Návrh multiplatformní mobilní aplikace pro rychlou komunikaci vyučujících se studenty

Martin Gabriel

Bakalářská práce
2019

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2018/2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Martin Gabriel**
Osobní číslo:	**A15655**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informační a řídicí technologie**
Forma studia:	**kombinovaná**
Téma práce:	**Návrh multiplatformní mobilní aplikace pro rychlou komunikaci vyučujících se studenty**
Téma anglicky:	**A Multiplatform Mobile Application for Rapid Communication Between Students and Teachers**

Zásady pro vypracování:

1. **Popište vybrané technologie a doporučené postupy pro tvorbu multiplatformních mobilních aplikací.**
2. **Rozeberte daný problém a navrhněte vhodnou architekturu pro danou aplikaci.**
3. **Navrhněte implementaci aplikace s využitím vhodných technologií s důrazem na zasílání rychlých a cílených zpráv studentům.**
4. **Vytvořte ukázkovou aplikaci demonstrující klíčové prvky řešení a popište její významné části.**
5. **Demonstrujte výsledky a formulujte závěr.**

Rozsah bakalářské práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **MAREŠ, Amadeo. 1001 tipů a triků pro C#. Brno: Computer Press, 2008. ISBN 978-80-251-2125-2.**
2. **FREEMAN, Adam. Pro ASP.NET MVC 5. Fifth edition. New York, New York: Apress, 2013. Expert's voice in ASP.NET. ISBN 978-1-4302-6529-0.**
3. **LACKO, Ľuboslav. Mistrovství v SQL Server 2012: [kompletní průvodce databázového experta]. Brno: Computer Press, 2013. ISBN 978-80-251-3773-4.**
4. **COPELAND, Marshall. Microsoft Azure: planning, deploying, and managing your data center in the cloud. Berkeley, CA: Apress, 2015. Expert's voice in Microsoft Azure. ISBN 1484210441.**
5. **PETZOLD, Charles. Creating Mobile Apps with Xamarin.Forms [online]. Redmond, Washington 98052-6399: Microsoft Press, 2016. ISBN 978-1-5093-0297-0. Dostupné z: https://developer.xamarin.com/guides/xamarin-forms/creating-mobile-apps-xamarin-forms/**
6. **SINGLETON, James. ASP.NET Core 1.0 High performance. Birmingham: Packt Publishing, 2016. ISBN 9781785881893.**

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

Datum zadání bakalářské práce: **21. prosince 2018**

Termín odevzdání bakalářské práce: **15. května 2019**

Ve Zlíně dne 21. prosince 2018

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky

doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

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 diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- 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 – diplomovou/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 …….

podpis diplomanta

ABSTRAKT

Cílem práce je návrh vhodných technologií a postupů pro aplikaci pro rychlou a efektivní komunikaci vyučujících se studenty. Aplikace bude umožňovat zasílání rychlých a cílených zpráv studentům o změnách výuky a dále bude aplikace umožňovat studentům rychlé hodnocení výuky. V teoretické části budou popsány vybrané technologie, doporučené postupy, návrhové vzory a techniky pro tvorbu mobilních aplikací. V praktické části budou popsané klíčové části navrženého řešení.

Klíčová slova:

Xamarin, iOS, Android, Azure, .NET Core, C#, Push notifikace

ABSTRACT

The goal of this work is design appropriate technologies and process for application for quick and effective communication teachers with students. Application will allow send quick and targeted messages to students about changes in the learning plan and application will also allow quick evaluation of lesson. In the theoretical part will be described technologies and recommended procedures, design patterns and techniques for creating mobile applications. In practical part will be described key parts of proposed solution.

Keywords:

Xamarin, iOS, Android, Azure, .NET Core, C#, Push notification

Zde bych chtěl poděkovat vedoucímu práce Ing. et Ing. Eriku Královi, Ph.D. za vedení, rady a konzultace při psaní této bakalářské práce. Dále bych chtěl poděkovat své přítelkyni Veronice a celé rodině za podporu.

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

OBSAH

ÚVOD

Žijeme ve světě, kde téměř každý nosí u sebe mobilní telefon a ve většině případů je to chytrý telefon, který je neustále připojen k internetu. Okamžitá komunikace se dnes stává standardem. Denně používáme SMS zprávy, e-maily, využíváme instant messaging aplikace jako WhatsApp, nebo Messanger. Díky internetu jsme neustále propojeni s naším okolím. Motivací v této práci je umožnit učitelům komunikovat se studenty pomocí chytrého telefonu s okamžitou zpětnou vazbou. V některých případech je příliš zdlouhavé psát e-mail a doufat, že si ho někdo ze studentů přečte, když jde například o zpoždění začátku výuky, nebo o změnu učebny. Proto je cílem této práce najít způsob, jak pomocí chytrých telefonů zajistit rychlou komunikaci od učitele ke studentům a pokud možno to vše zpracovat v reálném čase. Jedním z hlavních požadavků je, aby mobilní aplikace byla multiplatformní, tedy aby výsledný kód šel kompilovat pro více platforem.

