pro použití našeho layoutu, abychom měli „responzivní“ uživatelského rozhraní před prvkem StackPanel, který se nepřizpůsobuje velikosti displeje.

# 6 Aplikace

Pro upřesnění tvorby aplikací v platformě Uno je součástí bakalářské práce souvislá aplikace. V tomto příkladu jsem se zaměřil na aplikaci s tématikou zajímavých míst a památek pojmenovanou TuristGo. Aplikaci je možné procházet za pomoci menu nabídky s odkazy na jednotlivé stránky aplikace. Příklad nám umožňuje prohlédnout si všechny místa v aplikaci, pouze mnou navštívená místa a základní operace s místy jako jsou přidat nové místo, odebrat zvolené místo nebo upravit námi vybrané místo. Obrázky demonstrující zobrazení mezi platformami jedné mnou vybrané menu nabídky uvedu do přílohy. Obrázky míst či památek, které jsem použil ve své aplikaci jako vzorová data jsem převzal z Instagram účtu vizitcz[21].




Obrázek 20: Ukázka vzhledu aplikace TuristGo na systému Windows

## 6.1 Komplikace

Během tvorby aplikace jsem narazil na řadu komplikací. První velká komplikace nastala v momentě, kdy jsem chtěl vytvořit metodu umožňující

[21] Vizitcz dostupné z adresy `https://www.instagram.com/visitcz/?hl=cs`

výběr souboru ze zařízení, na kterém je aplikace spuštěná. Tento přístup k zařízení se zkomplikoval v momentě, kdy jsem aplikaci spustil na jiném zařízení než na Windows. V momentě zjištění tohoto problému bylo pouze jedno řešení, a to nativní přístup k jednotlivým platformám. Naštěstí zrovna v tuto dobu vyšla dlouho očekávaná aktualizace s označením Uno Platform 3.6 umožňující použít mnoho přístupu z WinUI 3 na všechny platformy včetně přístupu k souborům zařízení.

Následující problém považuji za nejzávažnější, na které jsem narazil v rámci mé práce s platformou Uno. Zprvu jsem chtěl ukládat data do databáze, za využití možnosti vytvořit lokální databázi, za pomoci dvou NuGet balíčků, které jsem doinstaloval do projektu s aplikací, a to sqlite-net-pcl a pro web bylo zapotřebí Uno.SQLitePCLRaw.provider.wasm. Do mého fyzického počítače bylo třeba doinstalovat rozšíření WSL 2. Zpočátku se zdálo, že vše bude fungovat. Také fungovalo do chvíle, než jsem narazil na problém, kdy se nedokázala zobrazit data převzatá z databáze do vygenerovaných objektů, kam se načítaly popisky a obrázek daného místa na všech platformách mimo Windows, na které vše fungovalo dle očekávání. Tento problém jsem nedokázal vyřešit ani pomocí rad od uživatelů Discord fóra, na kterém se nachází hlavní uživatelská podpora pro práci s Uno Platform. Tímto jsem byl přinucen přistoupit k ukládání dat do kolekcí a pracovat tak pouze s daty, která jsou dostupná pouze za běhu aplikace.

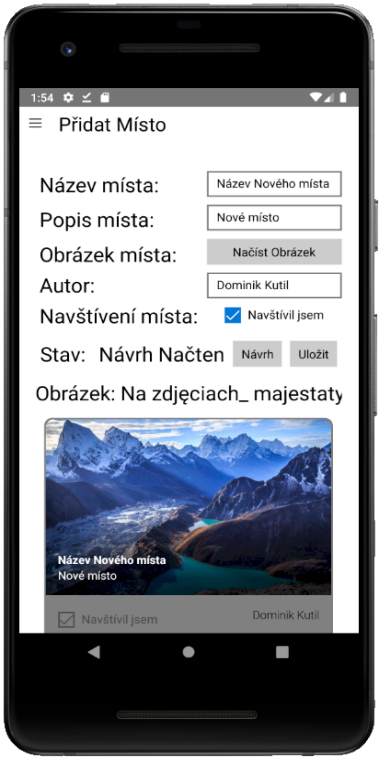
Poslední problém byl spojen s ukládáním obrázků, protože mezi platformami je odlišný způsob přístupu k lokálnímu úložišti. Musel jsem tedy přijít na univerzální metodu, jak ukládat obrázky, která by fungovala napříč platformami. Pro vyřešení tohoto problému jsem zvolil datový typ BitmapImage a ukládání souboru pomocí streamu při výběru nového souboru. Toto řešení je možné použít pouze v případě, že nepoužijeme databáze, protože databáze tento datový typ obrázku nepodporuje.

## 6.2 Průchod a funkcionalita aplikace

Při spuštění aplikací vytvořených v Uno Platform na Windows a webových prohlížečích se první zobrazí logo platformy Uno, na mobilních zařízení se logo nezobrazí a rovnou se zapíná aplikace. Po spuštění aplikace se načte stránka se všemi uloženými místy v aplikaci. Po celou dobu spuštění je dostupné postranní menu poskytující navigaci napříč aplikací. Menu se také přizpůsobuje velikosti okna aplikace. Celá aplikace je tvořená tak, aby se jakákoliv změna v datech, tedy v jednotlivých uložených místech, ihned projevila.

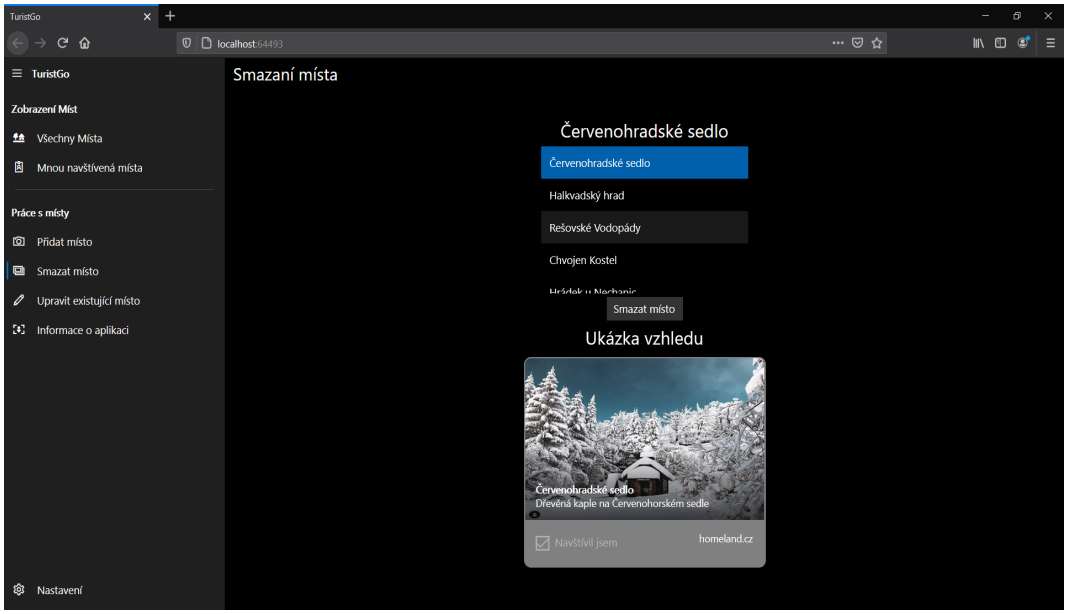
Aplikace dále obsahuje hlavní stránku, na které je postranní menu nabídka a ostatní stránky jsou načítány do prvku Frame, který se nachází na hlavní stránce s menu nabídkou. Postranní menu jsem rozdělil na dvě sekce. První sekce umožňuje nahlédnutí do všech míst a míst uvedených jako navštívených. Druhá sekce demonstruje funkce pro přidání, odstranění a upravení míst. První volitelná možnost v postranní menu nabídce zobrazuje všechny místa uložená v aplikaci, a to navštívená a nenavštívená místa. Další nabídka v postranním menu zobrazí pouze navštívená místa uložená v seznamu aplikace.

