Srovnání frameworku .NET MAUI a Xamarim.Forms

Lukáš Mahďar

Bakalářská práce
2022

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Lukáš Mahďar**
Osobní číslo:	**A19060**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Srovnání frameworku .NET MAUI a Xamarin.Forms**
Téma práce anglicky:	**Comparison of the .NET MAUI and Xamarin.Forms Frameworks**

Zásady pro vypracování

1. Popište současný stav technologií pro vývoj mobilních aplikací.
2. Zaměřte se na frameworky .NET MAUI a Xamarin.
3. Navrhněte ukázkovou aplikaci demonstrující novinky ve frameworku MAUI, definujte její funkční a nefunkční požadavky, případy použití.
4. Realizujte vývoj aplikace ve frameworku .NET MAUI.
5. Popište klíčové řešení a porovnejte řešení s možným řešením ve frameworku Xamarin.
6. Demonstrujte výsledky a formulujte závěr.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. HERMES, Dan. Xamarin Mobile Application Development: Cross-Platform C# and Xamarin.Forms Fundamentals. New York: Apress, 2015. ISBN 1484269381.
2. NET Multi-platform App UI documentation. Microsoft Docs [online]. Oficiálna dokumentácia firmy Microsoft Corporation [cit. 2021-10-16]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/maui/
3. SNIDER, Ed. Mastering Xamarin.Forms. 2nd Edition. Birmingham, UK: Packt Publishing Limited, 2018. ISBN 978-1788290265.
4. .NET documentation. Microsoft Docs [online]. Oficiálna dokumentácia firmy Microsoft Corporation [cit. 2021-10-16]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/
5. Xamarin documentation. Microsoft Docs [online]. Oficiálna dokumentácia firmy Microsoft Corporation [cit. 2021-10-16]. Dostupné z: https://docs.microsoft.com/en-us/xamarin/
6. PRICE, Mark J. C# 8.0 and .NET Core 3.0: Modern Cross-Platform Development: Build applications with C#, .NET Core, Entity Framework Core, ASP.NET Core, and ML.NET u. Birmingham, UK: Packt Pub, 2019. ISBN 9781788478120.

Vedoucí bakalářské práce: **Ing. Erik Král, Ph.D.**
Ústav počítačových a komunikačních systémů

Datum zadání bakalářské práce: **3. prosince 2021**
Termín odevzdání bakalářské práce: **23. května 2022**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 24. ledna 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 18.5.2022

Lukáš Mahďar, v.r.
podpis studenta

ABSTRAKT

V bakalárskej práci sme sa zamerali na súčasný vývoj mobilných aplikácií a ich technológií. V teoretickej časti sme upriamili našu pozornosť na vysvetlenie pojmov natívna, hybridná a progresívna webová aplikácia a na objasnenie ich výhod a nevýhod. Našu pozornosť sme ďalej upriamili na mobilný operačný systém a priblíženie najrožšírenejších systémov nachádzajúcich sa v mobilných zariadeniach. V ďalšej kapitole sme sa sústredili na samotný Xamarin a jeho jednotlivé časti. Ďalej sa budeme venovať .NETu ako celku a tiež jeho častiam. V poslednej kapitole teoretickej časti sem sa zamerali na rozdiely medzi frameworkami .NET MAUI a Xamarin.Forms. V praktickej časti si popíšeme ako začať pracovať s .NETom MAUI a ako ho nainštalovať. Ďalej si predstavíme aplikácie, ktoré sú napísané v .NETe MAUI a v Xamarin.Forms a ukážeme si rozdiely medzi nimi. V ďalšej kapitole sa budeme venovať dependency injection a jeho použitiu v aplikácii, konkrétne vzoru singleton. Neskôr si predstavíme vzor MVU a jeho použitie pomocou Comet v C#. V poslednej kapitole poukážeme na jednu z hlavných výhod .NETu MAUI a tou je práca v jednotnom projekte.

Klíčová slova: .NET MAUI, Xamarin.Forms, C#, natívna, hybridná, progresívna webová aplikácia

ABSTRACT

In the bachelor thesis we focus on current development of mobile aplications and their technologies. In the teoretical part we focus on explanation concepts native, hybrid and progresive web app and their cons and pros. Than on mobile OS and which are the most common in phones. The next chapter is about Xamarin and its individual parts. Next we will focus on .NET as a whole and its parts. At the last chapter of teoretical part we will describe differences between frameworks .NET MAUI and Xamarin.Forms. In the practical part we will describe how to install and start using .NET MAUI. Than will introduce apps wrote in .NET MAUI and Xamarin.Forms and show differences beetwen them. At the next chapter we will look at dependency injection, specifically singleton, and its use in app. Later we will introduce MVU pattern and its use by Comet in C#. At the last chapter we will look at the main advantage of .NET MAUI a it is a single project.

Keywords: .NET MAUI, Xamarin.Forms, C#, native, hybrid, progressive web app

Týmto by som chcel poďakovať svojej rodine za podporu počas práce a Ing. et Ing. Erikovi Královi, Ph.D. za rady a pomoc pri zhotovovaní práce.

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Bakalárska práca vysvetľuje vývoj súčasných mobilných aplikácií a zameriava sa na frameworky .NET MAUI a Xamarin.Forms. Pri vývoji riešime veľmi veľa otázok, od typu zariadenia až po jeho operačný systém a veľa ďalších odborných problémov. Hlavným bodom tejto bakalárskej práce bude práve vývoj mobilných aplikácií. V dnešnej dobe sa najviac rozširuje práve multiplatformový vývoj, ktorý obsiahne viacero zariadení naraz. Multiplatoformový vývoj sa spája k frameworkom Xamarin.Forms, ktorý tu je už dlhší čas a nový framework od spoločnosti Microsoft .NET MAUI. Tieto frameworky budú taktiež patriť, ako už aj náov práce napovedá, medzi hlavné body tejto bakalárskej práce. S novým frameworkom .NET MAUI prichádzajú aj rozdiely oproti predchádzajúcim, ale aj veľa výhod.

Bakalárska práca je rozdelená na teoretickú a praktickú časť. Na začiatku teoretickej časti sú spísané a vysvetlené druhy aplikácií, ich výhody a nevýhody. Je veľmi dôležité vysvetliť si tieto druhy aplikácií a ich delenie. Konkrétne sa jedná o natívne, hybridné a progresívne webové aplikácie. Tiež je dôležité, vedieť a poznať, aký mobilný operačný systém obsahujú rôzne typy zariadení. Najpoužívanejšie mobilné operačné systémy sú momentálne 3 a to Android, iOS a čiastočne aj Windows Phone. Existuje veľmi veľa frameworkov na vývoj aplikácií. Stačí sa pozrieť na internet a nájdeme veľmi veľa možností na vývoj aplikácií, nájdeme rôzne frameworky aj rozdielne jazyky.

Cieľom tejto práce je zoznámiť sa s novým frameworkom na vývoj multiplatformových aplikácií .NETom MAUI a porovnať ho s frameworkom Xamarin.Forms. Nová technológia prináša veľa nových možností pri vývoji aplikácií a tieto možnosti budú tiež v práci obsiahnuté.

I. TEORETICKÁ ČÁST

1 SÚČASNÝ STAV TECHNOLÓGIÍ PRE VÝVOJ MOBILNÝCH APLIKÁCIÍ

V súčasnej dobe sa stretávame s čoraz vyššími nárokmi na vývoj mobilných aplikácií, či už s výpočetnými nárokmi, alebo so schopnosťou obslúžiť väčšinu typov zariadení. Práve tá druhá vec nás momentálne bude zaujímať. Pri vývojí aplikácií riešime najčastejšie to, aký operačný systém zariadenie používa a tak sa aplikácie rozdeľujú do niekoľko druhov.

1.1 Natívne mobilné aplikácie

Natívne aplikácie sú také, ktoré sú vyvíjané na používanie na konkrétnej platforme alebo operačnom systéme. Dva hlavné mobilné operačné systémy sú Android a iOS. Natívne aplikácie pracujú s operačným systémom zariadenia tak, že umožňujú pracovať rýchlejšie a stabilnejšie ako ostatné typy mobilných aplikácií. Vďaka možnosti prepojiť sa so špecifickými zdrojmi zariadenia môžu natívne aplikácie rýchlo pristupovať k viacerým službám zariadenia, ako je napríklad mikrofón, fotoaparát alebo akcelerometer [1].

Výhody

Najväčšou výhodou natívnych aplikácií je plný prístup k mobilnému zariadeniu, čo znamená prístup ku všetkým funkciám dostupných v zariadení. Ak máme aplikáciu, ktorá má premyslenú grafiku alebo prispôsobené ovládacie prvky, tak najčastejšie budeme voliť natívne aplikácie. Taktiež ponúkajú väčšiu bezpečnosť a to vďaka vrstvám operačného systému. Natívne aplikácie môžu pracovať niekedy alebo aj stále v offline režime. Máme prístup k vyrovnávacej pamäti zariadenia a môžeme získavať údaje zo zariadenia bez potreby sieťového pripojenia [36].

Nevýhody

Medzi hlavné nevýhody patria vyššie náklady na vývoj a potreba skúsených vývojárov. Vývoj natívnej aplikácie si vyžaduje určitú formu správy vývoja a tým je zložitejší vývoj a uvedenie natívnej aplikácie na trh. Vyžaduje si tiež zložitejší prístup k aktualizáciám, obslúžiť viacero platforiem naraz a byť prívetivý k užívateľom [36].

Obrázok 1 Natívne vs Hybridné aplikácie [2]

1.2 Hybridné aplikácie

Hybridná aplikácia, ako už z názvu vypovedá, je kombinácia natívnej aplikácie, ktorá je dizajnovaná pre konktrétne zariadenia a webovej aplikácie. Pracuje podobne ako webová aplikácia, ale inštaluje sa do mobilného zariadenia ako natívna aplikácia. Hybridné aplikácie majú taktiež prístup k interným rozhraniam zariadenia, čo znamená, že môžu používať zdroje ako fotoaparát, GPS atď. Môžu poskytnúť najlepšie z oboch svetov, ale tým majú v sebe aj ich nevýhody [3].

Výhody

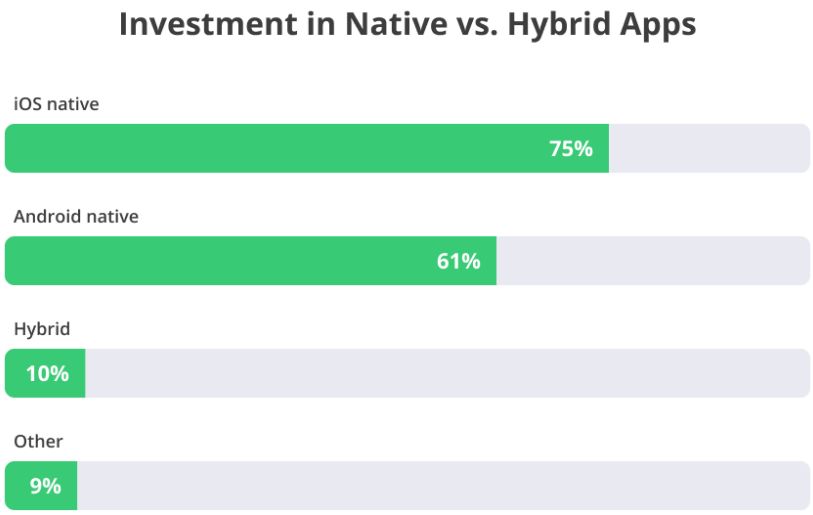
Najväčšou výhodou hybridných aplikácií je, že sú multiplatformové. Veľká flexibilita hybridných aplikácií bez potreby vyvíjať ich pre viaceré platformy zároveň, má za následok, že vývoj nie je až tak náročný a komplikovaný. Tým sa šetria vývojové náklady pri vytváraní viacerých verzií tej istej aplikácie a ich aktualizáciách [32].

Nevýhody

Hlavnou nevýhodou je určite nižšia rýchlosť a menšia bezpečnosť aplikácií. Hybridné aplikácie môžu pracovať offline, ale iba do toho času, ak framework nevyžaduje stále pripojenie k sieti. To má za následok, že niektoré funkcie nemusia byť v tú chvíľu dostupné a tým môže používateľ stratiť na záujme [32].

1.3 Hlavné rozdiely

Hybridné aplikácie pracujú rozdielne ako natívne aplikácie. Hybridné aplikácie sú stavané na webových aplikáciách a obsahujú rovnaké vyhľadávacie elementy. Taktiež hybridné aplikácie neobsahujú offline režim, takže pracujú iba ak je dostupné pripojenie k internetu. Naopak natívne aplikácie môžu pracovať aj v offline režime. Hybridné aplikácie sú stavané pomocou HTML a CSS. Vývojár napíše jeden zdrojový kód, ktorý zmenami upravuje, aby sa prispôsobil každej platforme. Všeobecne pre každú platformu pri hybridných aplikáciách je potrebých menej vývojárov ako pri natívnych aplikáciách [1][3].



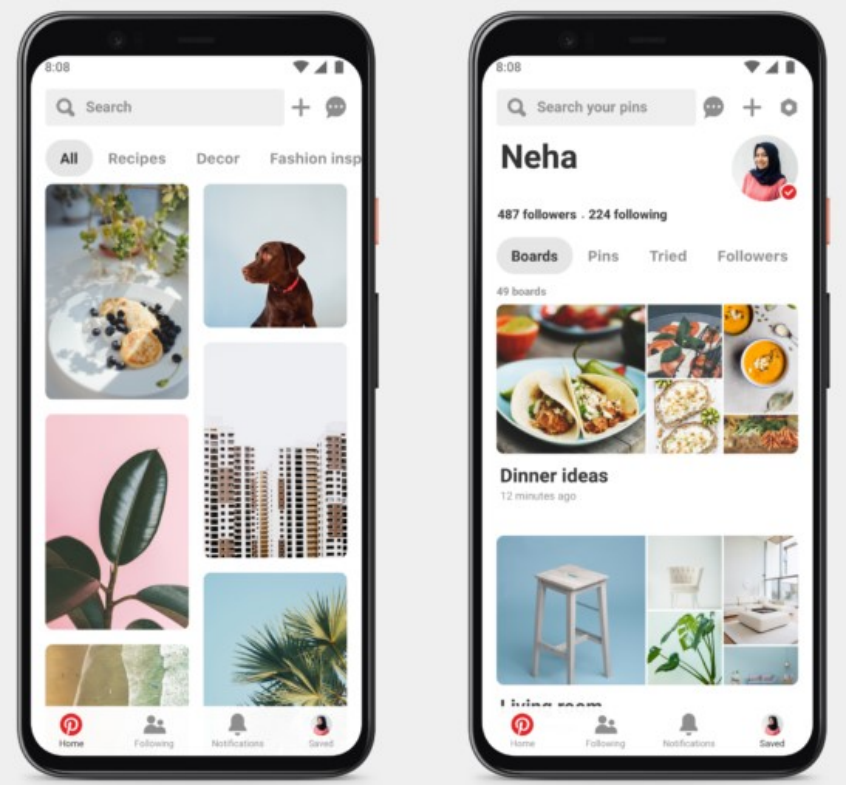
Obrázok 2 Graf podielu Natívnych vs Hybridných apliácií [4]

Na grafe môžeme vidieť, že firmy sa najčastejšie zameriavajú na vývoj natívnych aplikácií pre vývojom hybridných a to v celkovo vo veľkom pomere.