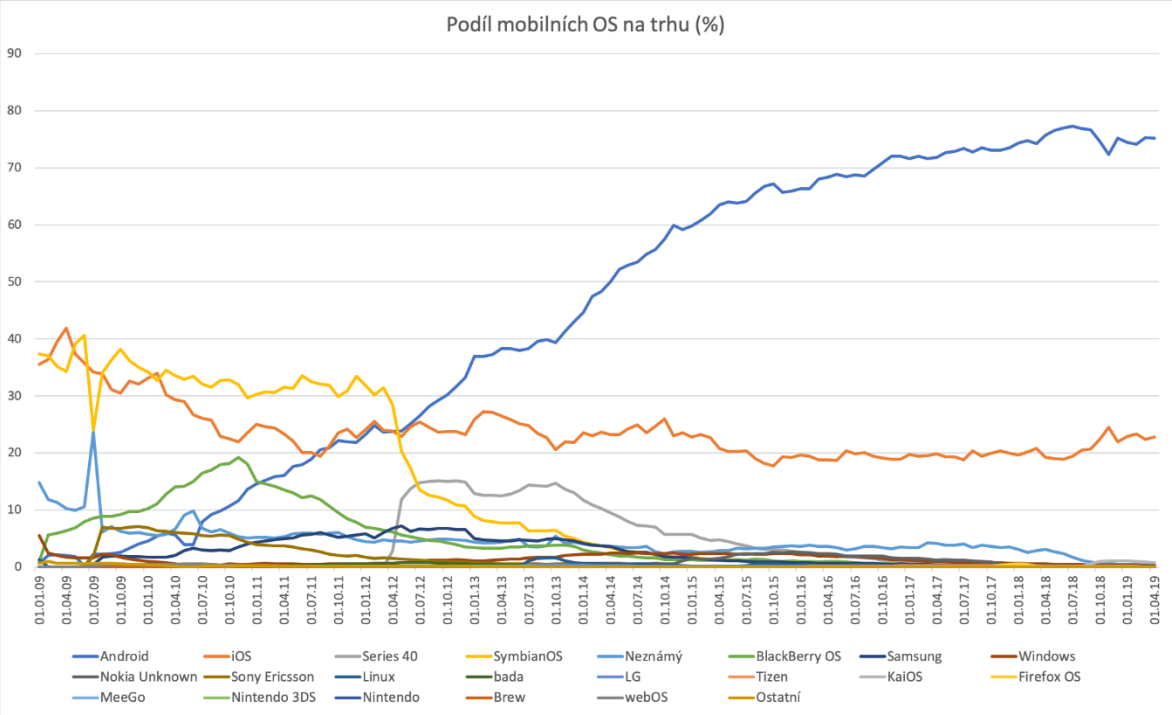
V teoretické části budou popsány mobilní platformy, multiplatformní vývoj s zaměřením na framework Xamarin. Dále bude popsána cloudová služba Azure s nasazením serverové aplikace v .NET Core. V poslední kapitole teoretické části budou popsány notifikace na nejrozšířenějších mobilních platformách. V praktické části bude rozebrána navržená architektura projektu zahrnující serverovou část napsanou v .NET Core a mobilní aplikaci napsanou ve frameworku Xamarin. Dále bude popsána implementace push notifikací pro jednotlivé platformy a několik příkladů využití ukázkové mobilní aplikace.

Cílem této práce je navrhnout architekturu projektu, která zajistí fungování multiplatformní mobilní aplikace pro rychlou komunikaci učitelů se studenty a použití této architektury v ukázkové mobilní aplikaci. Ukázková aplikace demonstruje možnosti architektury ve zpracování zasílaných zpráv a online hodnocení výuky v reálném čase.

I. TEORETICKÁ ČÁST

1 MOBILNÍ PLATFORMY

Pojmem mobilní platformy rozumíme operační systémy a ekosystémy, na kterých jsou postaveny současné mobilní telefony, tablety a jiné chytré zařízení. V rámci práce se budeme zabývat především mobilními telefony a tablety. Mezi nejvýznamnější systémy patří operační systémy Android od společnosti Google a iOS od společnosti Apple.