Nyní se dostáváme již k funkční sekci aplikace. První možností v této sekci je Přidání nového místa. Nové místo se přidává po vyplnění nadpisu, popisu, autora, přidání obrázku a zvolení možnosti, zda bylo nové místo navštíveno či nikoliv. Aby bylo zabráněno přidání napůl vyplněného místa, má přidání podmínku vyplnění základních údajů o novém místě a jeho obrázek. I když je splněna tato podmínka, nejdříve se musí načíst první náhled místa a až poté ho lze uložit do seznamu míst v aplikaci, a to pokud již neexistuje místo se stejným názvem.




Obrázek 21: Ukázka přidání místa v systému Android

Odstraňování míst jsem řešil načtením názvů míst do seznamu, ze kterého si uživatel vybere místo. To se načte do náhledu a poté ho lze nenávratně vymazat. Další možností je úprava již existujícího místa. Tato úprava spočívá v tom, že se vybere místo ze seznamu názvů míst a jeho vlastnosti se načtou do formuláře. Po kliknutí na tlačítko Upravit místo se zpřístupní formulář, ve kterém lze upravit parametry vybraného místa a kliknutím na druhé tlačítko Uložit změny můžeme úpravy uložit. Aplikace obsahuje menu nabídku pro informace o aplikaci, ve které je uveden rok autor a účel aplikace.




Obrázek 22: Ukázka smazání místa v prostředí prohlížeče Mozilla Firefox

Poslední nabídkou v postranním menu je nastavení obsahující čtyři funkce. Mezi ně jsem zařadil smazání všech míst v aplikaci, nastavit všechna místa jako navštívená, nastavit všechna místa jako nenavštívená a funkci, která způsobí, že když aplikace neobsahuje žádná data, nahrají se testovací data obsahující dvanáct mnou připravených míst. Testovací data jsou kvůli neúspěchu propojení aplikace s lokální databází zařazena do seznamu míst a přístupná hned po zapnutí aplikace.


Obrázek 23: Ukázka nastavení apliakce v prostředí iOS

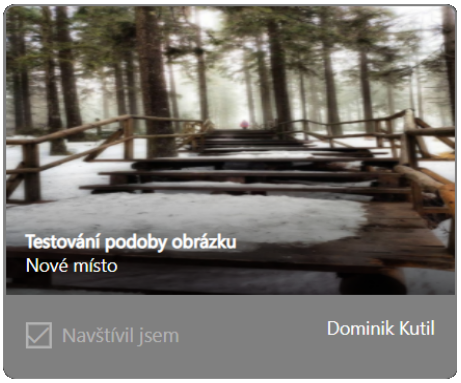
## 6.3 Testování aplikace

Testování jsem prováděl na platformách Windows, tedy na svém lokálním počítači, Android emulátoru, iOS emulátoru za pomocí virtuálního zařízení macOS a ve webových prohlížečích Google Chrome, Mozilla Firefox, Microsoft Edge nebo v prohlížeči Safari (verzi pro Windows). Protože aplikace nemá databázi a pracuje pouze s daty aplikace, rozhodl jsem se zatím nikde aplikaci nepublikovat. Po případném vyřešení problému s ukládáním dat do databáze by aplikace mohla být publikována například na Google Play. Aplikace má vygenerované řešení i pro nově podporované platformy, které přibyly již během rozpracovanosti mé práce a nebyly do ní zahrnuty. Avšak funkcionalita aplikace na těchto platformách nebyla mnou testována.

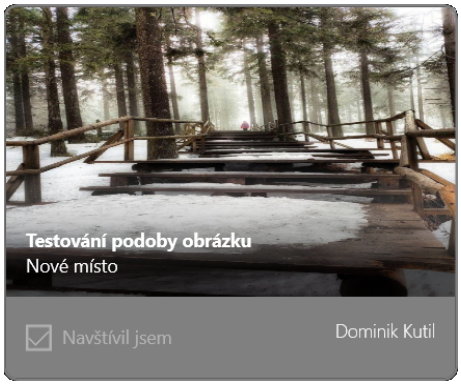
Testování jsem se rozhodl provést metodou porovnávání odlišností v zobrazení a funkcionality, s ohledem na to, že se jedna kódová základna

používá na více než na dvě platformy. Hlavním problémem během testování dělala lokální databáze. Po odebrání databáze z aplikace a použití kolekcí z C# jsem nenarazil na problém s předáváním dat mezi platformami.

Testování funkcionality a zobrazení mezi prohlížeči ukázalo na jeden rozdíl oproti jiným platformám. Vkládání obrázku prostřednictvím funkce FileOpenPicker, přidaná do Uno platformy na konci března, má o poznání horší kvalitu vloženého obrázku než na některých platformách, avšak konkrétně v prohlížečích Google Chrome a Microsoft Edge, což muže být zapříčiněno nedostatkem v jejich společném jádře Chromium. Věřím, že vývojáři opraví tento problém v příští aktualizaci, pokud je příčinou samotná Uno platforma.




Obrázek 24: Přidaný obrázek v prohlížeči Google Chrome




Obrázek 25: Přidaný obrázek Windows

Pro porovnání mezi prohlížeči jsem záměrně testoval prohlížeč Safari pro Windows, protože jeho podpora pro tento operační systém skončila skoro před 10 lety, a tudíž, jak z testování vyplynulo, se aplikace na této zastaralé verzi ani nespustila, avšak na nejnovější verzi pro macOS pracuje spolehlivě. K témuž výsledku jsem docílil při spuštění aplikace v prohlížeči Internet Explorer. Tímto jsem chtěl poukázat na fakt, že Uno Platform opravdu pracuje jen s moderními prohlížeči, které mají pravidelné aktualizace a stále se udržují.

Mezi menší rozdíl bych mohl také zařadit odlišné zobrazení fontu textu mezi platformami, a to tloušťkou písma. Další problém, na který jsem narazil byl

se zobrazením prvního náhledu ihned po zapnutí aplikace v systému Android. Problém spočíval ve špatném načtení obrázků, avšak vždy pomohla změna orientace zobrazení nebo znovu načíst danou sekci pomocí postranní menu nabídky. Poslední rozdíly, na které jsem narazil, byly v prostředí iOS, v niž se sekce nastavení pojmenovala anglicky a občas se projevovali pomalejší reakce v zobrazovaní na Androidu a iOS, což bylo zapříčiněno samotnými emulátory a výkonem mého počítače.

# 7 Závěr

Bakalářská práce se zabývá tvorbou multiplatformních aplikací za pomoci Uno Platform. Platformu Uno můžeme použít pro tvorbu aplikací na platformy Windows, iOS, Android, web a macOS. Během rozpracování mé bakalářské práce se změnila situace a byla přidána další podpora platforem, a to pro Linux, čímž přibyla další problematika ke zpracování.

V teoretické části jsem představil multiplatformní aplikace, a to jejich popis, definice, možné metody vývoje a jejich výhody a nevýhody. Před začátkem práce v Uno platformě jsem uvedl a popsal platformy, na které má Uno Platform výstup včetně jejich architektur a vybraných metod vývoje.