1.4 Progresívne webové aplikácie

Progresívne webové aplikácie sú weby, ktoré dokážeme nainštalovať do mobilného zariadenia. Tvária sa ako klasické natívne aplikácie, a tak aj užívateľovi pripadajú. Takiež môžu pracovať aj v režime offline a majú prístup k službám zariadenia, ako napríklad push notifikácie alebo fotoaparát. Veľkou výhodou je responzívny dizajn, čo znamená, že sa prispôsobí veľkosti daného zariadenia, na ktorom je aplikácia nainštalovaná. PWA sú vyvíjané ako web app first, čo znamená, že musia pracovať na všetkých operačných systémoch a prehliadačoch. Takže máme jeden zdrojový kód pre rôzne operačné systémy, čo vo veľkom aj

ušetrí náklady na vývoj a čas. Najpopulárnejšie frameworky pre tvorbu PWA sú Ionic, Angular a React. PWA používame takmer každý deň, dobrým príkladom sú napríklad Twitter, Pinterest alebo Uber [5].



Obrázok 3 Ukážka PWA Pinterest [6]

Na obrázku môžeme vidieť ukážku progresívnej webovej aplikácie Pinterest, ktorá slúži na zdieľanie a prezeranie fotiek a obrázkov.

Výhody

Progresívne webové aplikácie sa inštalujú priamo z webového prehliadača do zariadenia, takže nie je potrebný žiadny obchod, ako App Store alebo Google Play. Nainštalovanú aplikáciu si môžeme uložiť na plochu, kde potrebujeme a je spustiteľná cez ikonu. Taktiež sú veľmi dobrou voľbou pokiaľ chceme, aby aplikácia fungovala na viacerých zariadeniach a na rôznych veľkostiach. Tu prichádza ďalšia výhoda - responzívny dizajn[18]. Vďaka tomu, že sú multiplatformové, skracujú čas na vývoj a vydanie aplikácie používateľom. Výhoda je aj na strane bezpečnosti, pretože PWA používajú HTTPS k udržovaniu a správe bezpečnosti dát, aby zabránili k úniku informácií a narušeniu bezpečnosti aplikácie. Ako som spomínal PWA môžu posielať push-notifikácie, ktoré sa zobrazia používateľovi na

obrazovke zariadenia. Taktiež sa aktualizujú automaticky a nepotrebujú používateľovo povolenie [17].

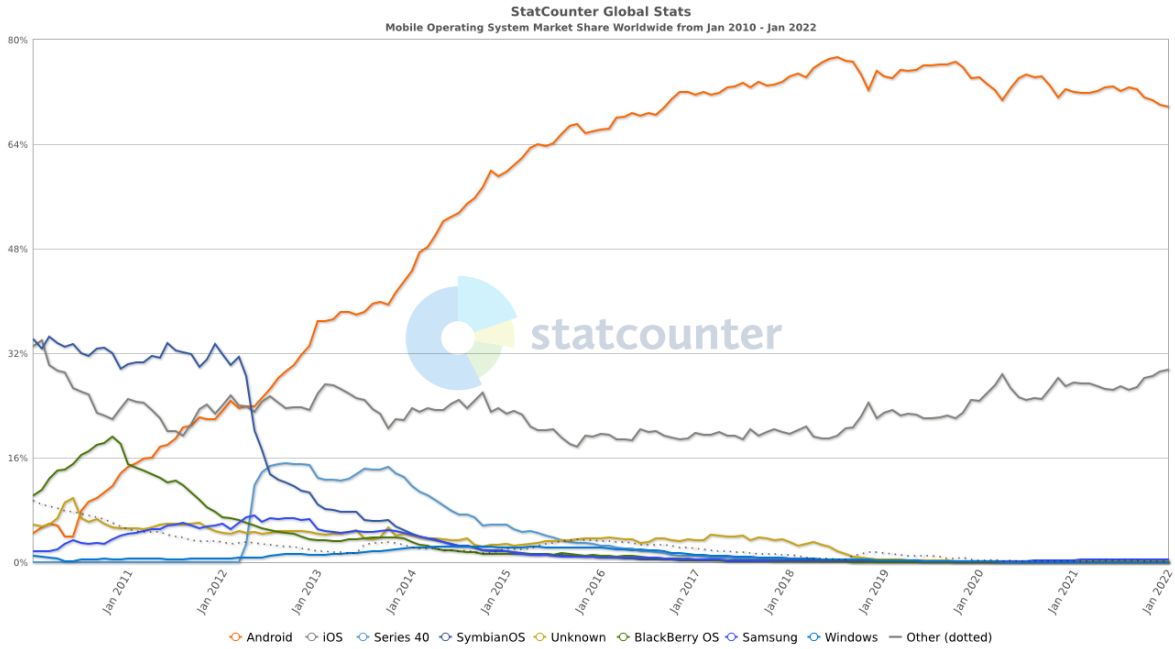
Nevýhody

Niektoré staršie verzie iOSu nepodporujú, aby PWA využívali všetky služby zariadenia ako Touch ID alebo Face ID. Nižšia rýchlosť oproti natívnym aplikáciám. Môžu mať za následok rýchlejšie vybíjanie batérie, pretože vyžaduje intenzívnejšie využitie CPU zariadenia [19].

2 MOBILNÉ OPERAČNÉ SYSTÉMY

Ako sme vyššie spomínali, v mobilných zariadeniach sa nachádzalo veľa rôznych operačných systémov. V dnešnej dobe sa ich udržalo už iba pár. My si predstavíme a popíšeme najhlavnejšie a najviac využívané mobilné operačné systémy.

Mobilný operačný systém sa líši od klasického desktopového tým, že je prispôsobený ovládaniu dotykom [33]. Rôzni výrobcovia používajú rôzne operačné systémy, aj keď v dnešnej dobe, keď si zoberiete do ruky mobilný telefón, takmer s určitosťou môžeme povedať, že má v sebe operačný systém Android alebo iOS [3].



Obrázok 4 Graf podielu mobilných operačných systémov [7]

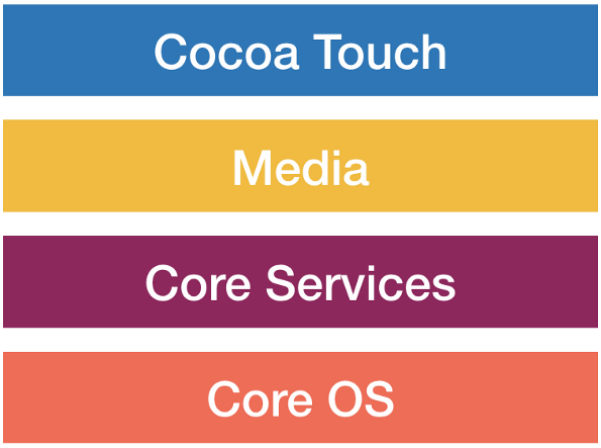
Z grafu môžeme vidieť, že takmer celé zastúpenie operačných systémov je medzi Androidom a iOS-om, ostatné operačné systémy majú zanedbateľné percento. V grafe tiež môžeme vidieť dominanciu systému Android so 70% , na ktorého sa zatiaľ nedoťahuje druhý spomínaný iOS s 29,5%.

2.1 iOS

Operačný systém iOS od spoločnosti Apple je druhý najpopulárnejší mobilný operačný systém. IOS je odvodený od Unixu a poháňa všetky mobilné zariadenia Apple. Operačný systém je dostupný v tabletoch a mobilných zariadeniach len od spoločnosti Apple, čo prináša výhodu oproti operačnému systému Android. To má za následok lepšiu optimalizáciu

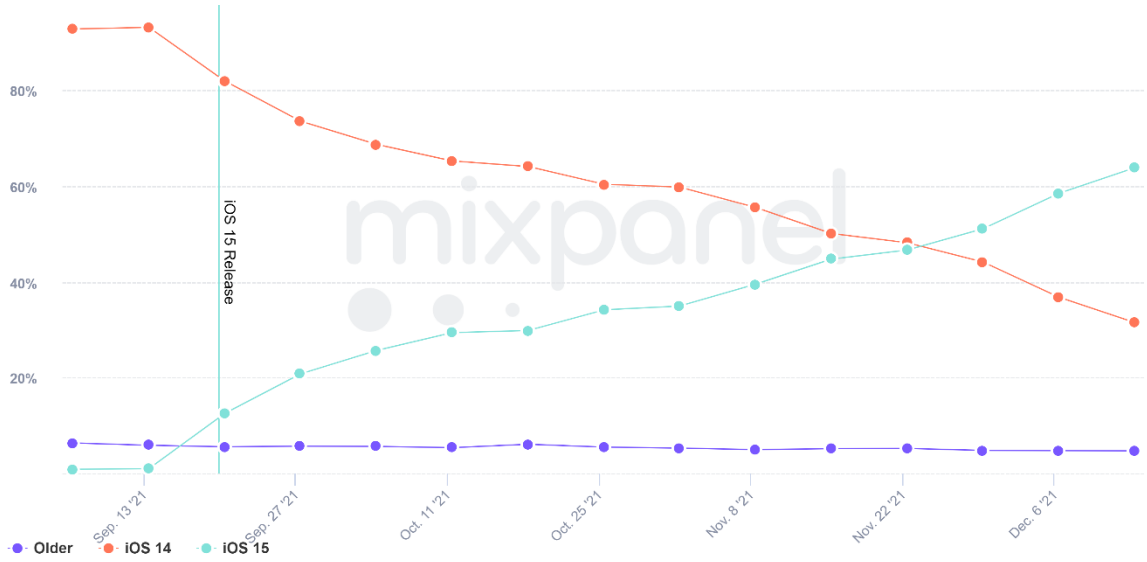
a bezpečnosť, taktiež nové aktualizácie sú rýchlejšie a ďaleko stabilnejšie. Hovoríme o tzv. uzavretom ekosystéme. V posledných rokoch sa mobilné telefóny nazývané Iphone stali najpopulárnejšie na celom svete [8].

Architektúra iOS je rozdelená do štyroch vrstiev. Na najvyššej úrovni iOS pracuje ako sprostredkovateľ medzi základným hardwerom a aplikáciami, pretože aplikácie s ním nekomunikujú priamo. Aplikácie komunikujú pomocou systémových rozhraní. Tieto rozhrania umožňujú jednoduchšie programovanie aplikácií, ktoré pracujú aj na zariadeniach s rozdielnym hardwerom. Nižšie vrstvy poskytujú základné služby všetkým aplikáciám a vyššie vrstvy poskytujú zase grafiku a služby súvisiace s rozhraním [9][10].



Obrázok 5 Názvy vrstiev iOS [11]

- **Cocoa touch:** v tejto vrstve sa nachádzajú frameworky, ktoré sú potrebné na vývoj vizuálneho rozhrania pre aplikácie. Takže frameworky v tejto vrstve poskytujú základnú aplikačnú infraštruktúru pre implementáciu užívateľského rozhrania, multitasking, interakciu s užívateľom a ďalšie vysokoúrovňové systémové služby.
- **Media:** táto vrstva poskytuje zvukové, grafické a video frameworky, čo umožňuje vytváranie grafických aplikácií, videí a zvuku.
- **Core Services:** vrstva poskytuje základné systémové služby, ktoré sú potrebné pre aplikácie a stará sa o vysokoúrovňové služby. Nachádzajú sa tu služby ako na podporu lokalizácie a iCloud-u.
- **Core OS:** základná vrstva, ktorá sa stará o nízkoúrovňové služby ako správa pamäte, súborový systém a zabezpečovacie služby [10].



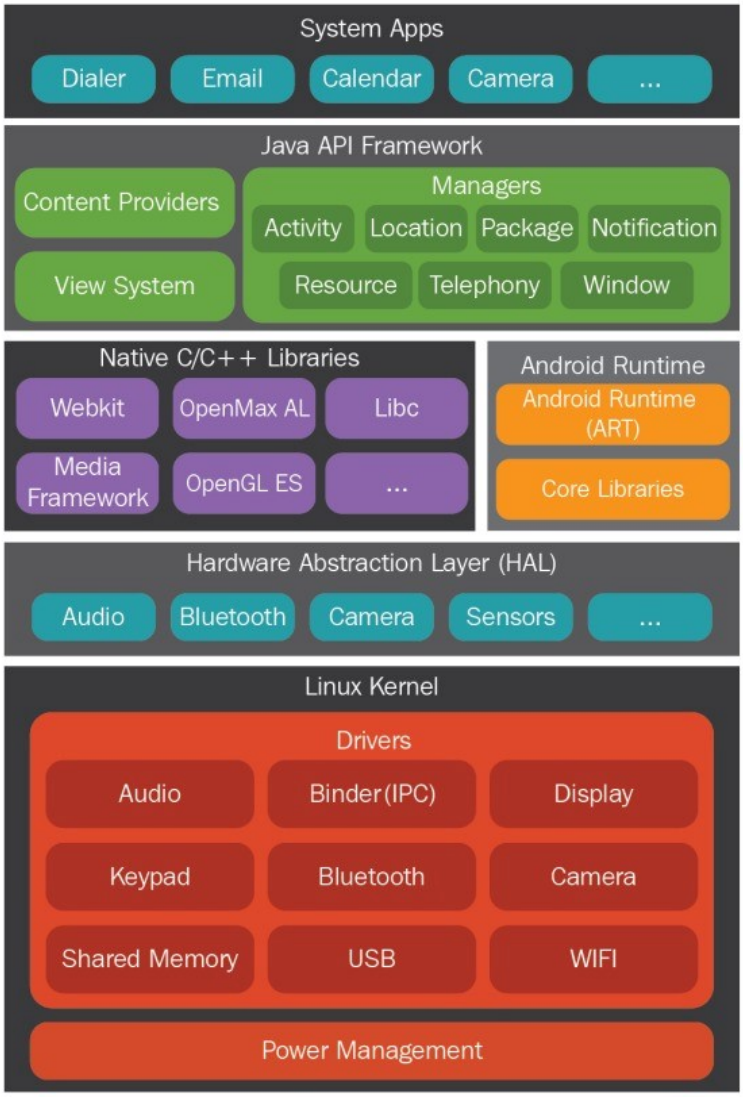
Obrázok 6 Graf verzií iOS [12]

Z grafu môžeme vidieť, že momentálne prevláda najnovšia verzia iOS 15 a staršia upadajúca verzia iOS 14 je na druhom mieste. Ostatné staršie verzie sa držia pod 4% pri momentálnom pomaly zostupnom trende.