Obr. 1. Podíl mobilních OS na trhu [1]

Z grafu vidíme postupnou dominanci operačních systémů Android a iOS. Ostatní operační systémy mají zanedbatelný podíl na trhu mobilních zařízení.

1.1 Android

Android je mobilní operační systém založený na jádře Linuxu, který vyvíjí uskupení Open Handset Alliance (pod hlavičkou společnosti Google) a je distribuován jako opensource. Systém Android má největší zastoupení na mobilním trhu.

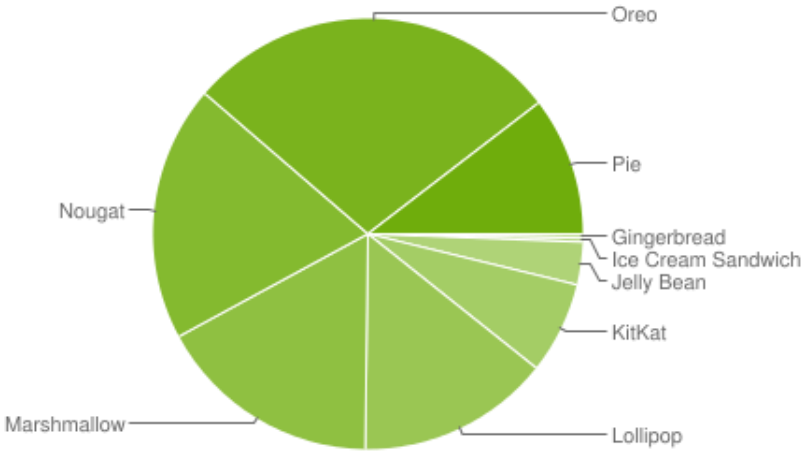
1.1.1 Architektura systému Android

Architektura systému je rozdělena na 5 vrstev [2]:

1) Linux Kernel - nejnižší vrstvou je jádro operačního systému, které se stará o práci mezi software a hardware.

2) Knihovny – jsou napsány v jazyce C/C++, využívají různé komponenty systému, slouží vývojářům prostřednictvím Android Application Framework k práci s práci s audio/video soubory, webovým prohlížečem, relační databází, vykreslování 3D grafiky a další
3) Android Runtime – obsahuje virtuální stroj Dalvik, základní knihovny jazyka Java,
4) Application network – poskytuje přístup ke službám, které mohou být použity v aplikacích, tyto služby mohou zpřístupňovat data v jiných aplikacích (například kontakty, aktuální polohu, atd.), přístup k hardware, notification manager, aktivity manager a další
5) Aplikace – základní aplikace s uživatelským rozhraním (například mail klient, webový prohlížeč, mapy a podobně)

Operační systém Android je nejrozšířenějším mobilním operačním systémem. Nasazuje jej do svých zařízení široká škála výrobců, z čehož pramení jeho nevýhoda – velké množství nasazených verzí s nejistou možností aktualizací systému (například bezpečnostních záplat), protože aktualizace závisí pouze na výrobci.



Obr. 2. Podíl verzí systému Android na trhu v Q1 2019 [3]

1.2 Operační systém iOS

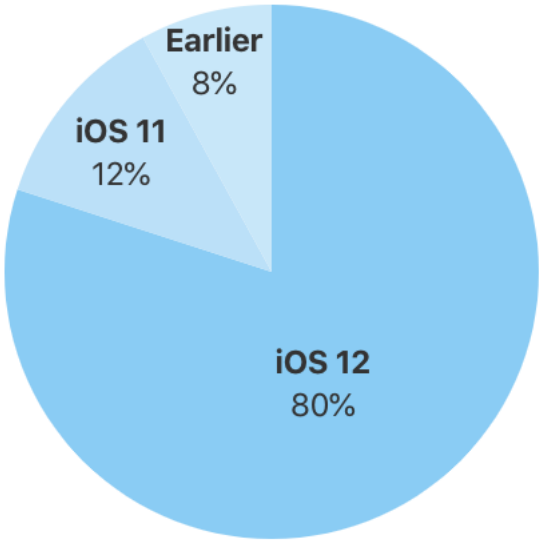
iOS je mobilní operační systém založený na macOS (je jeho odlehčenou verzí). Jedná se o systém UNIXového typu. Původně byl iOS vytvořen pouze pro mobilní telefony iPhone (pod názvem iPhoneOS), ale později se začal používat na dalších zařízeních jako hudební přehrávače iPod, nebo tablety iPad.