V praktické části jsem stručně popsal Uno Platformu, předvedl její strukturu a dostupné vývojové prostředí. Dále jsem předvedl tvorbu v platformě Uno od založení projektu a spouštění aplikace až po triviální příklady, které objasnily metody tvorby. Po těchto příkladech jsem popsal rozdíly v zobrazení mezi jednotlivými platformami a na závěr jsem metody použité v triviálních příkladech použil na souvislém příkladu.

Myslím si, že cíle práce jsem splnil, avšak totéž nemohu říct o svých cílech. Mým cílem u závěrečné aplikace bylo, aby měla lokální databázi, kterou vývojáři používali jako příklad v ukázce Uno platformy s databází SQLite, avšak řešení zobrazení dat mezi platformami má stále nějaké nedostatky. Tvorba této bakalářské práce mi umožnila zdokonalit se v tvorbě aplikací s použitím jazyku C# a XAML, se kterým jsem neměl před psaním této práce zkušenosti. Ačkoliv se mi nepodařilo plně propojit lokální databázi s aplikací, myslím, že také ve směru tvorby a práce s databázemi jsem se také zdokonalil. Rád bych dále v navazujícím studiu pokračoval s Uno Platform v diplomové práci.

Před samotným začátkem tvorby v platformě Uno bych každému doporučil, pokud nemá zkušenosti s UWP, si přečíst její dokumentaci. Vzhledem k tomu, že Uno je stavěné na metodě první vytvořit UWP tak si těmito znalostmi

do značné míry vystačíme. Jednou by Uno Platform mohlo zcela nahradit frameworky, nejen určené pro tvorbu aplikací pro systém Windows, ale i pro ostatní systémy. Vzhledem k neustálému vývoji platformy Uno se těším, co nového nám poskytne po další aktualizaci.

# Seznam použité literatury a zdrojů

[1] *What is Cross-Platform Software?* Bobology [online]. 2020 [cit. 2020-11-03]. Dostupný z: https://www.bobology.com/public/What-is-CrossPlatform-Software.cfm.

[2] *11 Advantages of Cross-Platform App Development.* Felgo [online]. 2020 [cit. 2020-11-03]. Dostupný z: https://blog.felgo.com/cross-platform-app-development/advantages.

[3] *Pros and Cons of Cross-Platform Mobile App Development.* InfoQ [online]. 2016 [cit. 2020-11-05]. Dostupný z: https://www.infoq.com/articles/mobile-cross-platform-app-development/.

[4] *Native Vs Cross-Platform Development: Pros & Cons Revealed.* Uptech [online]. 2020 [cit. 2020-11-09]. Dostupný z: https://uptech.team/blog/native-vs-cross-platform-app-development.

[5] *Co je UX/UI design (webů a aplikací)?* UX/UI design [online]. 2020 [cit. 2020-11-10]. Dostupné z: https://www.cojeuxui.cz/.

[6] *Webová, Nativní a Hybridní apliakce: Srovnávámepro a proti.* Rascasone [online]. 2020 [cit. 2020-11-11]. Dostupné z: https://www.rascasone.com/cs/blog/webova-nativni-hybridni-aplikace-klady-zapory.

[7] *The Ultimate Guide to Cross-Platform Mobile App Development.* Felgo [online]. 2020 [cit. 2020-11-12]. Dostupné z: https://blog.felgo.com/cross-platform-app-development/the-ultimate-guide#benefits.

[8] *CO JE JEDNOSTRÁNKOVÁ APLIKACE (SPA) A KDY JI VYUŽÍT?* Rascasone [online]. 2020 [cit. 2020-11-14]. Dostupné z: https://www.rascasone.com/cs/blog/jednostrankova-webova-aplikace-spa.

[9] *PROČ JSOU PROGRESIVNÍ WEBOVÉ APLIKACE (PWA) BUDOUCNOSTÍ?* Rescasone [online]. 2020 [cit. 2020-11-

16]. Dostupné z: https://www.rascasone.com/cs/blog/pwa-progresivni-webove-aplikace.

[10] *What is Hybrid App Development?* Ionic [online]. ©2020 [cit. 2020-11-16]. Dostupné z: https://ionicframework.com/resources/articles/what-is-hybrid-app-development.

[11] *Where Do Cross-Platform App Frameworks Stand in 2020?* Net solutions [online]. 2020 [cit. 2020-11-20]. Dostupné z: https://www.netsolutions.com/insights/cross-platform-app-frameworks-in-2019/.

[12] *What is Cross Platform Development?* ClearTech Interaktive [online]. 2020 [cit. 2020-11-22]. Dostupné z: https://www.cleart.com/what-is-cross-platform-development.html.

[13] *Getting Started with Android Programming.* Studytonight [online]. 2020 [cit. 2020-11-23]. Dostupné z: https://www.studytonight.com/android/introduction-to-android#.

[14] *Android.* Aktuálně.cz [online]. 2011 [cit. 2020-11-25]. Dostupné z: https://www.aktualne.cz/wiki/veda-a-technika/android-google/r~i:wiki:1424/.

[15] *What is Android Architecture (Platform Architecture).* Tutorial By EyeHunts [online]. 2018 [cit. 2020-11-26]. Dostupné z: https://tutorial.eyehunts.com/android/android-architecture-platform-architecture/.

[16] *iOS.* TechTerms [online]. 2011 [cit. 2020-11-27]. Dostupné z: https://techterms.com/definition/ios.

[17] *iOS.* LetemSvětem Applem [online]. 2019 [cit. 2020-11-27]. Dostupné z: https://www.letemsvetemapplem.eu/2019/08/13/jailbreak-pro-iphone-v-roce-2019-ma-jeste-vubec-smysl/.

[18] *Co je Mac OS X?* INDIANA UNIVERSITY [online]. 2018 [cit. 2020-11-27]. Dostupné z: https://kb.iu.edu/d/aghe.

[19] *Macintosh Operating System (Mac OS).* Techopedia [online]. 2018 [cit. 2020-11-28]. Dostupné z: https://www.techopedia.com/definition/2639/macintosh-operating-system-mac-os.

[20] *What is the difference between macOS and iOS?* Quora [online]. 2016 [cit. 2020-11-28]. Dostupné z: https://www.quora.com/What-is-the-difference-between-macOS-and-iOS.

[21] *IOS Technology Overview.* Arizona State University [online]. © 2012 Apple Inc. [cit. 2020-11-29]. Dostupné z: http://pooh.poly.asu.edu/Mobile/ClassNotes/Papers/MobilePlatforms/iOSTechnicalOverview.pdf.

[22] *Mac Technology Overview.* Documentation Archive [online]. 2015 [cit. 2020-12-15]. Dostupné z: https://developer.apple.com/library/archive/documentation/MacOSX/Conceptual/OSX_Technology_Overview/About/About.html#//apple_ref/doc/uid/TP40001067-CH204-TPXREF101.

[23] *RUSSINOVICH, Mark a David SOLOMON.* Windows Internals, Part 1. 6th edition. Redmond, Washington 98052-6399: Microsoft Press, 2012. ISBN 978-0-7356-4873-9.

[24] *Google Chrome browser.* SearchMobileComputing [online]. 2013 [cit. 2021-01-16]. Dostupné z: https://searchmobilecomputing.techtarget.com/definition/Google-Chrome-browser.

[25] *Firefox.* Hope [online]. 2018 [cit. 2021-01-16]. Dostupné z: https://www.computerhope.com/jargon/f/firefox.htm.

[26] *Microsoft Edge.* Whatls.com [online]. 2015 [cit. 2021-01-16]. Dostupné z: https://whatis.techtarget.com/definition/Microsoft-Edge.

[27] *What Is Safari?* Lifewire [online]. 2020 [cit. 2021-01-16]. Dostupné z: https://www.lifewire.com/what-is-safari-4173608.