2.2 Android

Android je najpopulárnejší mobilný operačný systém od spoločnosti Google, ktorý sa nachádza nielen v mobilných zariadeniach, ale aj v tabletoch alebo smart televízoroch. Odhaduje sa, že poháňa približne 2,5 miliardy zariadení po celom svete [13].

Je to operačný systém založený na Linuxe, vyvíjaný pre dotykové ovládanie. Na rozdiel do operačného systému iOS, je Android open-source. To znamená, že ktokoľvek má k nemu prístup a môže si ho upravovať podľa seba. Najčastejšie to využívajú výrobcovia mobilných telefónov a s tým je spojená aj najväčšia nevýhoda Androidu. Touto nevýhodou je veľká fragmentácia, pretože niektorí výrobcovia vytvárajú akúsi nadstavbu Android-u vo svojich zariadeniach. Čiže pridávajú vlastné používateľské rozhranie. Toto má za následok horšiu stabilitu operačného systému a taktiež podporu na zariadeniach oproti iOS-u. Na sťahovanie aplikácií slúži Google Play obchod, niektoré zariadenia ho však nemusia podporovať a na to slúžia obdobné obchody. Android má tiež výhodu, že podporuje sťahovanie aplikácií priamo z prehliadača, ale najskôr musí užívateľ povoliť sťahovanie aplikácií z nedôveryhodných zdrojov [14][10].



Obrázok 7 Rozdelenie vrstiev Androidu [10]

Rovnako ako v iných operačných systémoch aj Android pracuje vo vrstvách. Každá vrstva poskytuje služby vrstvám, ktoré sú nad ňou a vykonáva operácie, ktoré podporujú funkcie operačného systému. Tieto vrstvy sú:

- **Linux Kernel(jadro):** OS Android je postavený nad jadrom Linuxu s niektorými zmenami. Jadro pracuje ako abstrakcia medzi hardwérom a softwérom a spravuje základné funkcie operačného systému, ako je správa procesov, pamäte alebo bezpečnosť.
- **Hardware Abstraction:** Táto vrstva pozostáva z niekoľko knižníc modulov, ktoré implementujú rozhrania pre špecifický typ komponentu. Toto umožňuje výrobcom hardwéru implementovať funkčnosť bez zmeny systému vyššej úrovne.

- **Libraries:** V tejto vrstve sa nachádzajú natívne knižnice Android. Pomáhajú zariadeniu zvládať rozdielne druhy dát. V tej istej vrstve sa nachádza Android Runtime, ktorý je zodpovedný za spúšťanie aplikácií.
- **Java API framework:** Vrstva zodpovedá za obsluhu základných funkcií, ako je napríklad správa zdrojov. Cez túto vrstvu nainštalované aplikácie priamo komunikujú.
- **System Apps:** Táto vrstva umožňuje užívateľovi priamo interagovať so zariadením [10].

2.3 Windows Phone

Za zmienku určite stojí operačný systém od spoločnosti Microsoft, ktorý pred niekoľkými rokmi bol v značnej miere obsiahnutý v mobilných zariadeniach. Dnes už to tak nie je. Poslednú stabilnú verziu, ktorú Microsoft vydal, bola verzia Windows 10 Mobile. Na konci roku 2019 ju Microsoft prestal podporovať a vydávať nové updaty [15].

Obrázok 8 Ukážka domovskej obrazovky Windows Phone[16]

Na rozdiel od Android-u a iOS-u Windows Phone prišiel s novým užívateľským prostredím, ktoré môžeme vidieť na obrázku. Klasické ikony boli zmenené na takzvané dlaždice, ktoré si môže používateľ upraviť podľa seba.

3 XAMARIN

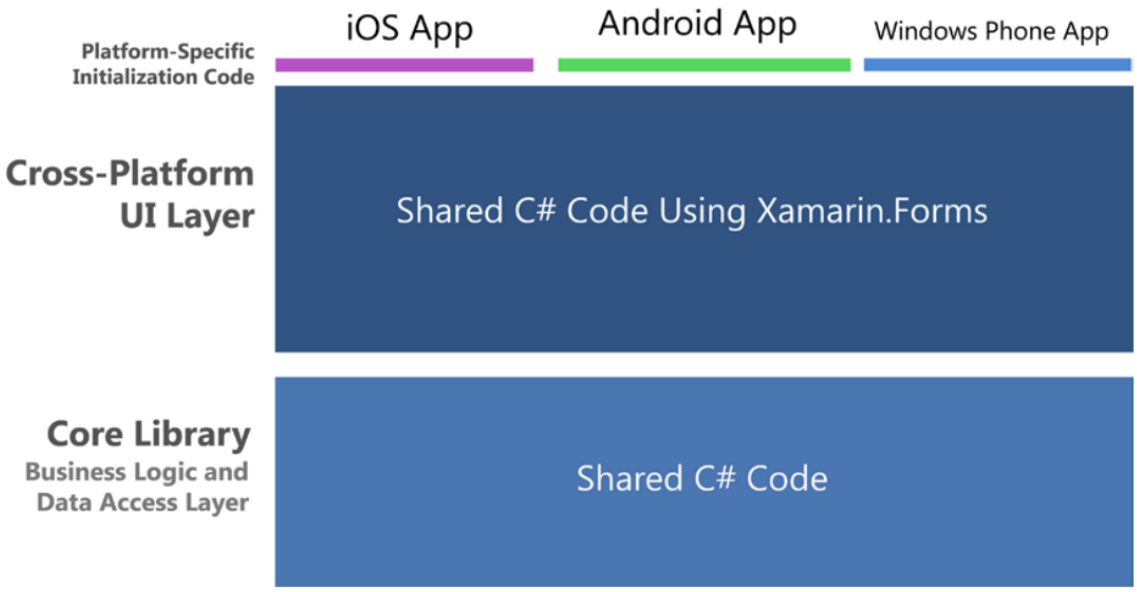
Xamarin je open source platforma na vývoj od spoločnosti Mircrosoft, ktorá nám umožňuje programovať natívne a multiplatformové iOS, Android a Windows Phone aplikácie v C#. Čiže je určená pre vývoj, kde potrebujeme zdieľať spoločný kód. Môžeme povedať, že Xamarin je abstraktná vrstva, ktorá riadi komunikáciu zdieľaného kódu a umožňuje zdieľať až 90% kódu. Xamarin používa C# a natívne knižnice zabalené do .NET vrstvy pre multiplatformový vývoj [20].

Obrázok 9 Architekúra Xamarin [21]

3.1 Xamarin.Forms

Xamarin.Forms je open-source multiplatformový framework na vývoj aplikácií pre iOS, Android a Windows Phone. Môžeme povedať, že to je súprava nástrojov multiplatformových tried užívateľského rozhrania postavená na základe tried UI, špecifické pre danú platformu ako Xamarin.iOS a Xamarin.Android. Xamarin.iOS a Xamarin.Android mapujú triedy do ich natívnych SDKs. Toto poskytuje multiplatformovú sadu uživateľského rozhrania, ktoré sa vykresľuje v každom z troch natívnych operačných systémov.

Xamarin.Forms poskytuje multiplatformový nástroj pre stránky, layouty a kontroléry. Tieto prvky sú vytvorené pomocou jazyka XAML (Extensible Application Markup Language) alebo pomocou C# použitím tried Page, Layout a View. Toto rozhranie poskytuje niekoľko vstavaných multiplatformových vzorov pre uživateľské rozhrania. Xamarin.Forms obsahuje triedy, ktoré nie sú závislé od platformy, ale sú viazané na natívne prvky špecifické pre danú platformu. Vďaka tomuto môžeme vyvíjať natívne používateľské rozhrania pre všetky platformy (iOS, Android, Windows Phone) [20].



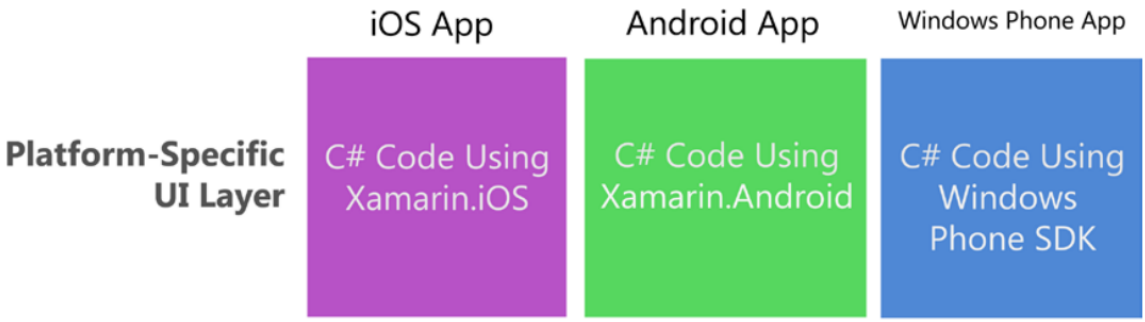
Obrázok 10 Architektúra Xamarin.Forms [20]

Graf architektúry Xamarin.Forms, kde môžeme vidieť, ako funguje zdieľanie kódu.

3.2 Xamarin.Android

Xamarin.Android je framework, ktorý nám umožňuje vytvárať aplikácie pre OS Android. Väzby C# nás spájajú s Android rozhraním. Android aplikácie sa skladajú z layoutov a aktivít, čiže môžeme povedať, že sú to View a Controllers. Zvyčajne sú layouty ako XML súbory upravované pomocou UI dizajnéra, ktorý definuje ovládacie prvky na obrazovke. Activity je trieda, ktorá zvyčajne riadi životný cyklus jedného layoutu. Nachádzajú sa tu menšie layouty, ktoré sa nazývajú fragmenty. Tie môžeme kombinovať tak, aby vytvorili obrazovku. Ovládacie prvky sa nazývajú Views a umiestňujeme ich na iný druh Layoutu. Tieto Layouty dedia od triedy ViewGroup a môžu byť generované dynamicky pomocou dátových väzieb tried Adapters. Layouty môžeme vytvoriť pomocou XML súborov alebo

naprogramovať ich v C#. XML súbory majú výhodu, že sú lepšie čitateľné a tak podporujú priamu úpravu XML kódu, preto mobilní developeri uprednostňujú túto cestu [21].



Obrázok 11 Architekúra Xamarin.Android[21]

Architektúra riešenia špecifická pre Android platformu.

3.3 Xamarin.iOS

Xamarin.iOS je framework, ktorý nám umožňuje vytvárať aplikácie pre operačný systém iOS. V Xamarin.iOS nás C# väzby prepojujú s natívnym rozhraním API pre IOS, ktorý sa nazýva UIKit. Views sú obvykle postavené pomocou návrhových nástrojov a výsledkom toho je XML súbor. ViewController je trieda ovládača, ktorá má na starosti Views a spravuje ich [21].

Pracujeme tu vo vrstvách ako zobrazenie lišty kariet, zobrazenie navigácie a obrázky, ktoré prekrývajú hlavné View. Toto všetko je vnorené do UIWindow. Ovládacie prvky zahrnujú napríklad UILabel alebo UITextField a ďalšie užívateľské rozhrania. Tieto ovládacie prvky sa nachádzajú v triede view, ktorá sa nazýva UIView. Táto trieda sa dedí, pokiaľ chceme vytvoriť View, ktoré je viazané na dáta, ako je napríklad UICollectionView pre mriežky a zoskupenie. IOS layouty sú vytvorené pomocou AutoLayout, čo je technika založená na obmedzeniach medzi Views, ktoré sa dynamicky pohybujú a menia veľkosť v závislosti od kontextu zobrazenia. Toto všetko je súčasťou UIKitu, vývojového frameworku iOS UI. Máme dva spôsoby vytvorenia obrazovky v iOSe. Prvým je ručné písanie kódu v C#, pokiaľ chceme využiť dynamické dátové viazanie a tok dát medzi stránkami a druhým, viac pohodlnejším, je pomocou použitia nástroja pre dizajnérov, napríklad ako Xamarin Designer. Tieto nástroje vytvárajú XML súbor alebo .xib a ručné písanie kódu sa prevádza priamo v iOS view-controllere triedy C# [21].

4 .NET

.NET je open-source a multiplatformová platforma používaná na vývoj aplikácií. Pomocou .NETu môžeme používať viacero vývojových jazykov, knižníc a editorov na vývoj aplikácií. Z jazykov tu používame C#, F# a Visual Basic [22].

4.1 .NET Core

.NET Core vyšiel v roku 2016 ako nová multiplatformová verzia frameworku, ktorý zahrňuje multiplatformovú implementáciu Common Language Runtime (CLR) nazvanú ako CoreCLR. Posledná verzia .NET Core 3.1 bola vydaná v roku 2019, odvtedy ho Microsoft premenoval na .NET. Ďalšia verzia bola nazvaná už len ako .NET 5.0, ktorý obsahoval ASP.NET Core, Xamarin a Entity Framework Core. Microsoft týmto chcel zjednotiť .NET ekosystém a tým najnovšia 6. verzia pokračuje v zjednotení platformy [23].