1.2.1 Architektura systému iOS

Architektura systému je rozdělena na 4 základní vrstvy [4]:

1) Core OS – stará se o nízkoúrovňový přístup k hardware, dále se stará například o bluetooth konektivitu, služby zabezpečení, autentifikaci a další
2) Core Services Layer – stará se o vysokoúrovňové služby, například lokační služby, relační databáze, správa úloh v aplikaci, podpora XML dokumentů a další
3) Media Layer – umožňuje vytváření grafických aplikací, přehrávání animací, videí a zvuků
4) Cocoa Touch – obsahuje nejdůležitější frameworky pro vývoj aplikací, poskytuje infrastrukturu pro implementaci grafického uživatelského rozhraní, interakce s uživatelem a poskytuje vysokoúrovňové systémové služby, například multitasking, push notifikace, rozpoznávání gest a další

Operační systém iOS je distribuován pouze pro zařízení společnosti Apple. Svou koncepcí se jedná o uzavřený ekosystém, což má výhodu v optimalizaci a bezpečnosti, ale nevýhodu v nulové upravitelnosti. Oproti Androidu se nové verze iOS aktualizují daleko rychleji na větší poměr zařízení na trhu.



Obr. 3. Podíl verzí systému iOS na trhu Q1 2019 [5]

1.3 Další mobilní operační systémy

Další operační systémy mají na trhu minimální zastoupení. Mezi nadějnou konkurenci donedávna patřil operační systém Windows Phone od společnosti Microsoft (dříve znám jako

Windows Mobile, později jen jako Windows), který ale v současné době není vyvíjen a v roce 2019 končí oficiální podpora tohoto systému. Mezi další neúspěšné projekty paří například FirefoxOS [6], nebo Symbian. Existuje několik aktivně vyvíjených alternativ jako například Sailfish OS.

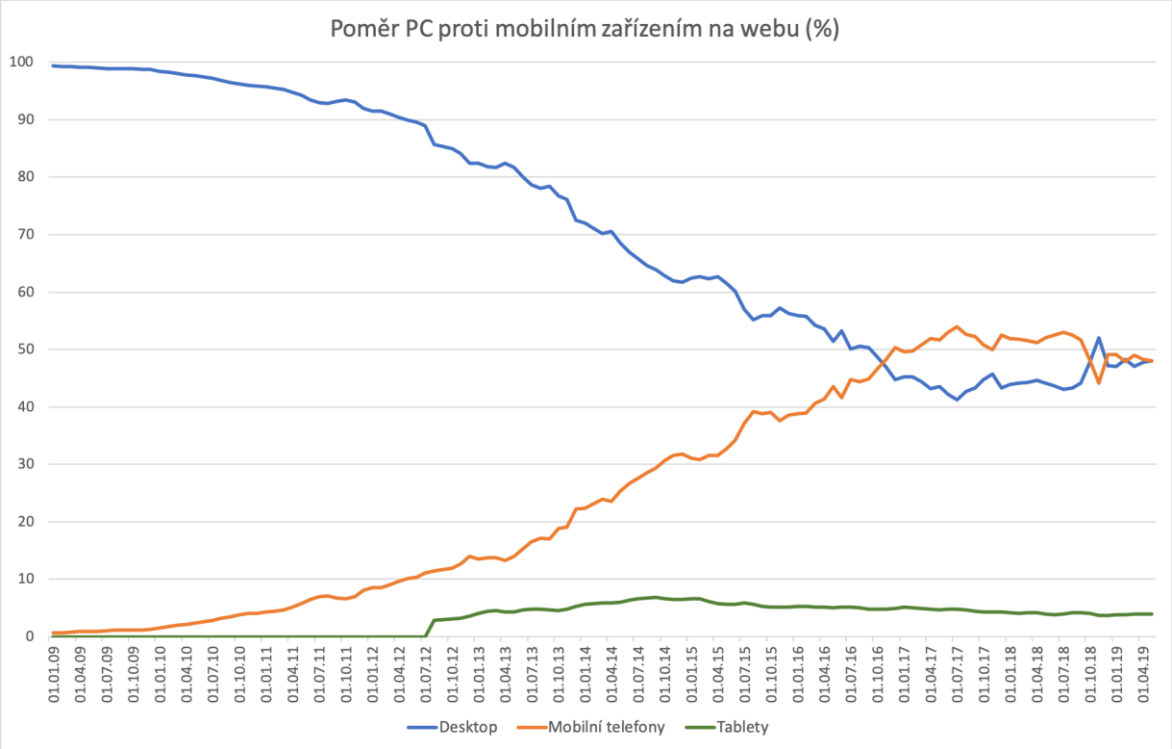
2 MULTIPLATFORMNÍ VÝVOJ

Multiplatformní vývoj chápeme jako vývoj software, jehož výsledek lze spustit na více platformách, nebo více operačních systémech. Existuje několik druhů multiplatformního vývoje a dělíme je podle použití.