[28] *KROGH, Einar.* An introduction to Windows Operating System [online]. 2nd edition. bookboon.com, 2017 [cit. 2021-02-08]. ISBN 978-87-403-1935-4. Dostupné z: http://www.musaliarcollege.com/e-Books/CSE/an-introduction-to-windows-operating-system.pdf.

[29] *ANDROID versions and their features.* EngineersGarage [online]. 2020 [cit. 2021-01-27]. Dostupné z: https://www.engineersgarage.com/tech-articles/android-versions-and-their-features/.

[30] *List Of All Apple iOS Version History And The First iPhone (2021).* GKGIGS [online]. 2021 [cit. 2021-02-27]. Dostupné z: https://www.gkgigs.com/list-apple-ios-version-history/#iOS_14.

[31] *List of Mac OS versions.* Fandom [online]. 2014 [cit. 2021-02-27]. Dostupné z: https://apple.fandom.com/wiki/List_of_Mac_OS_versions.

[32] *Windows.* TechTerms [online]. 2012 [cit. 2021-02-27]. Dostupné z: https://techterms.com/definition/windows.

[33] *Co je Xamarin?* Microsoft [online]. 2020 [cit. 2021-02-27]. Dostupné z: https://docs.microsoft.com/cs-cz/xamarin/get-started/what-is-xamarin.

[34] *Blazor WebAssembly 3.2.0 now available.* Microsoft [online]. 2020 [cit. 2021-02-27]. Dostupné z: https://devblogs.microsoft.com/aspnet/blazor-webassembly-3-2-0-now-available/.

[35] *What's a Universal Windows Platform (UWP) app?* Microsoft [online]. 2020 [cit. 2021-02-27]. Dostupné z: https://docs.microsoft.com/en-us/windows/uwp/get-started/universal-application-platform-guide.

[36] *Windows UI Library (WinUI).* Microsoft [online]. 2020 [cit. 2021-02-27]. Dostupné z: https://docs.microsoft.com/cs-cz/windows/apps/winui/.

[37] *ZIKMUND, Martin.* Uno PLatform – with Martin Zikmund. In: Youtube [online]. 2021-01-21 [2021-01-23]. Dostupné z: https://www.youtube.com/watch?v=2fU8H_6_0mM&t=190s&ab_channel=.NETOxford.

[38] *BULL, Petr.* Uno Platform. In: Youtube [online]. 2020-06-30 [2020-12-27]. Dostupné z: https://www.youtube.com/watch?v=P83m_py8bdk&t=901s&ab_channel=Hainton.

[39] *FAQ.* Uno Platform Doc [online]. [b.r.] [cit. 2021-03-07]. Dostupné z: https://platform.uno/docs/articles/faq.html.

[40] *How it works.* Uno Platform [online]. 2021 [cit. 2021-03-07]. Dostupné z: https://platform.uno/how-it-works/.

[41] *Getting Started on Visual Studio.* Uno Platform Doc [online]. [b.r.] [cit. 2021-03-07]. Dostupné z: https://platform.uno/docs/articles/get-started-vs.html.

[42] *Getting Started on JetBrains Rider.* Uno Platform Doc [online]. [b.r.] [cit. 2021-03-07]. Dostupné z: https://platform.uno/docs/articles/get-started-rider.html.

[43] *Getting Started on VS Code.* Uno Platform Doc [online]. [b.r.] [cit. 2021-03-07]. Dostupné z: https://platform.uno/docs/articles/get-started-vscode.html.

[44] *Create a Single Page App with Uno Platform.* Uno Platform Doc [online]. [b.r.] [cit. 2021-03-08]. Dostupné z: https://platform.uno/docs/articles/getting-started-tutorial-1.html.

[45] *Vývoj Uno Platform (část 1.).* Martin Zikmund [online]. 2019 [cit. 2021-03-10]. Dostupné z: https://blog.mzikmund.com/2019/03/vyvoj-uno-platform-cast-1/.

[46] *Understanding XAML Schemas and namespace Declarations (UWP Apps).* Microsoft [online]. 2016 [cit. 2021-03-10]. Dostupné z: https://social.technet.microsoft.com/wiki/contents/articles/33392.understanding-xaml-schemas-and-namespace-declarations-uwp-apps.aspx.

[47] *How It Works.* Uno Platform [online]. 2018 [cit. 2021-04-01]. Dostupné z: https://platform.uno/how-it-works-old/.

[48] *Uno Platform 3.6: WinUI 3, podpora WCT 7.0, nástroje pro výběr data a souborů a další.* Uno Platform [online]. 2021 [cit. 2021-04-01]. Dostupné z: https://platform.uno/blog/press-release-uno-platform-3-6/.

# Seznam obrázků


1 Android architektura (převzato z [15]) . . . . . 20
2 iOS architektura (převzato z [21]) . . . . . 24
3 MacOS architektura (převzato z [22]) . . . . . 26
4 Architektura systému Windows (převzato z [23]) . . . . . 30
5 Oficiální logo Uno Platform (převzdato z [45]) . . . . . 36
6 Architektura Uno Platfrom(převzato z [47]) . . . . . 38
7 Uno Playground . . . . . 40
8 Potřebné sady ve Visual Studiu . . . . . 41
9 Správa Rozšíření . . . . . 41
10 Založení Uno projektů . . . . . 43
11 Rozložení projektů . . . . . 44
12 Nastavení zobrazení kódu pro UWP . . . . . 44
13 Nastavení emulátoru Android . . . . . 46
14 Ukázka příkladu Prvky pro layout v prohlížeči Google Chrome . 50
15 Ukázka příkladu Základní vstupy na systému Android . . . . . 57
16 Před přidáním nového tlačítka . . . . . 60
17 Po přidání nového tlačítka . . . . . 60
18 Ukázka příkladu Navigace na systému iOS . . . . . 64
19 Ukázka příkladu Animace na systému Windows . . . . . 65
20 Ukázka vzhledu aplikace TuristGo na systému Windows . . . . . 68
21 Ukázka přidání místa v systému Android . . . . . 71
22 Ukázka smazání místa v prostředí prohlížeče Mozilla Firefox . . 72
23 Ukázka nastavení apliakce v prostředí iOS . . . . . 73
24 Přidaný obrázek v prohlížeči Google Chrome . . . . . 74
25 Přidaný obrázek Windows . . . . . 74
26 Příklad Prvky pro layout v prohlížeči . . . . . 88
27 Příklad Prvky pro layout v systému Windows . . . . . 88
28 Příklad Prvky pro layout Android . . . . . 89

29 Příklad Prvky pro layout iOS . . . . . 89
30 Příklad Základní vstupy v prohlížeči . . . . . 90
31 Příklad Základní vstupy v systému Windows . . . . . 90
32 Příklad Základní vstupy Android . . . . . 91
33 Příklad Základní vstupy iOS . . . . . 91
34 Příklad Navigace v prohlížeči . . . . . 92
35 Příklad Navigace v systému Windows . . . . . 92
36 Příklad Navigace Android . . . . . 93
37 Příklad Navigace iOS . . . . . 93
38 Příklad Animace v prohlížeči . . . . . 94
39 Příklad Animace v systému Windows . . . . . 94
40 Příklad Animace Android . . . . . 95
41 Příklad Animace iOS . . . . . 95
42 Ukázka zobrazení TuristGo aplikace ve webovém prohlížeči . . . 96
43 Ukázka zobrazení TuristGo aplikace v systému Windows . . . . 96
44 Ukázka zobrazení TuristGo aplikace v systému Android . . . . . 97
45 Ukázka zobrazení TuristGo aplikace v systému iOS . . . . . 97