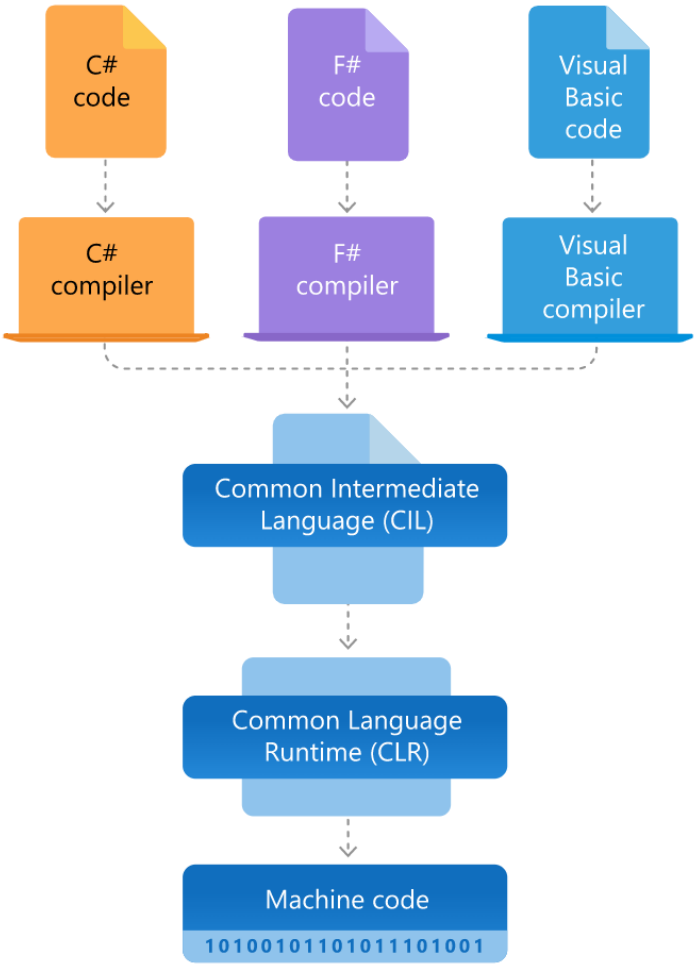
Version	Released	Edition	Published
.NET Core RC1	November 2015	First	March 2016
.NET Core 1.0	June 2016		
.NET Core 1.1	November 2016		
.NET Core 1.0.4 and .NET Core 1.1.1	March 2017	Second	March 2017
.NET Core 2.0	August 2017		
.NET Core for UWP in Windows 10 Fall Creators Update	October 2017	Third	November 2017
.NET Core 2.1 (LTS)	May 2018		
.NET Core 2.2 (Current)	December 2018		
.NET Core 3.0 (Current)	September 2019	Fourth	October 2019
.NET Core 3.1 (LTS)	December 2019		
Blazor WebAssembly 3.2 (Current)	May 2020		
.NET 5.0 (Current)	November 2020	Fifth	November 2020
.NET 6.0 (LTS)	November 2021	Sixth	November 2021
.NET 7.0 (Current)	November 2022	Seventh	November 2022
.NET 8.0 (LTS)	November 2023	Eighth	November 2023

Obrázok 12 Tabuľka verzií .NET [23]

Z obrázku môžeme vidieť jednotlivé verzie a ich dátumy vydania alebo budúceho vydania, ktoré nás ešte len čakajú.

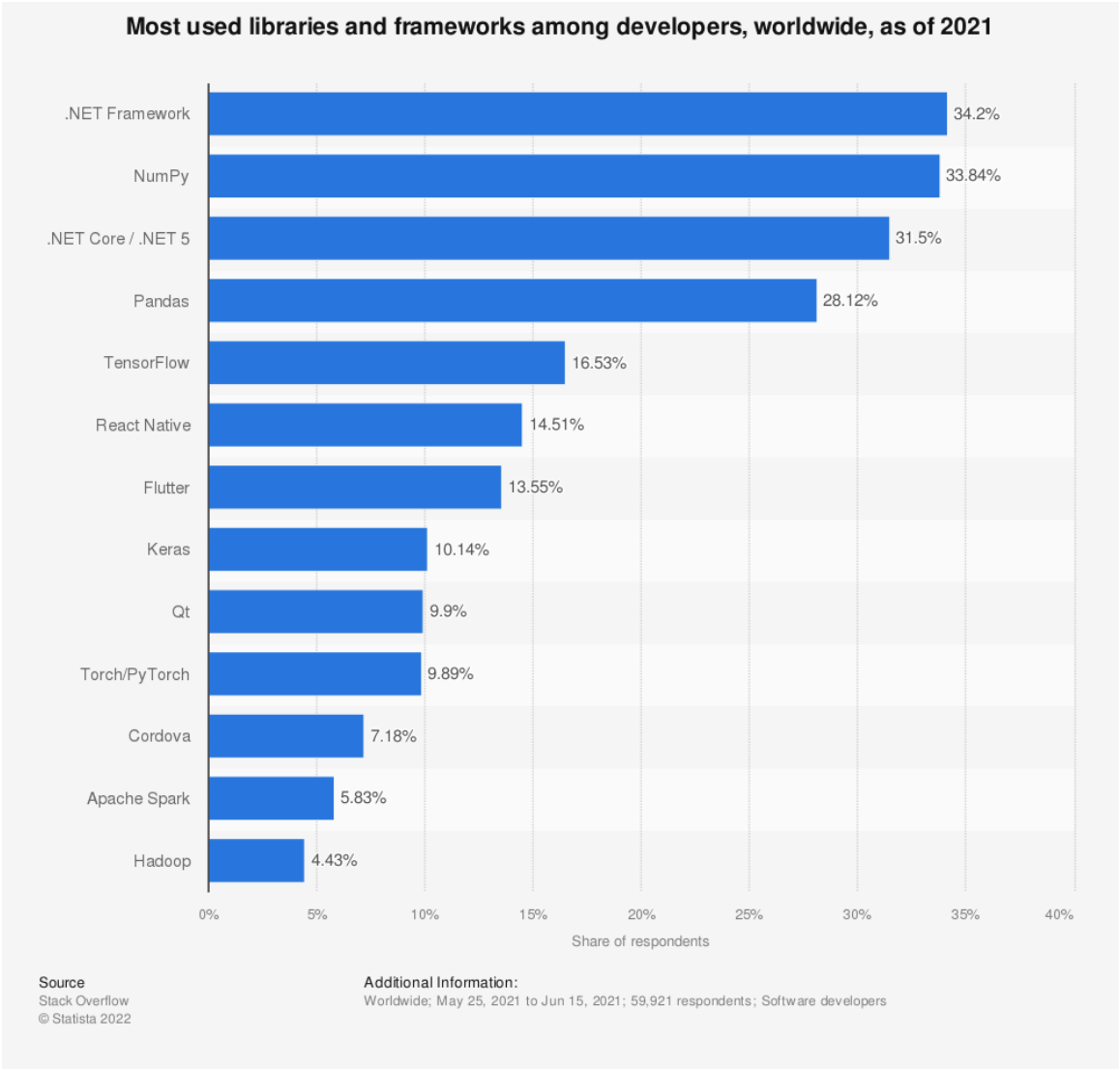
4.2 .NET Framework

.NET Framework je vývojová platforma, ktorá obsahuje Common Language Runtime (CLR) a .NET Framework Class Library. CLR je spúšťací modul, ktorý spracováva spustené aplikácie a poskytuje služby ako napríklad spracovanie výnimok, správu vlákien, verifikáciu bezpečnosti kódu a iné systémové funkcie [24]. Bezpečnosť je veľmi dôležitá, a tak spravovaným súčastiam je pridelený stupeň dôvery, ktorý závisí napríklad od ich pôvodu ako internet, podniková sieť alebo lokálny počítač. To má za následok, že spravovaná súčasť alebo komponent môže alebo nemusí mať prístup k niektorým operáciám ako prístup k súborom alebo prístup do registrov, aj keď sa používa v rovnakej aktívnej aplikácii. .NET Framework Class Library je objektovo orientovaná a poskytuje sadu rozhraní API pre funkcie, čítanie a zápis súborov, pripojenie k databáze. Taktiež poskytuje typy pre reťazce, dáta a čísla [23].



Obrázok 13 Architektúra .NET Framework [24]

Za spomenutie určite stojí nasledujúci štatistický graf, ktorý znázorňuje použitie knižníc a frameworkov celosvetovo za rok 2021. Na ňom môžeme vidieť, že .NET Framework je najpoužívanejší framework a za ním je tesne NumPy, čo je knižnica jazyka Python [25].

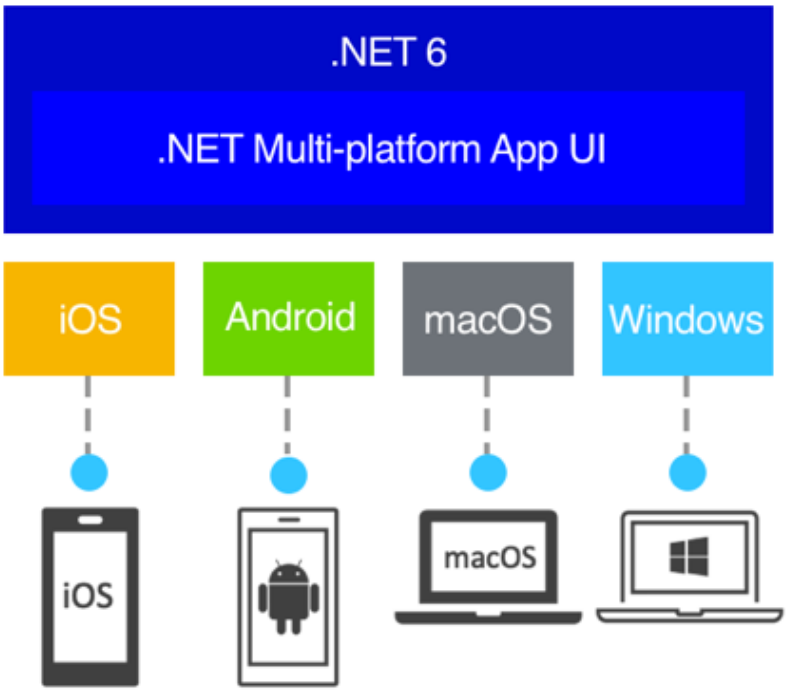


Obrázok 14 Najpoužívanejšie frameworky a knižnice [25]

4.3 .NET MAUI

.NET MAUI alebo celý názov Multi-platform App User Interfaces je open-source a multiplatformový framework pre vytváranie natívnych mobilných a desktopových aplikácií pomocou jazyka C# a XAML. Môžeme vyvíjať aplikácie pre Android, iOS, Windows alebo macOS z jedného zdieľaného kódového základu. Microsoft týmto chcel vytvoriť akési pokračovanie a vylepšenie Xamarin.Forms tak, že ho rozšíril o možnosť vyvíjať desktopové aplikácie s ovládacími prvkami UI prestavanými pre lepší výkon a rozšíriteľnosť[26].

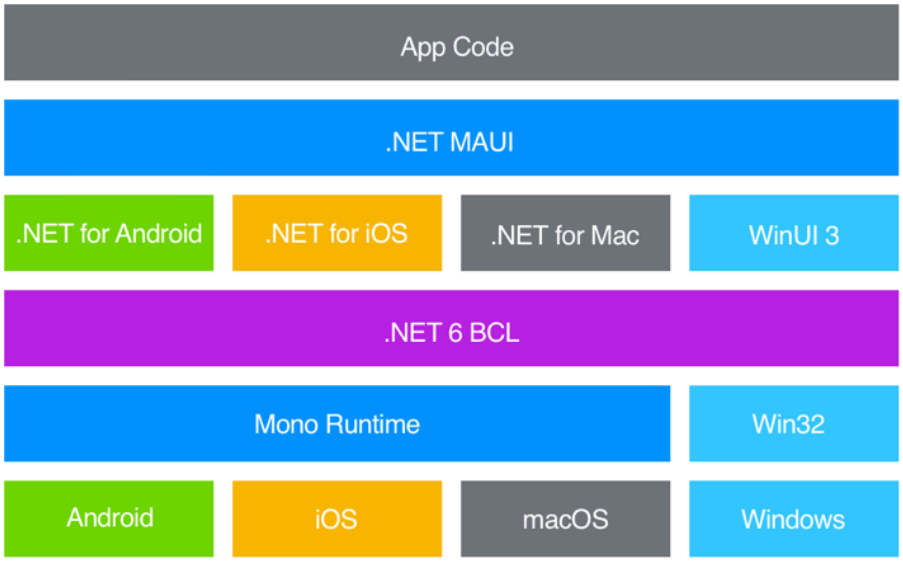
Xamarin.Forms a .NET MAUI sú si veľmi podobné, ale existujú aj určité rozdiely, ktoré si nižšie vysvetlíme a popíšeme.



Obrázok 15 .NET MAUI [26]

.NET MAUI pracuje tak, že zjednocuje rozhrania Android, iOS, Windows a macOS do jedného rozhrania API, ktorý umožňuje takzvané write-once run-everywhere, čo je pri voľnom preklade jedenkrát napíš a pusti kdekoľvek. Navyše poskytuje väčší prístup ku každej časti natívnej knižnice. Taktiež poskytuje jednotný framework pre vytváranie užívateľských rozhraní pre mobilné aj desktopové aplikácie [26].

V .NET MAUI píšeme kód, ktorý pracuje primárne z .NET MAUI API rozhraním. Potom priamo používa API natívnej platformy a tým môže priamo využívať toto rozhranie danej platformy [26].



Obrázok 16 .NET MAUI architektúra[26]

Aplikácie pre Android, ktoré vytvoríme pomocou .NET MAUI, sa kompilujú zo C# do takzvaného intermediate language (IL), ktorý je najnižší čitateľný programovací jazyk používaný kompilátormi [27]. Ďalej sa skompilujú do natívneho zostavenia pri spustení aplikácie [26].