# Seznam příkladů

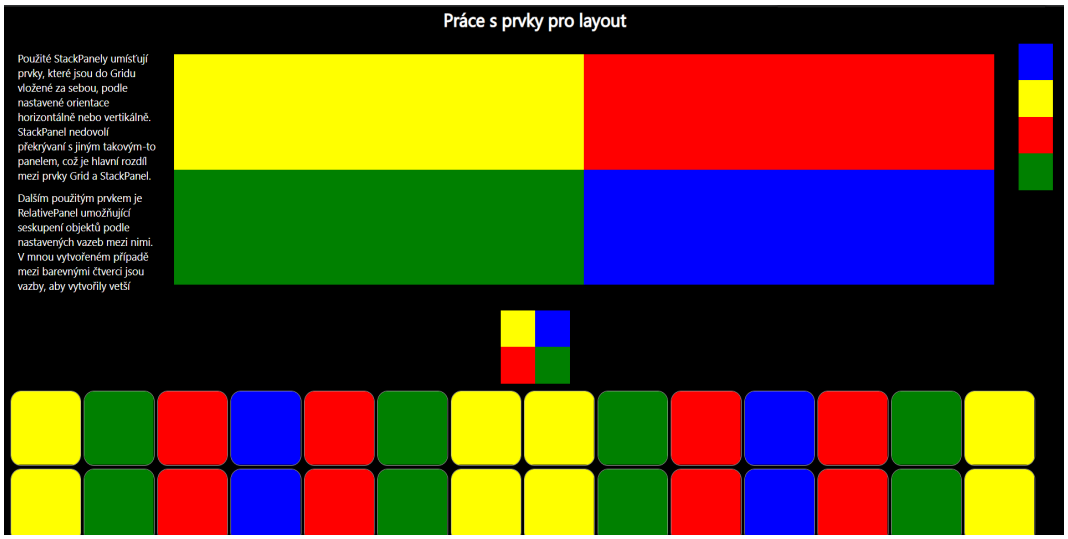
1 Schéma XAML . . . . . 47
2 Schéma příkladu Prvky pro layout . . . . . 51
3 Vnořený Grid . . . . . 52
4 RelativePanel . . . . . 53
5 GridView . . . . . 54
6 Třída Data . . . . . 55
7 Definice dat z třídy Data . . . . . 56
8 Jmený prostor pro iOS . . . . . 56
9 Prvek CheckBox . . . . . 57
10 Prvek RadioButton a jeho metoda . . . . . 58
11 Ukázka Image a ComboBox . . . . . 59
12 Ukázka prvků Slider a ProgressBar . . . . . 59
13 Ukázka prvku Button . . . . . 60
14 Ukázka metody prvku Button . . . . . 60
15 Ukázka ToggleSwitche a ProgressRing . . . . . 61
16 Ukázka metody ToggleSwitch . . . . . 61
17 XAML Kód UI navigace . . . . . 62
18 Kód metody NavigaceZmena . . . . . 63
19 Ukázka zápisu EntranceThemeTransition . . . . . 65
20 Ukázka zápisu RepositionThemeTransition . . . . . 66
21 Ukázka zápisu ReorderThemeTransition . . . . . 66

# A Příloha

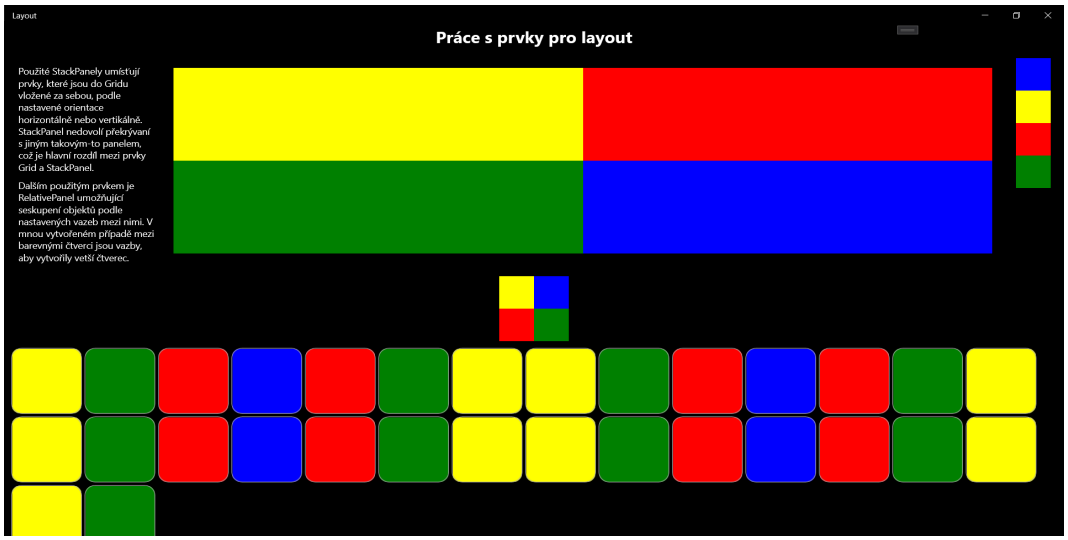
DVD - na přiloženém DVD disku se nachází:

- Bakalářská práce ve formátu PDF
- Ukázkový příklad Prvky pro layout
- Ukázkový příklad Základní vstupy
- Ukázkový příklad Navigace
- Ukázkový příklad Animace
- Závěrečná aplikace TuristGo

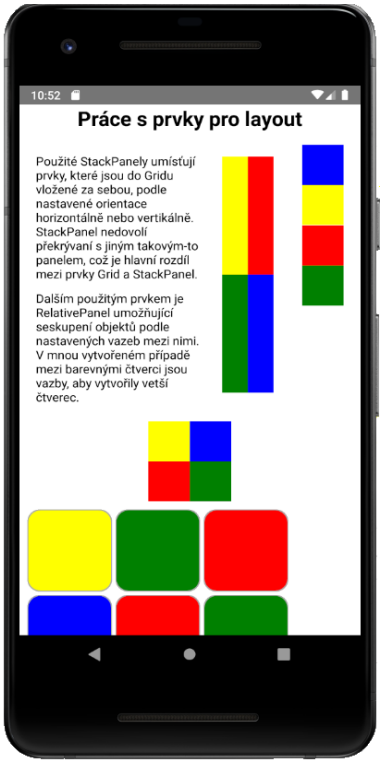
# B Obrázky zobrazení příkladu Prvky pro layout




Obrázek 26: Příklad Prvky pro layout v prohlížeči



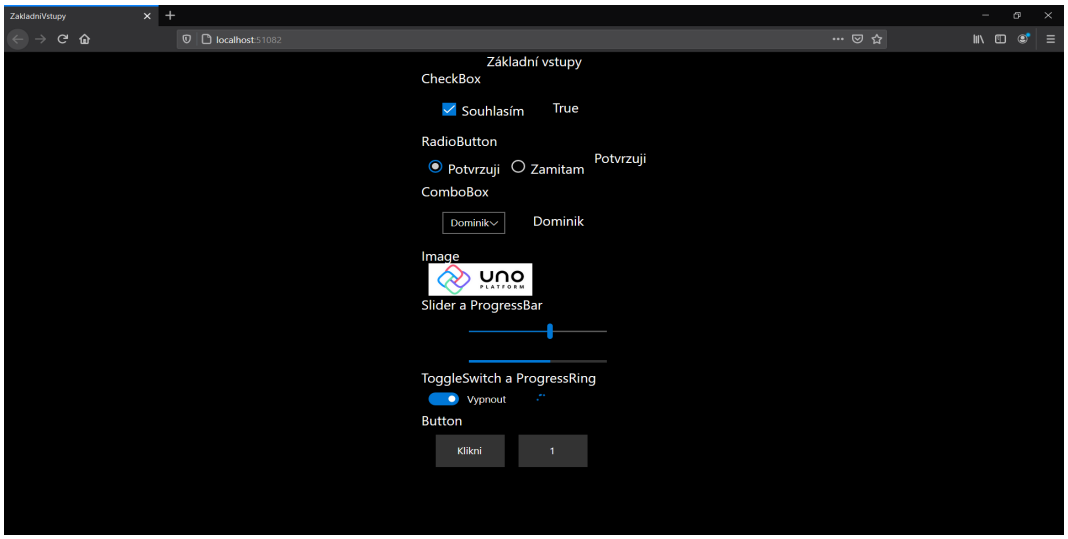

Obrázek 27: Příklad Prvky pro layout v systému Windows



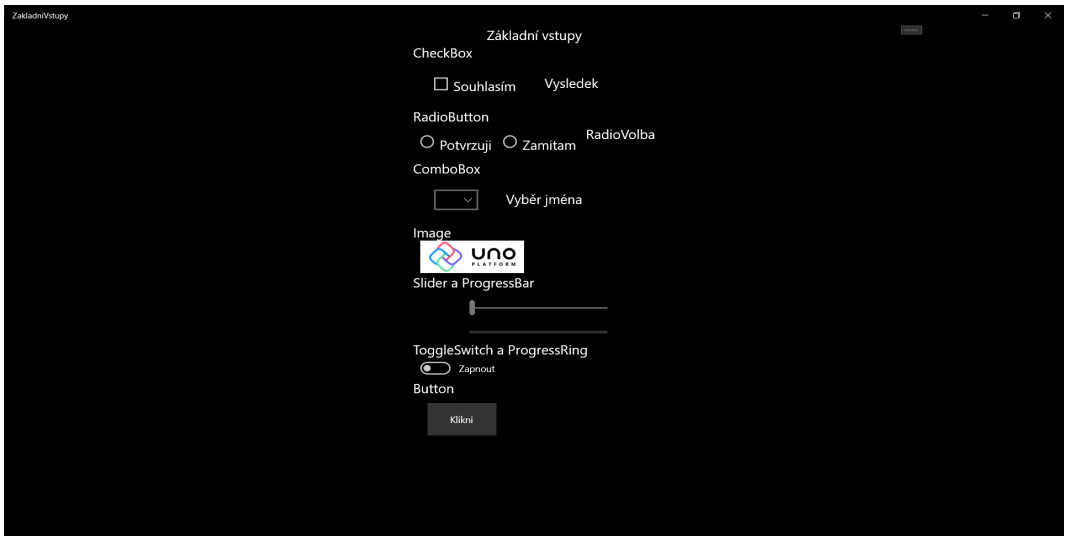

Obrázek 28: Příklad Prvky pro layout Android


Obrázek 29: Příklad Prvky pro layout iOS

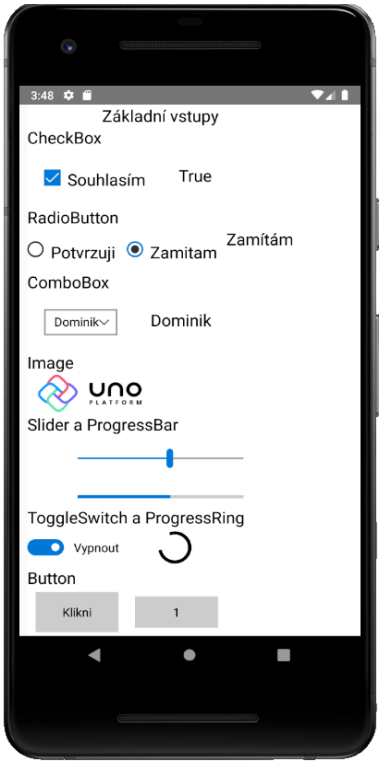
# C Obrázky zobrazení příkladu Vstupní prvky