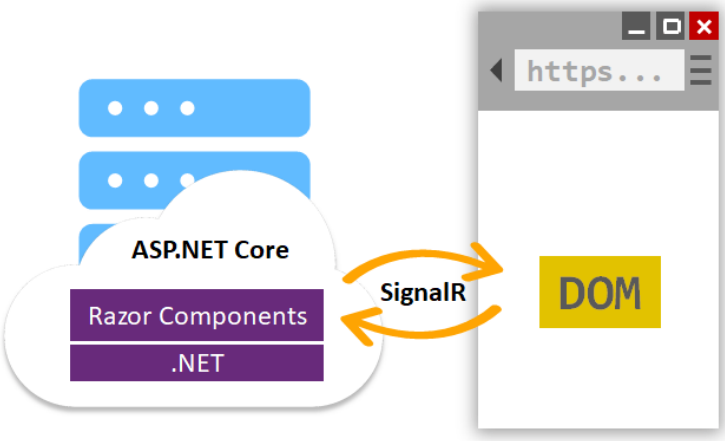
iOS Aplikácie sú zase plne kompilované ahead-of-time zo C# do natívného kódu. Ahead-of-time znamená kompiláciu, kde je kód vyšších jazykov prevedený do natívného strojového kódu tak, aby výsledný binárny súbor mohol byť sputený natívne [26].

4.4 Blazor

Za zmienku určite stojí Blazor, ktorý je framework určený na vytváranie interaktívného webového užívateľského rozhrania na strane klienta. Blazor obsahuje niekoľko modelov hostingu. To znamená, že s dobrým plánovaním môžeme napísať Blazor komponenty iba raz a potom ich spustiť na strane webového serveru, webového klienta alebo v rámci desktopovej aplikácie [28].

Blazor server

Blazor server beží na strane serveru, takže kód napísaný v C# má plný prístup ku všetkým prostriedkom, ktoré by potreboval bez potreby overovania. Ku komunikácii s užívateľským rozhraním na strane klienta používa SignalR. Server musí udržovať stále pripojenie SignalR ku každému klientovi a sledovať ich stav [23].

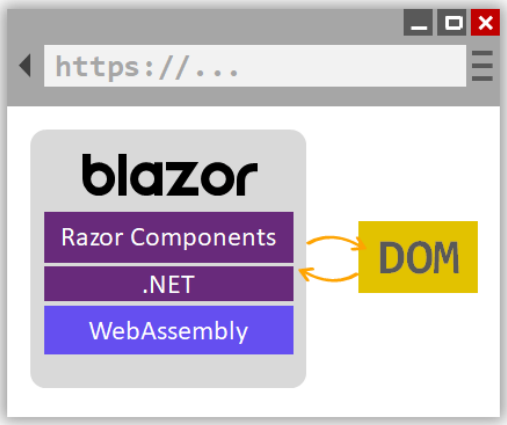


Obrázok 17 Blazor server[28]

DOM alebo Document Object Model je dátová reprezentácia objektov, ktoré tvoria štruktúru a obsah dokumentu na webe [28].

Blazor WebAssembly

Blazor WebAssembly beží na strane klienta, takže kód napísaný v C# má plný prístup k zdrojom iba v prehliadači a musí prevádzať volanie HTTP, predtým ako mu bude umožnený prístup k zdrojom na serveri [23].



Obrázok 18 Blazor WebAssembly [28]

Blazor Hybrid

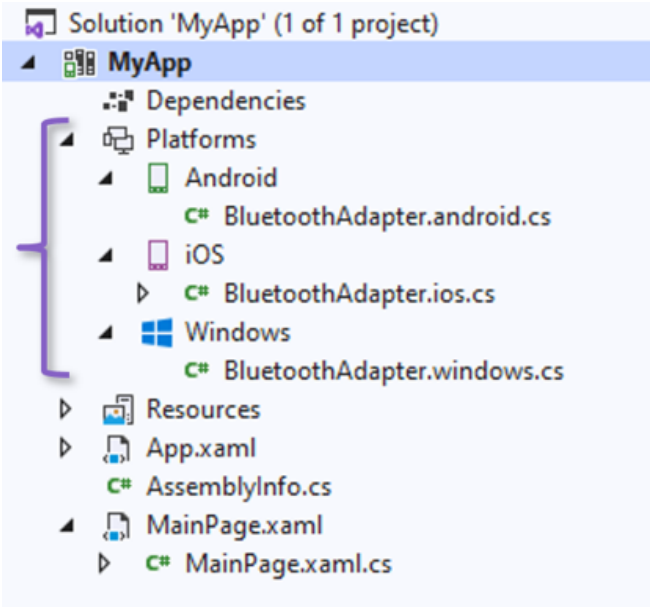
Aplikácie postavené na Blazor Hybrid používajú Blazor v natívnej klientskej aplikácii. Blazor Hybrid beží natívne v procese .NET a vykresľuje webové užívateľské rozhranie do ovládacieho prvku webového zobrazenia pomocou miestneho interop kanálu [23].

5 POROVNANIE .NET MAUI A XAMARIN.FORMS

Vyššie sme si uvádzali, že Xamarin.Forms a .NET MAUI sú multiplatformové a opensource frameworky pre vývoj aplikácií s jedným kódovým základom, ktorý je napísaný v C#. Microsoft príchodom .NETu MAUI sľubuje lepšiu spoluprácu s nástrojmi a službami od Microsoftu a takiež lepší výkon. Ak vyvíjame v .NETe, musíme prejsť rôznymi variantmi, ako .NET Framework, .NET Core alebo Mono. Ako bolo zmienené, toto vyriešil Microsoft vydaním .NET 5, kde zjednotil všetky tieto varianty. Týmto zjednotením sme získali, že sa Xamarin sa stane súčasťou platformy .NET. Čiže spolu je to súčasť ekosystému, ktorý má podporu od tímov pracujúcich na .NET platforme a momentálne prioritizujú mobilný vývoj [29].

Pri vyvíjaní aplikácie očakávame, že výsledok zmien v kóde, ktoré spravíme, sa prejaví okamžite. Pri editácii potrebujeme mať možnosť okamžite vidieť a skontrolovať tieto zmeny. NET MAUI pridáva túto funkciu, ktorá sa nazýva hot reload. Vďaka tejto funkcii máme možnosť vidieť zmeny, ktoré spravíme okamžite v bežiacej aplikácií bez toho, aby sme ju museli znovu zostavovať [26].

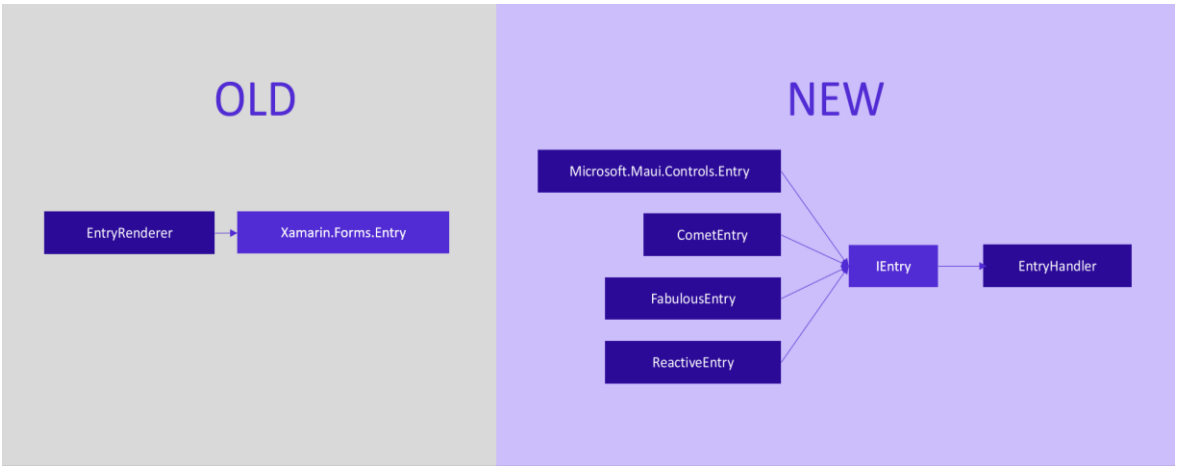
Štruktúra multiplatformovej aplikácie, ktorá je vytvorená pomocou Xamarin.Forms, sa skladá z jedného projektu pre platformu a samostatných súborov projektu pre každú platformu. Pokiaľ chceme vyvíjať mobilnú aplikáciu, napríklad pre Android alebo iOS, je potrebné vytvoriť a pracovať s tromi rôznymi súbormi projektu [29].



Obrázok 19 MAUI Projektové súbory

V .NETe MAUI ale pracujeme v rámci jedného rozhrania a projektového súboru, kde je možné písať kód multiplatforne alebo pre špecifickú platformu [29].

Je bežné, že aplikácie vyvíjané pomocou Xamarinu môžu mať výkonnostné problémy, čo sa týka doby načítania aplikácie. Jedným dôvodom, prečo sa to deje, je vykresľovacia architektúra, ktorá je zabudovaná v Xamarin.Forms už nejaký čas. V .NETe MAUI tento problém rieši funkcia nazvaná "handlers" [29]. Táto nová architektúra odstraňuje View wrapping, čím sa znižuje počet ovládacích prvkov užívateľského rozhrania potrebných na vykreslenie pohľadu [30].



Obrázok 20 MAUI Handlers[30]

V Xamarin.Forms má každý vykresľovač odkaz na multiplatformový prvok. Miesto týchto vykresľovačov má .NET MAUI nový vzor „handler" [30].

Najnovšia verzia Xamarin.Forms 5.0 bola vydaná minulý rok a už neobdrží žiadne nové funkcie, ale Microsoft bude riešiť chyby a vydávať servisné updaty do novembra roku 2022. Čiže Xamarin.Forms bude končiť. Microsoft však sľúbil jednoduchú migráciu tak, že nebude potrebné prepisovať už existujúce Xamarin.Forms aplikácie, aby mohli prejsť do .NETu MAUI [29] [35].

.NET MAUI však nie je jediný vývoj Xamarin.Forms, teraz môžeme vyvíjať aj webové aplikácie pomocou frameworku Blazor. Tento framework slúži pre vytváranie interaktívneho užívateľského rozhrania na strane klienta s .NETom. Taktiež MAUI umožňuje vytváranie hybridných aplikácií práve pomocou Blazoru, čo je veľká výhoda pre webových vývojárov, a tak môžu vytvárať užívateľské rozhrania s HTML a CSS a tiež majú prístup k funkciám zariadenia, aby pracovali ako natívne aplikácie [29].

5.1 Hlavné výhody .NET MAUI oproti Xamarin.Forms

1. Práca v jednom projektovom súbore s .NET CLI

.NET MAUI umožňuje prácu len v jednom projektovom súbore, ale aj tak niektoré špecifické operácie pre danú platformu musíme previesť. Taktiež prináša podporu pre nástroje .NET CLI pre bezproblémové vytváranie, vývoj a spustenie .NET aplikácií [31].

2. Moderné vzory

.NET MAUI podporuje MVU (model-view-update) a Blazor vývojové vzory. Vzor MVU poskytuje jednotný spôsob, ako vytvoriť viaceré multiplatformové natívne frontendy z jedného základného kódu. Taktiež umožňuje písanie užívateľského rozhrania a logiku v C#. Rozširuje použitie Blazoru tak, aby jeho cieľové scenáre zahrňovali natívne desktopové aplikácie[31]. Taktiež podporuje Dependency Injection a jeho výhody [34].

3. Podpora hot reload

Plná podpora hot reload je obsiahnutá v .NETe MAUI

	XAML Hot Reload	.NET Hot Reload
Xamarin.Forms	**Experimental:** SDK 4.x & Visual Studio 2019 prior to version 16.9 **Feature Complete:** SDK 5.x & Visual Studio 2019 version 16.9 or newer	iOS/Android : No UWP : Limited support for runtime edits using .NET's Edit and Continue feature.
.NET MAUI	Complete support	Complete Support

Obrázok 21 Tabuľka podpory hot reload [31]

Na obrázku môžeme vidieť, akú má hot reload momentálnu podporu u .NET MAUI a Xamarin.Forms.

4. Spojenie knižníc

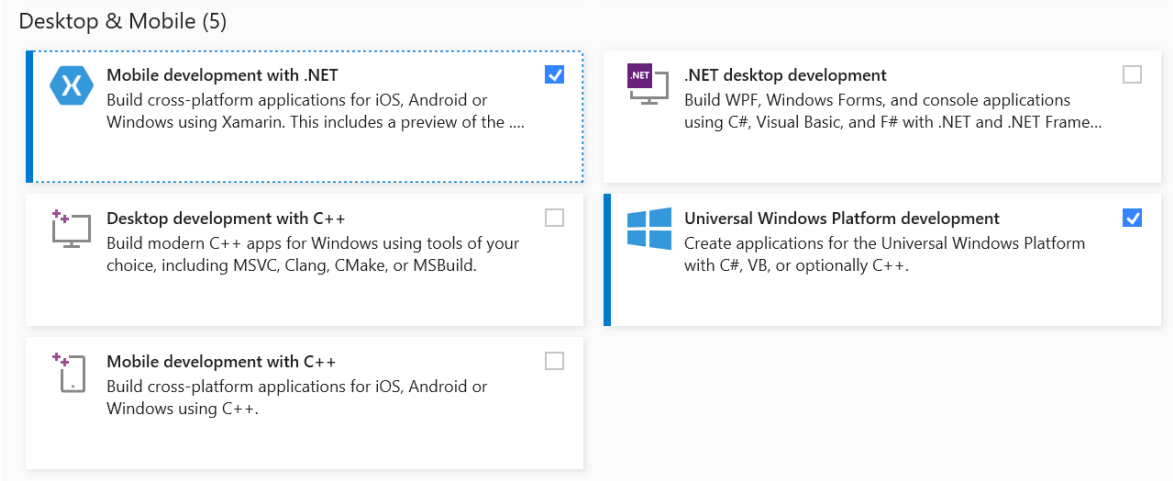
.NET MAUI prichádza so zjednotením knižníc. Tým poskytuje hneď niekoľko výhod. Zlúčením knižnice Xamarin.Essentials môžeme jednoducho používať funkcie zariadenia ako napríklad prístup ku kamere, kontaktom alebo jeho senzorom [31].