Obrázek 30: Příklad Základní vstupy v prohlížeči



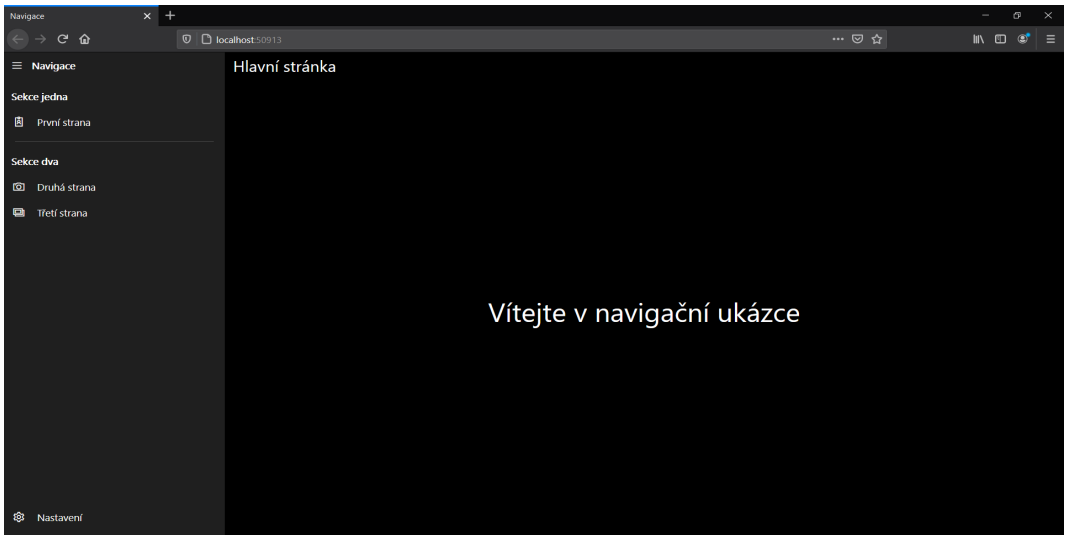

Obrázek 31: Příklad Základní vstupy v systému Windows



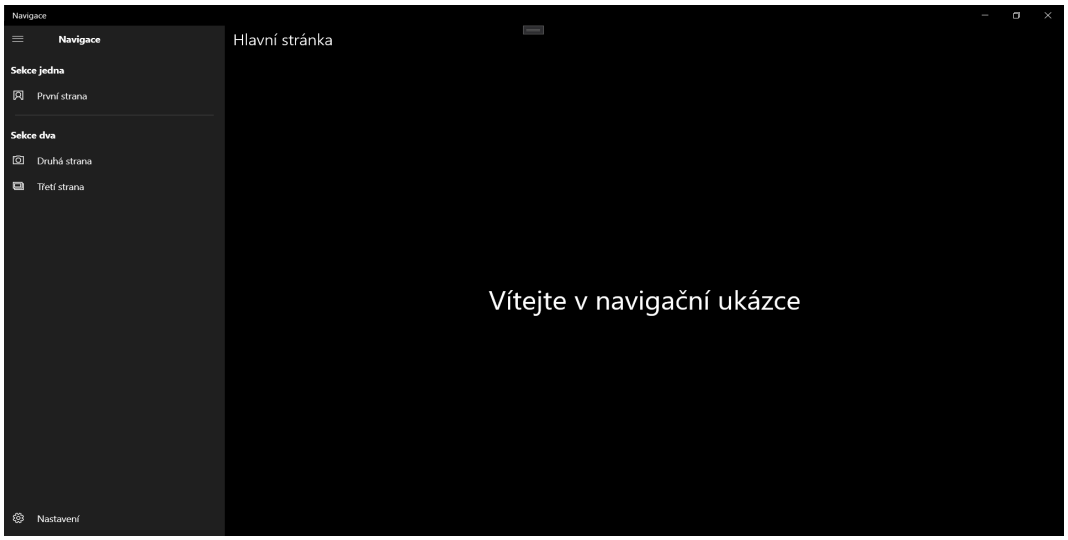

Obrázek 32: Příklad Základní vstupy Android


Obrázek 33: Příklad Základní vstupy iOS

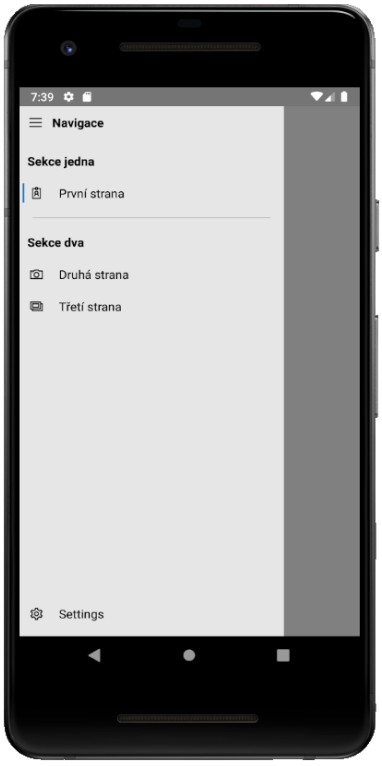
# D Obrázky zobrazení příkladu Navigace



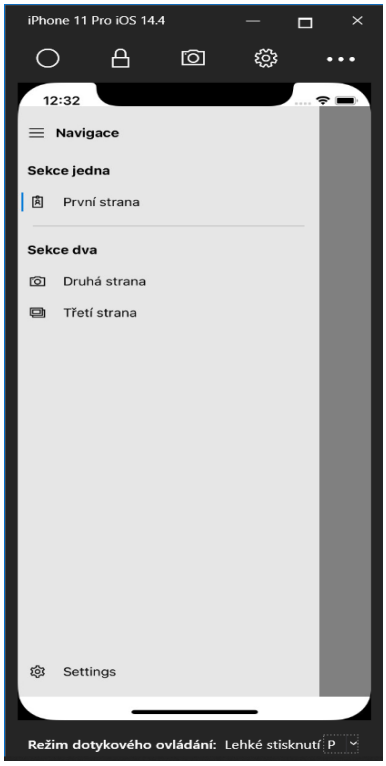

Obrázek 34: Příklad Navigace v prohlížeči




Obrázek 35: Příklad Navigace v systému Windows

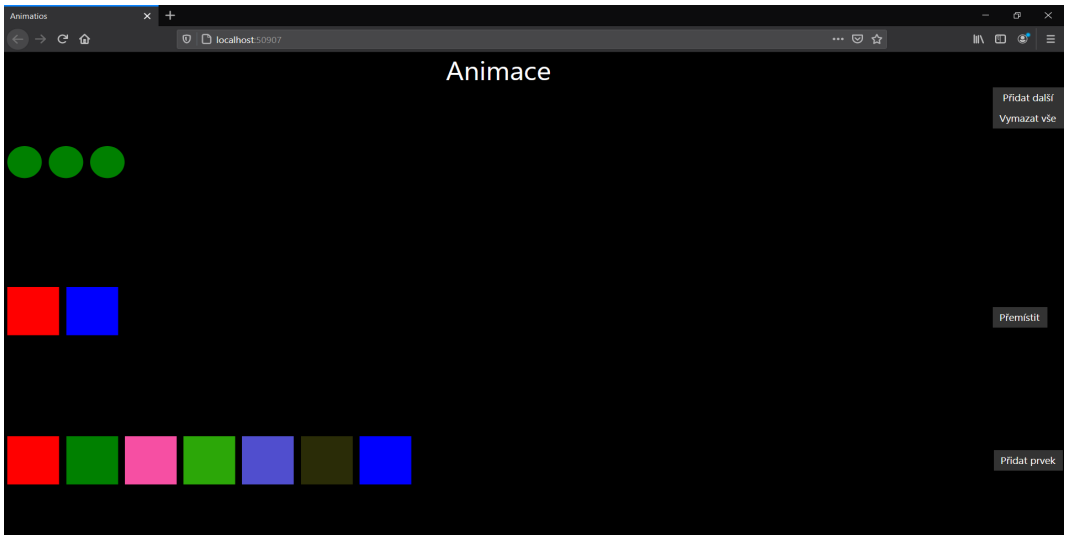



Obrázek 36: Příklad Navigace Android

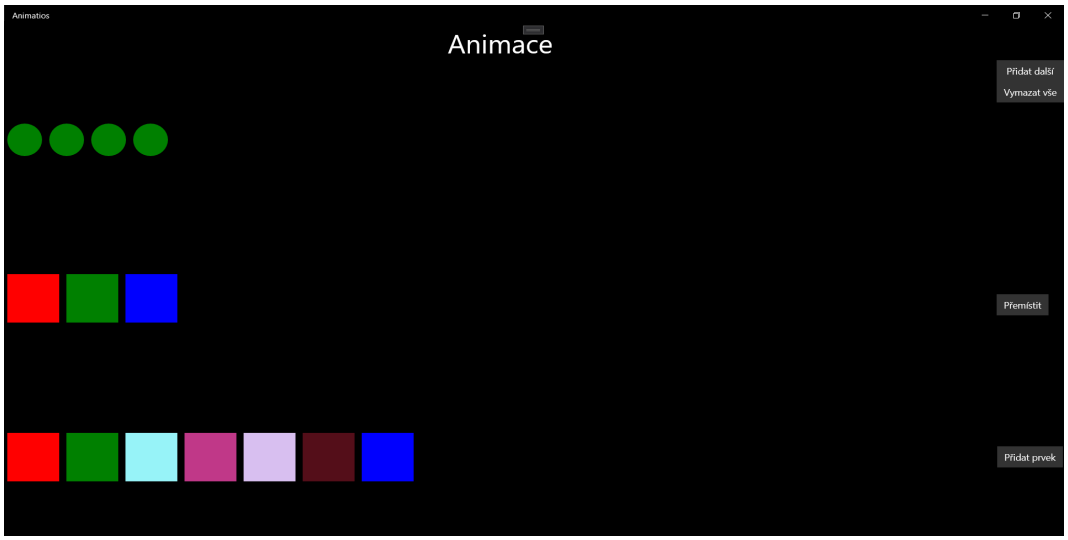



Obrázek 37: Příklad Navigace iOS

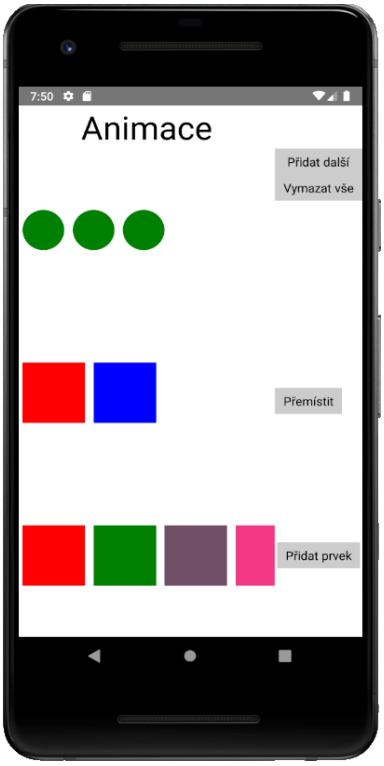
# E Obrázky zobrazení příkladu Animace



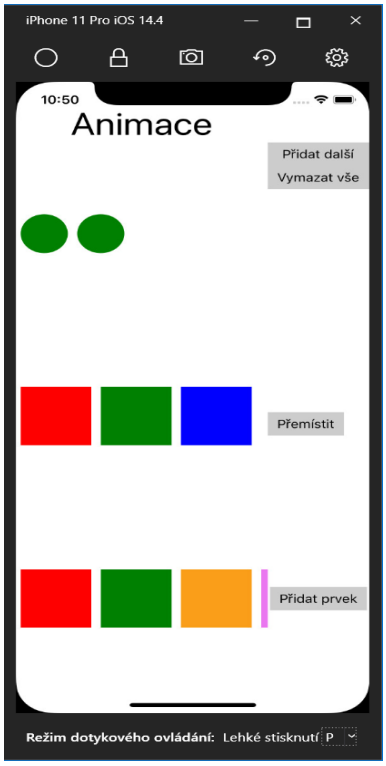

Obrázek 38: Příklad Animace v prohlížeči




Obrázek 39: Příklad Animace v systému Windows

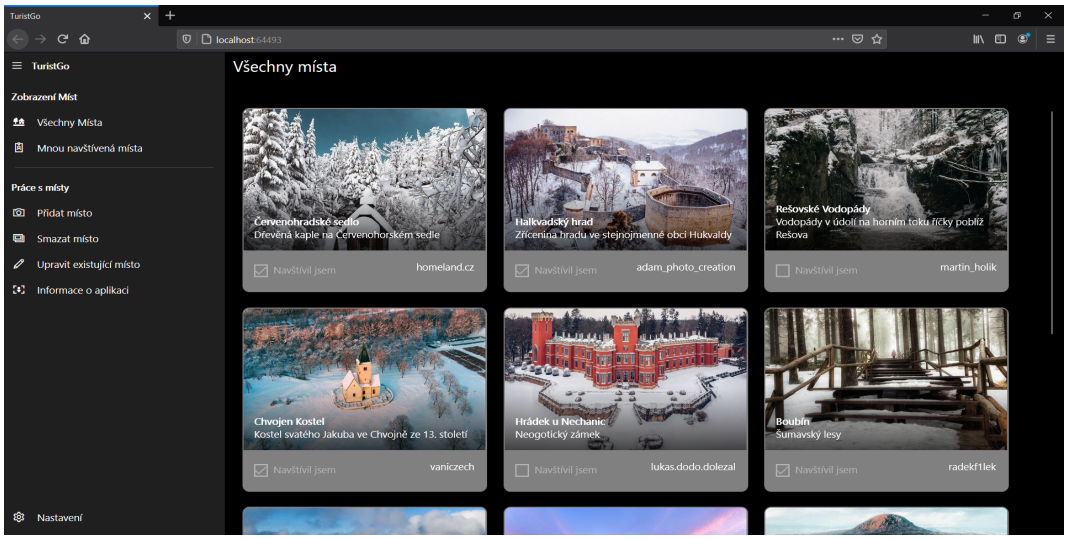



Obrázek 40: Příklad Animace Android

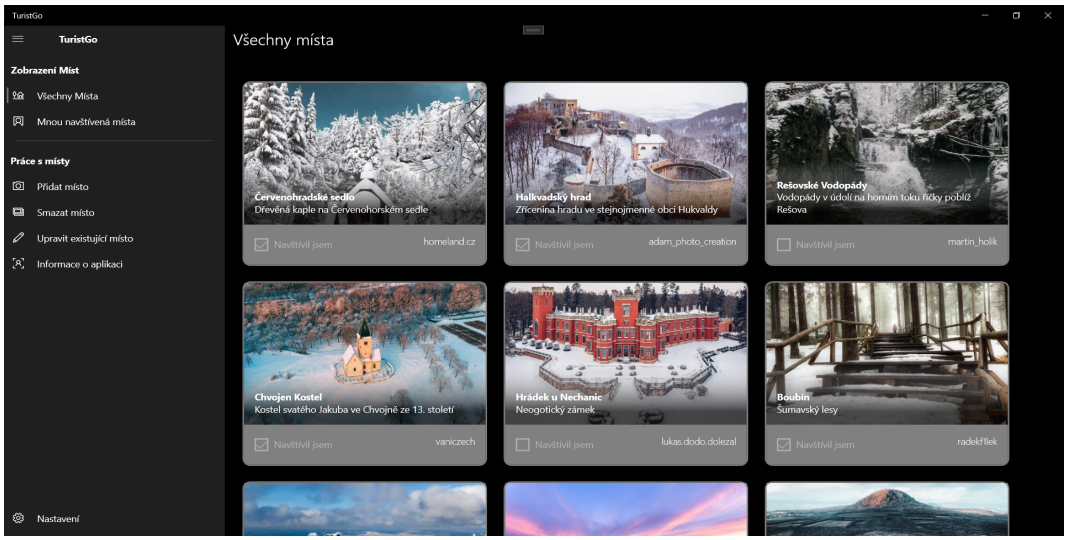



Obrázek 41: Příklad Animace iOS

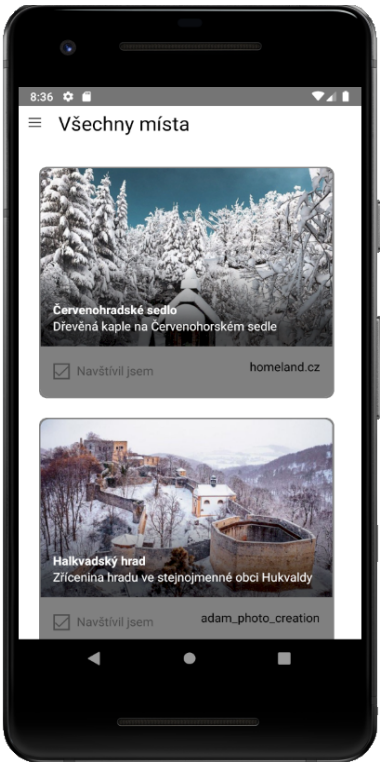
# F Obrázky sekce Všech míst z aplikace TuristGo



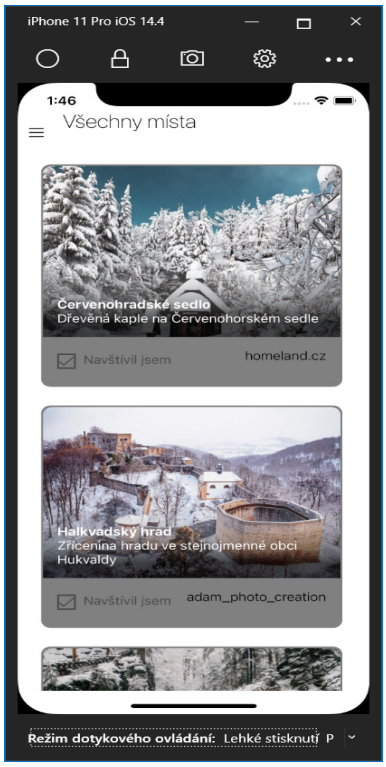

Obrázek 42: Ukázka zobrazení TuristGo aplikace ve webovém prohlížeči




Obrázek 43: Ukázka zobrazení TuristGo aplikace v systému Windows




Obrázek 44: Ukázka zobrazení TuristGo aplikace v systému Android




Obrázek 45: Ukázka zobrazení TuristGo aplikace v systému iOS