II. PRAKTICKÁ ČÁST

6 PRÍPRAVA NA PROGRAMOVANIE V .NET MAUI

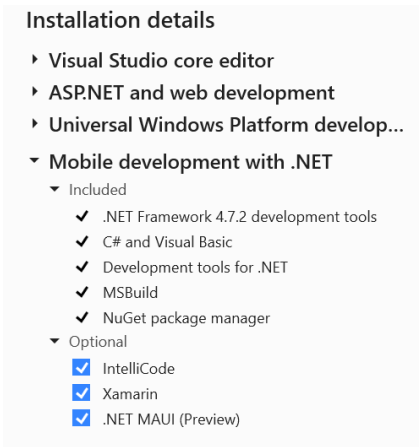
Ak chceme programovať pomocou frameworku .NET MAUI, budeme potrebovať stiahnuť najnovšiu verziu Visual Studia 2022, ktorá je zatiaľ len vo verzii preview, a tiež najnovší .NET 6.

Po stiahnutí a inštalácii Visual Studio Instalerru v záložke Workloads musíme zvoliť dôležité workloady,, aby sme mohli funkčne programovať pomocou NETu MAUI.



Obrázok 22 Workloads

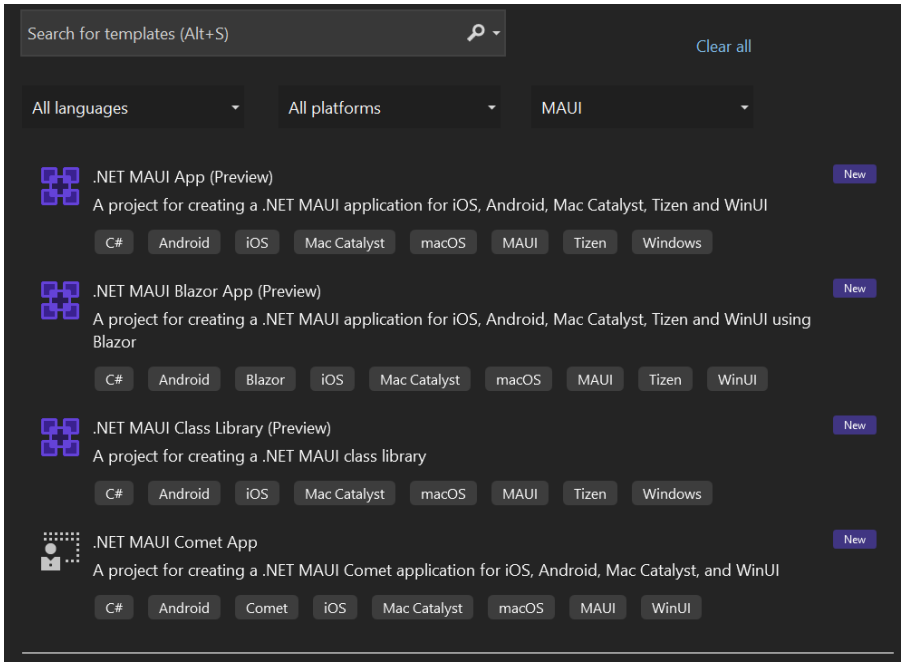
Workloady, ktoré potrebujeme, si zaklikneme a zobrazí sa nám pri nich modrobiela fajka (Obr. 22). Na pravej strane obrazovky potrebujeme zvoliť .NET MAUI (Preview). Tento krok je tiež veľmi dôležitý pre správne fungovanie (Obr. 23).



Obrázok 23 Details

Ďalej sa na nám zobrazia Individual components, všetky komponenty, ktoré potrebujeme už sú automaticky zvolené. Vyberieme si jazyk, odporúčaný je angličtina a dáme inštalovať.

Po inštalácii spustíme Visual Studio 2022, klikneme na “Create a new project” a zvolíme typ projektu MAUI. Zobrazí sa nám okno s možnosťami, akú aplikáciu chceme vytvárať (Obr. 24). Vyberieme, pomenujeme projekt, zvolíme, kde ho chceme uložiť a vytvorí sa nám nový projekt.



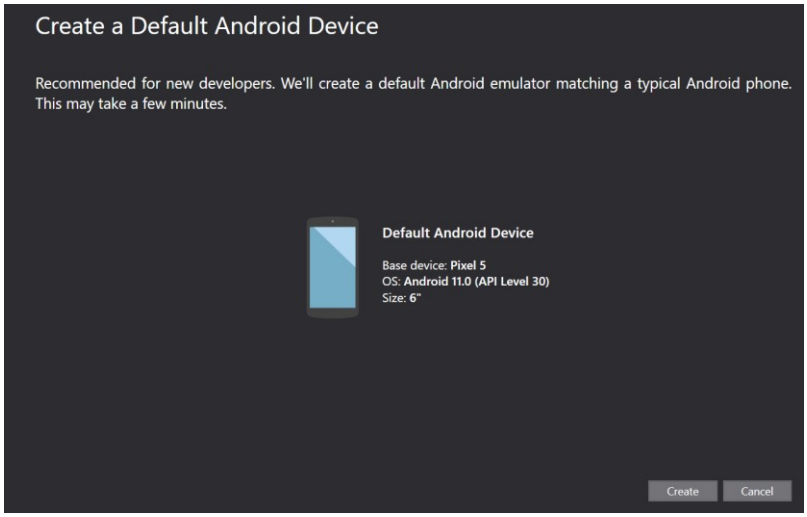
Obrázok 24 Vytvorenie projektu

My sme si zvolil prvú možnosť .NET MAUI App(Preview) na aplikáciu, ktorú si predstavíme nižšie. Používali sme aj Android Emulátor, ktorý je vstavaný do VS a nižšie uvedieme, ako ho spustiť.

Ako prvé vidíme v hornej lište Android Emulator, na toto tlačidlo klikneme (Obr. 25).

Obrázok 25 Android Emulator tlačidlo

Po kliknutí sa nám zobrazí dialógové okno, kde potvrdíme súhlas aplikácie o vytváraní zmien v zariadení. Ďalej sa nám zobrazí okno o vytvorení defaultného android zariadenia, ktoré nám bude úplne postačovať a klikneme na vytvoriť (Obr. 26).



Obrázok 26 Vytvorenie zariadenia

Po vytvorení spustíme Android Emulátor, ktorému to chvíľu trvá pri prvom spustení a môžeme začať programovať.

Pre podrobnejší opis inštalácie odporúčam navštíviť oficiálnu stránku Microsoftu [39].

7 POROVNANIE APLIKÁCIÍ

V praktickej časti si predstavíme riešenia aplikácie, ktoré budú napísané pomocou frameworkov .NET MAUI a Xamarin.Forms. Popíšeme si, ako fungujú, čo robia a aký je rozdielny kód medzi týmito dvomi frameworkami.

Funkčné požiadavky:

- používateľ zadá svoju váhu a hmotnosť,
- aplikácia mu zobrazí výsledný BMI index.

Nefunkčné požiadavky:

- aplikácia je multiplatformová.

7.1 BMI ukazovateľ v .NET MAUI

Najprv sa zameriame na aplikáciu BMI ukazovateľ. Používateľ si pomocou slidera nastaví svoju výšku a hmotnosť a následne je mu v reálnom čase ukázaný jeho BMI index aj s popisom jeho váhy.

7.1.1 MainPage.xaml

V MainPage máme styles a nabindované source. Začali sme s tým, že sme si nastavili farby a pomenovali ich, aby sme ich nižšie mohli ľahko používať. Ďalej sme definovali štýly lablu, priradili sme k ním kľúč, aby sme ich mohli identifikovať. Nastavili sme horizontálne zarovnanie

```
<ContentPage
    x:Class="BMI_app.MainPage"
    xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    BackgroundColor="{DynamicResource SecondaryColor}">

    <ContentPage.Resources>
        <Color x:Key="SliderColor">#003459</Color>
        <Color x:Key="TitleColor">#1E48B8</Color>
        <Color x:Key="LabelColor">#00171f</Color>
```

Definovali sme štýl lablu, priradili kľúč, aby sme ho mohli identifikovať, nastavili horizontálne zarovnanie a farbu, ktorú sme vytvorili vyššie. Tento štýl nebudeme aplikovať, ale poslúži nám ako základ pre ostatné štýly.

```
        <Style x:Key="StyleLabel" TargetType="Label">
            <Setter Property="HorizontalOptions" Value="Center" />
            <Setter Property="TextColor" Value="{StaticResource LabelColor}"
/>
```

```
        </Style>
```

Tento štýl slúži pre label, ktorý vyzve užívateľa na zadanie výšky a bude dediť od štýlu, ktorý sme vytvorili vyššie. Nastavíme len veľkosť textu a padding.

```
        <Style
            x:Key="TitleStyle"
            BasedOn="{StaticResource StyleLabel}"
            TargetType="Label">
            <Setter Property="FontSize" Value="Title" />
            <Setter Property="Padding" Value="30" />
        </Style>
```

Tento štýl je pre výpis hodnôt zo slideru, nastavíme farbu a veľkosť textu a tiež dedí od štýlu, ktorý sme vytvorili ako prvý.

```
      <Style
            x:Key="ValueStyle"
            BasedOn="{StaticResource StyleLabel}"
            TargetType="Label">
            <Setter Property="FontSize" Value="Title" />
            <Setter Property="TextColor" Value="{StaticResource TitleColor}"
/>
        </Style>
```

Nakoniec štýl pre samotný slider s nastavením farby.

```
        <Style TargetType="Slider">
            <Setter Property="ThumbColor" Value="{StaticResource SliderColor}"
/>
            <Setter Property="MinimumTrackColor" Value="{StaticResource
SliderColor}" />
            <Setter Property="MaximumTrackColor" Value="{StaticResource
SliderColor}" />
        </Style>

    </ContentPage.Resources>
```

Na to, aby sme zobrazili hodnoty zo slideru, použijeme data binding. To znamená, že pomocou Binding Source nastavíme, aby zdrojové hodnoty pochádzali zo zvoleného slideru. Taktiež tu nastavujeme zobrazenie na 2 desatinné miesta, maximum a minimum.

```
    <FlexLayout
        Padding="40"
        Direction="Column"
        JustifyContent="SpaceEvenly">
        <StackLayout>
            <Label Style="{DynamicResource TitleStyle}" Text="How tall are you
?" />
            <Label Style="{DynamicResource ValueStyle}" Text="{Binding
Source={x:Reference SliderH}, Path=Value, StringFormat='{0:F0} cm'}" />
            <Slider
                x:Name="SliderH"
                Maximum="240"
                Minimum="40"
```

```
                Value="{Binding Height}" />
        </StackLayout>
```

Taktiež ako v predchádzajúcom prípade pomocou Binding Source nastavíme, aby zdrojové hodnoty pochádzali zo zvoleného slideru a nastavujeme zobrazenie na 2 desatinné miesta, maximum a minimum.

```
        <StackLayout>
            <Label Style="{DynamicResource TitleStyle}" Text="How much are you
weight?" />
            <Label Style="{DynamicResource ValueStyle}" Text="{Binding
Source={x:Reference SliderW}, Path=Value, StringFormat='{0:F0} kg'}" />
            <Slider
                x:Name="SliderW"
                Maximum="320"
                Minimum="30"
                Value="{Binding Weight}" />
        </StackLayout>
```

Pridáme ovládacie prvky pre výpis výpočtu BMI spolu s popisom klasifikácie.

```
      <StackLayout>
            <Label Style="{DynamicResource StyleLabel}" Text="Your BMI is: "
/>
            <Label
                FontSize="48"
                Style="{DynamicResource StyleLabel}"
                Text="{Binding Bmi}" />
            <Label Style="{DynamicResource StyleLabel}" Text="{Binding Classi-
fication}" />
        </StackLayout>
    </FlexLayout>
</ContentPage>
```

7.1.2 MainPageViewModel.cs

Aby sme mohli riadiť užívateľské rozhranie, vytvorili sme si ViewModel. ViewModel je trieda, ktorá reprezentuje stav a správanie užívateľského rozhrania.

Ako prvé si nalinkujeme tento ViewModel do View tak, že v MainPage.xaml.cs pridáme “BindingContext = new MainPageViewModel();” .

Implementujeme interface InotifyPropertyChanged, to pomáha upozorniť UI na zmenu hodnoty property. Ďalej pridáme properties, ktoré reprezentujú stav UI. Pridáme konštantu na to, aby hodnota slideru išla po 1.

```
internal class MainPageViewModel : INotifyPropertyChanged
    {
        private double height = 150;
        private double weight;
        private const double Step = 1.0;
```

Pri Height aj Weight uložíme do premennej hodnotu, ktorú sme dostali od užívateľa a následne voláme metódu UpdateBMI na jeho výpočet.

```
public double Height {
    get => height;
    set
    {
        height = NextStep(value);
        UpdateBMI();
    }
}

public double Weight {
    get => weight;
    set
    {
        weight = NextStep(value);
        UpdateBMI();
    }
}
```

Výpočet BMI so vzorcom, ktorý nám vracia zaokrúhlený výsledok a používa výšku a váhu ako vstup.

```
public double Bmi => Math.Round(Weight / Math.Pow(Height / 100, 2),
2);
```

Classification property slúži na klasifikáciu výpočtu Bmi a priradí pomocou podmienok, o akú osobu sa jedná.

```
public string Classification
{
    get
    {
        if (Bmi < 18.5)
            return "You are underweight";
        else if (Bmi < 25)
            return "You have normal weight";
        else if (Bmi < 30)
            return "You are overweight";
        else
            return "You are obese";
    }
}
```

V tejto metóde voláme pomocnú metódu, aby sme informovali UI o tom, že sa Bmi a klasifikácia zmenila.

```
private void UpdateBMI()
{
    RaisePropertyChanged(nameof(Bmi));
    RaisePropertyChanged(nameof(Classification));
}
```

Metóda, ktorá vypočítava ďalší krok slideru od užívateľa.

```
        private double NextStep(double value) => Math.Round(value / Step) *
Step;

        public event PropertyChangedEventHandler PropertyChanged;
```

Pomocná metóda, ktorá vyvoláva Property Change event na žiadosť, aby sme mohli akutalizovať UI.

```
        private void RaisePropertyChanged(string propertyName)
        {
            PropertyChanged?.Invoke(this,new PropertyChangedEventArgs(proper-
tyName));
        }
    }
```

7.2 BMI ukazovateľ v Xamarin.Forms

Taktiež rovnako ako vyššie sme vytvorili BMI aplikáciu, ale vo frameworku Xamarin.Forms

7.2.1 MainPage.xaml

V kóde, ktorý sa nachádza nižšie, môžeme vidieť, že MainPage.xaml je totožná s MainPage, ktorú sme už vytvorili, ale vo frameworku .NET MAUI. Nachádzajú sa tu rozdiely, ale len vo farbách textu a slideru.

```
<ContentPage
    x:Class="BMIapp2.MainPage"
    xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml">
    <ContentPage.Resources>
        <Color x:Key="SliderColor">#003459</Color>
        <Color x:Key="TitleColor">#1E48B8</Color>
        <Color x:Key="LabelColor">#00171F</Color>

        <Style x:Key="LabelStyle" TargetType="Label">
            <Setter Property="HorizontalOptions" Value="Center" />
            <Setter Property="TextColor" Value="{StaticResource LabelColor}"
/>
        </Style>

        <Style
            x:Key="TitleStyle"
            BasedOn="{StaticResource LabelStyle}"
            TargetType="Label">
            <Setter Property="FontSize" Value="Large" />
        </Style>

        <Style
            x:Key="ValueStyle"
            BasedOn="{StaticResource LabelStyle}"
            TargetType="Label">
            <Setter Property="FontSize" Value="Large" />
            <Setter Property="TextColor" Value="{StaticResource TitleColor}"
/>
        </Style>
```

```
        <Style TargetType="Slider">
            <Setter Property="ThumbColor" Value="{StaticResource SliderColor}"
/>
            <Setter Property="MinimumTrackColor" Value="{StaticResource
SliderColor}" />
            <Setter Property="MaximumTrackColor" Value="{StaticResource Title-
Color}" />
        </Style>
    </ContentPage.Resources>

    <FlexLayout
        Padding="40"
        Direction="Column"
        JustifyContent="SpaceEvenly">
        <StackLayout>
            <Label Style="{StaticResource TitleStyle}" Text="How tall are
you?" />
            <Label Style="{StaticResource ValueStyle}" Text="{Binding
Source={x:Reference HeightSlider}, Path=Value, StringFormat='{0:F0} cm'}" />
            <Slider
                x:Name="HeightSlider"
                Maximum="240"
                Minimum="40"
                Value="{Binding Height}" />
        </StackLayout>

        <StackLayout>
            <Label Style="{StaticResource TitleStyle}" Text="How much do you
weight?" />
            <Label Style="{StaticResource ValueStyle}" Text="{Binding
Source={x:Reference WeightSlider}, Path=Value, StringFormat='{0:F0} kg'}" />
            <Slider
                x:Name="WeightSlider"
                Maximum="320"
                Minimum="30"
                Value="{Binding Weight}" />
        </StackLayout>

        <StackLayout>
            <Label Style="{StaticResource LabelStyle}" Text="Your BMI is: " />
            <Label
                FontSize="48"
                Style="{StaticResource LabelStyle}"
                Text="{Binding Bmi}" />
            <Label Style="{StaticResource LabelStyle}" Text="{Binding Classi-
fication}" />
        </StackLayout>
    </FlexLayout>
</ContentPage>
```

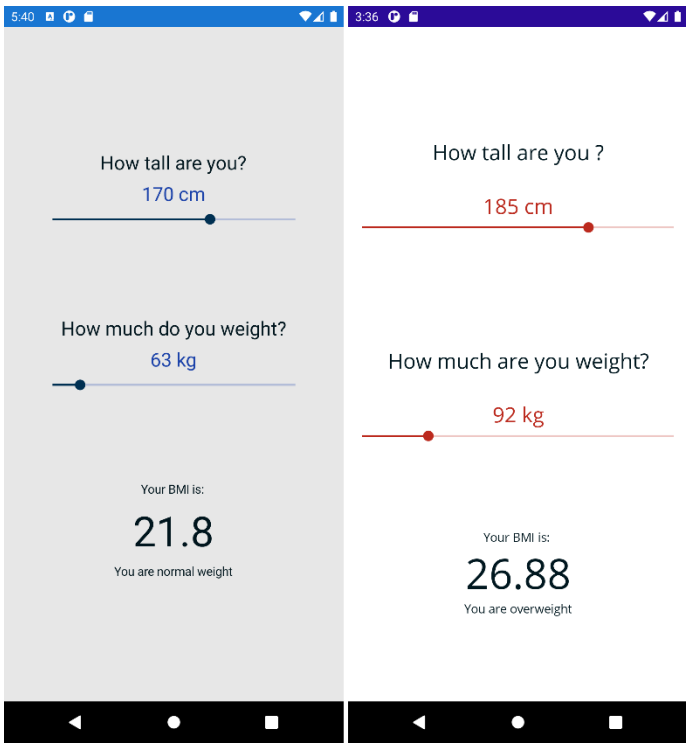
7.2.2 MainPageViewModel.cs

Vytvorili sme si tiež ViewModel, aby sme mohli riadiť užívateľské rozhranie a nalinkovali si ho do View v MainPage.xaml.cs. ViewModel je totožný s tým, ktorý sme vytvorili v .NETe MAUI.

```
namespace BMIapp2
{
    internal class MainPageViewModel : INotifyPropertyChanged
    {
        private double height = 150;
        private double weight;
        private const double Step = 1.0;
        public double Height
        {
            get => height;
            set
            {
                height = NextStep(value);
                UpdateBMI();
            }
        }
        public double Weight
        {
            get => weight;
            set
            {
                weight = NextStep(value);
                UpdateBMI();
            }
        }
        public double Bmi => Math.Round(Weight / Math.Pow(Height / 100, 2),2);
        public string Classification
        {
            get
            {
                if (Bmi < 18.5)
                    return "You are underweight";
                else if (Bmi < 25)
                    return "You are normal weight";
                else if (Bmi < 30)
                    return "You are overweight";
                else
                    return "You are obese";

            }
        }
        private void UpdateBMI()
        {
            RaisePropertyChanged(nameof(Bmi));
            RaisePropertyChanged(nameof(Classification));
        }
        private double NextStep(double value) => Math.Round(value / Step) *
Step;

        public event PropertyChangedEventHandler PropertyChanged;

        private void RaisePropertyChanged(string propertyName)
        {
            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(proper-
tyName));
        }
    }
}
```

7.3 Porovnanie BMI aplikácií

Z funčného hľadiska môžeme povedať, že tieto aplikácie fungujú rovnako. Vo funkčnom kóde nevidíme takmer žiadne rozdiely. MainPageViewModel majú obe aplikácie totožné. Z toho výzorového sú drobné rozdiely ako farba písma, slideru alebo odskočenie textu.



Obrázok 27 BMI aplikácie

Na obrázku môžeme vidieť, ako aplikácie vyzerajú v bežiacom stave. Na prvom mieste je aplikácia vytvorené pomocou Xamarin.Forms a na druhom aplikácia .NET MAUI.

7.4 Dependency Injection v BMI aplikácii

Aplikáciu, ktorú sme vyššie ukazovali, sme upravili pomocou vzorov Dependency Injection, ktoré sú v .NETe MAUI dostupné. Predtým sme ich v Xamarin.Forms nemohli používať, pretože neboli podporované, no s príchodom .NETu MAUI ich môžeme naplno využívať.

7.4.1 MauiProgram.cs

Ako prvé sme pridali v MauiProgram.cs Services aby sme mohli používať vzor Singleton. Takže sme najskôr registrovali MainPage ako singleton a neskôr aj ViewModel pre MainPage.

```
public static class MauiProgram
{
    public static MauiApp CreateMauiApp()
    {
        var builder = MauiApp.CreateBuilder();
        builder
            .UseMauiApp<App>()
            .ConfigureFonts(fonts =>
            {
                fonts.AddFont("OpenSans-Regular.ttf", "OpenSansRegular");
            });
        builder.Services.AddSingleton<MainPage>();
        builder.Services.AddSingleton<MainPageViewModel>();

        return builder.Build();
    }
}
```

7.4.2 App.xaml.cs

V App.xaml.cs sme pridali do konštruktoru MainPage a nastavíme MainPage ako page, keďže používame Singleton, tak nechceme, aby sa nám vytvárala stále nová MainPage. Zaregistrovali sme našu MainPage do kontajnera Dependency Injection. V programe MAUI je všetko prepojené s host builderom, pretože už registrujeme MianPage v predchádzajúcom kóde, tak to automaticky uvidí, či sa daná MainPage už nachádza v MAUI kontajnere. Ak sa nachádza, čo je aj náš prípad, vloží závislosť do konštruktoru a nastaví MainPage ako page a bude ju používať.

```
public partial class App : Application
{
    public App(MainPage page)
    {
        InitializeComponent();
        MainPage = page;
    }
}
```

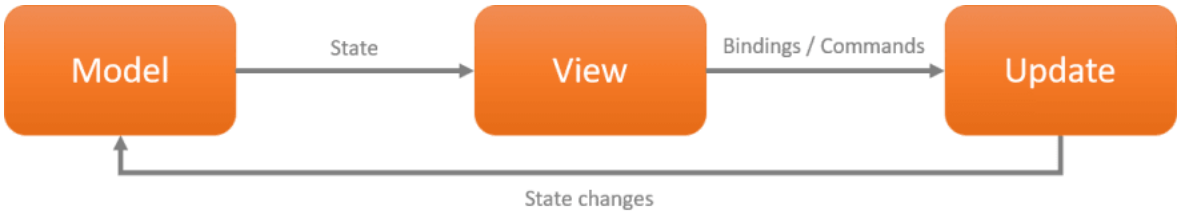
7.4.3 MainPage.xaml.cs

Do konštruktoru vložíme ViewModel pre MainPage, ktorý sme vytvorili a tak isto nabindujeme view model. Týmto pripojí BindingContext do view modelu a nemusíme vytvárať nový MainPage view model, pretože je už zaregistrovaný v kontajnere dependency injection.

```
public partial class MainPage : ContentPage
{
    public MainPage(MainPageViewModel viewModel)
    {
        InitializeComponent();
        BindingContext = viewModel;
    }}
```

8 MVU VZOR

Jedným z vylepšení je vzor Model View Update (MVU) alebo niekde môže byť označený aj ako Elm architektúra.



Obrázok 28 MVU vzor [37]

Na obrázku vyššie môžeme vidieť, ako vyzerá fungovanie tohto vzoru. View napísané v C# zobrazuje stavy, ktoré sú vystavené Modelom. Ak je vykonaný nejaký príkaz, je vyvolaný blok Update. Ten spraví to, že vykoná príkaz, ktorý môže zmeniť stav Modelu. Model ale nie je upraviteľný, takže Update vráti novú kópiu upravenú príkazom. Pretože Model je novou instanciou View, tak zobrazí zmeny. Na rozdiel od vzoru MVVM, v MVU dáta prúdia len jedným smerom, od stavu objektu ku view. [37]

8.1 Comet

Comet môžeme špecifikovať ako zjednodušený framework, ktorý je nadstavbou .NETu MAUI. Je zatiaľ v experimentálnej fáze a je stavaný na vzore MVU s použitím C#.

Všetko je zhrnuté do view. Užívateľské interakcie alebo služby prevádzajú aktualizácie objektov alebo inak povedané modelov a Comet aktualizuje view za nás automaticky.

Všetko užívateľské rozhranie je view, Body zase anotuje metódu, ktorá vracia konštrukciu view. Tu môžeme viazať stav s užívateľským rozhraním. Nasledujúci kód je šablóna, ktorú získame z tejto stránky [38].

```
[State]
readonly CometRide comet = new();

[Body]
View body()
     => new VStack {
         new Text(()=> $"({comet.Rides}) rides taken:{comet.CometTrain}")
                    .Frame(width:300)
                    .LineBreakMode(LineBreakMode.CharacterWrap),

         new Button("Ride the Comet! 🥓", ()=>{comet.Rides++;})
                    .Frame(height:44)
```

```
                .Margin(8)
                .Color(Colors.White)
                .Background(Colors.Green)
            .RoundedBorder(color:Colors.Blue)
            .Shadow(Colors.Grey,4,2,2),
    };
```

Comet je distribuovaný ako NuGet package, po inštalácii získame šablónu projektu, ktorej časť vidíme vyššie.

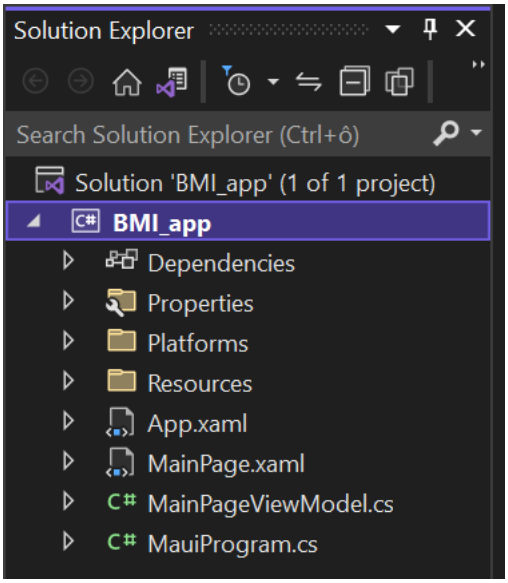
```
public class CometRide : BindingObject
    {
        public int Rides
        {
            get => GetProperty<int>();
            set => SetProperty(value);
        }
        public string CometTrain
        {
            get
            {
                return "📎".Repeat(Rides);
            }
        }
    }
```

Každé view má svoj stav, ktorý môžeme definovať. Aby sme získali aktualizácie o zmenách vlastností, budeme musieť rozšíriť BindingObject a použiť gettery a settery. Toto môžeme vidieť v šablóne projektu Comet.

Viac informácií o inštalácií a používaní Comet nájdem na tejto stránke [38].

9 .NET MAUI SINGLE-PROJECT

Medzi jednu z veľkých predností určite patrí single-project prístup. Na rozdiel od Xamarin.Forms, kde sme museli mať viacero projektov v prostredí, v .NETe MAUI nám stačí iba jeden.



Obrázok 29 BMI app Solution

Na obrázku môžeme vidieť súbory aplikácie, ktorú sme vytvárali vyššie. Prakticky všetko, okrem zložky Platforms, je zdieľané medzi sebou. Môžeme tu vytvárať nové Pages alebo zložky, ktoré stále budú patriť medzi zdieľaný kód. V zložke Platforms sa nachádzajú metadáta špecifické pre danú platformu. Zdieľaný kód máme aj v zložke Resources, kde máme ďalšie podzložky Fonts a Images. Tu môžeme uložiť typ písma alebo obrázky, ktoré chceme používať. Ďalej sa tam nachádza appicon.svg a appiconfg.svg, svg formát je preferovaný formát v NETe MAUI. Tento formát má veľkú výhodu, pretože ho môžeme veľkostne meniť donekonečna a všetky obrázky, ktoré používame v aplikácii, sa automaticky pretransformujú do požadovaných škál. Takže nemusíme vytvárať všetky veľkosti sami.

9.1 Súbor .csproj

V aplikačnom súbore .csproj môžeme nájsť skupinu ItemGroup.

V tejto skupine sa ako prvé nachádza MauiIcon, ktoré slúži na pridanie ikonky aplikácie. Ak máme vytvorenú ikonu, ktorú chceme použiť, vložíme ju sem a automaticky bude použitá

na všetky platformy. Taktiež musí byť vo formáte .svg, aby sme ju mohli používať multiplatforne.

```
<ItemGroup>
	<!-- App Icon -->
	<MauiIcon Include="Resources\appicon.svg" ForegroundFile="Resources\ap-
	piconfg.svg" Color="#512BD4" />
```

Rovnako to platí aj pre Splash Screen, ktorú sem môžeme vložiť a automaticky sa zarovná na stred a použije.

```
	<!-- Splash Screen -->
	<MauiSplashScreen Include="Resources\appiconfg.svg" Color="#512BD4"
	BaseSize="128,128" />
```

Tu sa stretávame s novým pojmom wildcard charakter, ktorým je hviezdička (*). Týmto znakom je označené s ktorými všetkými súbormi sa v zložke bude narábať.

```
	<!-- Images -->
	<MauiImage Include="Resources\Images\*" />
	<MauiImage Update="Resources\Images\dotnet_bot.svg" BaseSize="168,208"
/>

	<!-- Custom Fonts -->
	<MauiFont Include="Resources\Fonts\*" />

	<!-- Raw Assets (also remove the "Resources\Raw" prefix) -->
	<MauiAsset Include="Resources\Raw\**" LogicalName="%(Recur-
	siveDir)%(Filename)%(Extension)" />
</ItemGroup>
```

ZÁVĚR

Bakalárska práca vysvetlila a opísala súčasný vývoj mobilných aplikácií. Taktiež v teoretickej časti vysvetlila, aké aplikácie poznáme, čo riešime pri vývoji aplikácií, aké sú mobilné operačné systémy, vybrané frameworky a hlavný bod tejto práce .NET MAUI a jeho výhody a rozdiely oproti Xamarin.Forms.

Po prejdení si trhom a po práci s .NETom MAUI si dovolíme tvrdiť, že .NET MAUI môže naozaj zatriasť s trhom, čo sa týka vývoja multiplatformových aplikácií, pretože prichádza s radou výhod a otvára nové možnosti vývojárom. Umožňuje programovať mobilné aplikácie efektívnejšie, hlavne vďaka jednej kódovej základni, aj keď je ešte len vo verzii preview a nie v plnej sile. Prichádza s radou výhod, ktoré táto práca obsiahla, ako použitie dependency injection, lepšia navigácia a efektívnejšia práca so single-projectom alebo nový návrhový vzor MVU. MVU vzor ešte nie je úplne vyladený a taktiež použitie spolu s Comet je ešte len v experimentálnej fáze, takže v ňom môžu nastať nepríjemné problémy. Efektívnejšie je aj nastavovanie ikonky alebo splash screenu vďaka .csproj súboru v projekte, vývojár tak nepotrebuje nahrať všetky veľkosti, ale stačí mu len jedna .svg ikonka. Lepšie sa určite bude pracovať s plnou podporou hot-reloadu a hot-restartu.

V blízkej budúcnosti môžeme očakávať, že o NETe MAUI budeme čoraz viac počuť. Pre spracovanie bakalárskej práce bol tento framework veľmi zaujimavý a čo je veľmi dôležité, programovanie v ňom je zaujímavé a hlavne efektívne.

SEZNAM POUŽITÉ LITERATURY

[1] LEWIS, Shaun a Mike DUNN. *Native Mobile Development: A Cross-Reference for iOS and Android*. Kalifornia, USA: O'Reilly Media, 2019. ISBN 978-1492052876.

[2] Native App vs Hybrid App vs Web App, 2019. *AmiWebSolutions* [online]. Charlotte, NC: Ami Web Solutions [cit. 2022-02-28]. Dostupné z: https://www.amiwebsolutions.com/mobile-app-development/native-hybrid-web-apps/

[3] PANHALE, Mahesh. *Beginning Hybrid Mobile Application Development*. New York City, USA: Apress, 2015. ISBN 978-1484213155.

[4] LASTOVETSKA, Anastasiia, © 2009-2021. App Development Cost: Understand Your Budget To Build Powerful Apps. *MLSDev* [online]. MLSDev [cit. 2022-02-28]. Dostupné z: https://mlsdev.com/blog/app-development-cost

[5] NYAKUNDI, Hillary. What is a PWA? Progressive Web Apps for Beginners. *FreeCodeCamp* [online]. Oakland, CA: Free Code Camp, 2021 [cit. 2022-02-20]. Dostupné z: https://www.freecodecamp.org/news/what-are-progressive-web-apps/

[6] 10 Best Examples of Progressive Web Apps (PWA s) in 2020, 2020. *Appmaker.xyz* [online]. Appmaker [cit. 2022-02-28]. Dostupné z: https://appmaker.xyz/pwa-examples-successful-progressive-web-apps

[7] Mobile Operating System Market Share Worldwide, 2022. *Statcounter* [online]. © StatCounter [cit. 2022-02-28]. Dostupné z: https://gs.statcounter.com/android-version-market-share/mobile-tablet/worldwide

[8] KENTON, Will. Apple iOS. *Investopedia* [online]. New York: Dotdash, 2021 [cit. 2022-02-14]. Dostupné z: https://www.investopedia.com/terms/a/apple-ios.asp

[9] What is the architecture of iOS. *IntelliPaat* [online]. Karnataka, India: Intellipaat Software Solutions Pvt., 2021 [cit. 2022-02-14]. Dostupné z: https://intellipaat.com/blog/tutorial/ios-tutorial/ios-architecture/

[10] TAMMA, Rohit, 2020. *Practical Mobile Forensics: Forensically investigate and analyze iOS, Android, and Windows 10 devices*. 4th Edition. Birmingham, UK: Packt Publishing. ISBN 1838644423.

[11] CALDERA, Anurah, 2018. IOS Architecture. *Medium* [online]. Medium [cit. 2022-02-28]. Dostupné z: https://medium.com/@anuradhs/ios-architecture-a2169dad8067

[12] IOS 15 adoption, 2022. *Mixpanel* [online]. © 2021 Mixpanel [cit. 2022-02-28]. Dostupné z: https://mixpanel.com/trends/#report/ios_15

[13] BROWN, C. Scott. *What is Android? Here's everything you need to know* [online]. Android Authority, 2021 [cit. 2022-02-22]. Dostupné z: https://www.androidauthority.com/what-is-android-328076/

[14] KARCH. What Is Android?. *Lifewire* [online]. Dotdash, 2021 [cit. 2022-02-22]. Dostupné z: https://www.lifewire.com/what-is-google-android-1616887

[15] Windows 10 Mobile End of Support: FAQ. *Microsoft* [online]. © Microsoft, 2019 [cit. 2022-02-24]. Dostupné z: https://support.microsoft.com/en-us/windows/windows-10-mobile-end-of-support-faq-8c2dd1cf-a571-00f0-0881-bb83926d05c5

[16] Microsoft: all Lumia Windows Phone 8 devices will be upgraded to Windows 10, © 2022. *TheVerge* [online]. Vox Media [cit. 2022-03-01]. Dostupné z: https://www.theverge.com/2014/11/13/7216421/microsoft-says-every-lumia-windows-phone-8-device-will-get-windows-10

[17] KVARTALNYI, Nazar, 2021. BENEFITS OF PROGRESSIVE WEB APPS (PWA) – ADVANTAGES AND DISADVANTAGES. *Inoxoft* [online]. © 2022 INOXOFT, ALL RIGHTS RESERVED [cit. 2022-03-31]. Dostupné z: https://inoxoft.com/blog/benefits-of-progressive-web-apps-pwa-advantages-and-disadvantages/

[18] HAJIAN, Majid, 2019. *Progressive Web Apps with Angular: Create Responsive, Fast and Reliable PWAs Using Angular*. New York City: Apress. ISBN 1484244486.

[19] Progressive Web Apps: Advantages and Disadvantages, 2021. *Brainhub* [online]. BRAINHUB 2022 [cit. 2022-03-31]. Dostupné z: https://brainhub.eu/library/progressive-web-apps-advantages-disadvantages/

[20] Xamarin documentation. Microsoft Docs [online]. Oficiálna dokumentácia firmy Microsoft Corporation [cit. 2021-10-16]. Dostupné z: https://docs.microsoft.com/en-us/xamarin/

[21] HERMES, Dan, 2015. *Xamarin Mobile Application Development: Cross-Platform C# and Xamarin.Forms Fundamentals*. New York City: Apress. ISBN 9781484202159.

[22] .NET documentation. Microsoft Docs [online]. Oficiálna dokumentácia firmy Microsoft Corporation [cit. 2021-10-16]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/

[23] PRICE, Mark J., 2021. *C# 10 and .NET 6 – Modern Cross-Platform Development: Build apps, websites, and services with ASP.NET Core 6, Blazor, and EF Core 6 using Visual Studio 2022 and Visual Studio Code*. 6th ed. Birmingham, UK: Packt Publishing. ISBN 1801077363.

[24] What is .NET Framework?. *Microsoft* [online]. © Microsoft 2022 [cit. 2022-04-03]. Dostupné z: https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet-framework

[25] Most used libraries and frameworks among developers, worldwide, as of 2021, 2021. *Statista* [online]. © Statista 2022 [cit. 2022-04-03]. Dostupné z: https://www.statista.com/statistics/793840/worldwide-developer-survey-most-used-frameworks/

[26] NET Multi-platform App UI documentation. Microsoft Docs [online]. Oficiálna dokumentácia firmy Microsoft Corporation [cit. 2021-10-16]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/maui/

[27] Intermediate Language (IL). *Techopedia* [online]. © 2022 Techopedia [cit. 2022-04-03]. Dostupné z: https://www.techopedia.com/definition/24290/intermediate-language-il-net

[28] ASP.NET Core Blazor, 2022. *Microsoft* [online]. © Microsoft 2022 [cit. 2022-04-03]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/blazor/?WT.mc_id=dotnet-35129-website&view=aspnetcore-6.0

[29] FERREIRA, Oberdan, 2021. What Microsoft's .NET MAUI means for the future of Xamarin and cross-platform app development. *ArcTouch* [online]. Copyright 2022 ArcTouch [cit. 2022-04-03]. Dostupné z: https://arctouch.com/blog/maui-xamarin/

[30] The New .NET Multi-platform App UI, 2021. *Microsoft* [online]. © Microsoft 2022 [cit. 2022-04-03]. Dostupné z: https://devblogs.microsoft.com/xamarin/the-new-net-multi-platform-app-ui-maui/

[31] GANAPATHY KATHIRESAN, Selva, 2021. *Syncfusion* [online]. © 2001 - 2022 Syncfusion [cit. 2022-04-03]. Dostupné z: https://www.syncfusion.com/blogs/post/advantages-net-maui-over-xamarin.aspx

[32] KATONA, Ján, 2020. Je lepšia natívna alebo hybridná aplikácia?. *Eliteml* [online]. © 2022 ELITE BLOG [cit. 2022-04-05]. Dostupné z: https://www.eliteml.sk/blog/je-lepsia-nativna-alebo-hybridna-aplikacia/

[33] What is a Mobile Operating System?: Features & Types. *Study* [online]. © copyright 2003-2022 Study.com [cit. 2022-04-05]. Dostupné z: https://study.com/academy/lesson/what-is-a-mobile-operating-system-features-types.html

[34] O, David, 2022. Announcing .NET MAUI Preview 12. *Microsoft* [online]. © Microsoft 2022 [cit. 2022-04-15]. Dostupné z: https://devblogs.microsoft.com/dotnet/announcing-net-maui-preview-12/

[35] ORTINAU, David, 2021. Feature Roadmap. *GitHub* [online]. © 2022 GitHub [cit. 2022-04-15]. Dostupné z: https://github.com/xamarin/Xamarin.Forms/wiki/Feature-Roadmap

[36] OLSON, Scott, 2012. *Professional Cross-Platform Mobile Development in C#.* Birmingham, England: Wrox. ISBN 1118157702.

[37] MVU or MVVM with MAUI App ?, 2022. *JM parent* [online]. Copyright © 2022 JM Parent [cit. 2022-05-10]. Dostupné z: https://www.jmparent.com/2022/01/18/mvu-or-mvvm-with-maui-app/

[38] Comet, 2022. *GitHub* [online]. © 2022 GitHub [cit. 2022-05-11]. Dostupné z: https://github.com/dotnet/Comet

[39] Build your first app, 2022. *Microsoft* [online]. © Microsoft 2022 [cit. 2022-05-11]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/maui/get-started/first-app?pivots=devices-android

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

MAUI	Multi-platform user interface
MVU	Model-view-update
MVVM	Model-view-viewmodel
GPS	Globálny lokalizačný systém
VS	Visual Studio
HTML	Hypertextový značkový jazyk
CSS	Kaskádové štýly
PWA	Progresívne webové aplikácie
HTTPS	Zabezpečený hypertextový prenosový protokol
CPU	Procesor základnej jednotky
OS	Operačný systém
UI	Užívateľské rozhranie
SDK	Software development kit
XAML	Extensible aplication markup language
XML	Extensible markup language
API	Application programming interface
CLR	Common language runtime
HTTP	Hypertextový prenosový protokol
CLI	Command-line interface
BMI	Index telesnej hmotnosti
VS	Visual Studio

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH

P1 CD so zdrojovými kódmi a bakalárskou prácou