2.1 Nativní aplikace

Nativní vývoj znamená vytvoření aplikace pro každou platformu zvlášť. Každá platforma má vlastní přístup k uživatelskému přístupu, vlastní systémové API, odlišný jazyk a vývojové prostředí. Tento přístup se používá hlavně na velkých projektech, kde je potřeba dosáhnout maximálního výkonu a není problémem časová náročnost spojená s vyššími finančními náklady na vývoj.

2.2 Progresivní webové aplikace



Obr. 4. Poměr PC proti mobilním zařízením na webu [1]

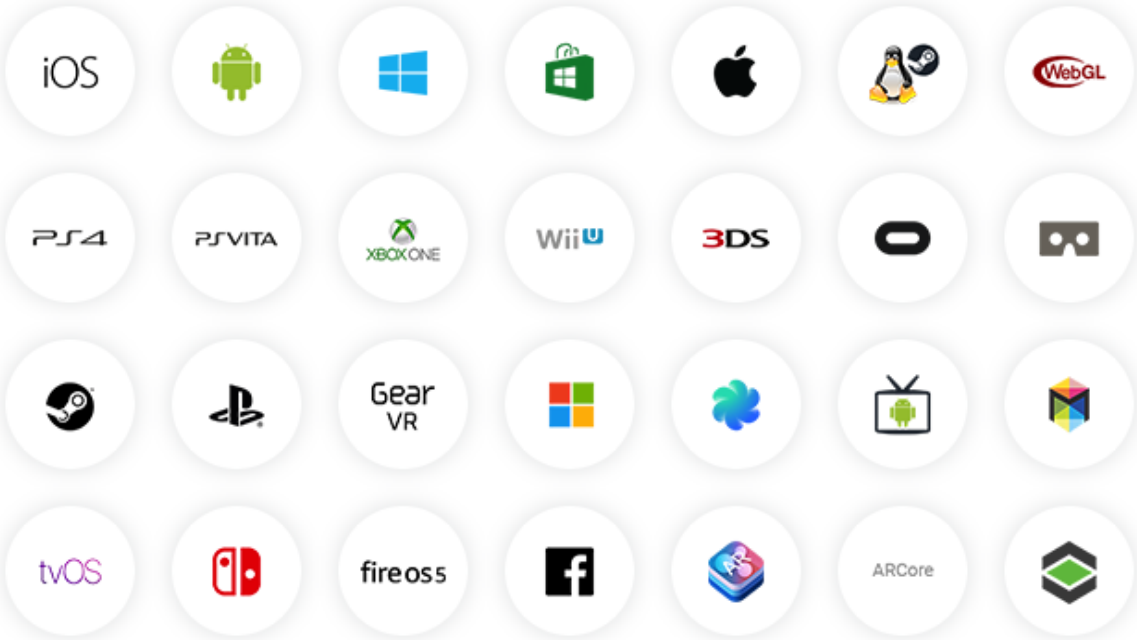
Webové stránky přizpůsobené pro mobilní zařízení mohou sloužit jako mobilní aplikace. Při vývoji webů je potřeba dbát na vzestupný trend prohlížení webů na mobilních zařízeních. Pro tento účel vznikl pojem responzivní design, který klade důraz na to, aby se stránka přizpůsobila velikosti displeje, na kterém je zobrazena. K tomuto typu vývoje slouží přístup

Mobile First, tedy návrh stránky nejprve pro mobilní zařízení, ze kterého později vyplyne i desktopová verze [7]. Nevýhodou tohoto přístupu je neustálá potřeba připojení k internetu.

Další možností jsou hybridní HTML5 aplikace, které vypadají jako nativní aplikace, ale prakticky jde o webový prohlížeč, který zobrazuje uživatelské rozhraní pomocí webových technologií a má přístup k nativnímu API k ovládání fotoaparátu, GPS, apod. Zástupcem tohoto přístupu je Framework PhoneGap distribuovaný jako open source, který vychází z Apache Cordova frameworku [8].

2.3 Herní vývoj

Pro vývoj her se používá herní engine, tedy software umožňující jednoduchý a v mnohých případech i multiplatformní vývoj her. Mezi nejznámější multiplatformní herní engine patří například Unity3D. Nabízí vývoj her pro více než 25 platforem včetně mobilních telefonů, osobních počítačů a konzolí. U mobilních telefonů dosahuje nativního vysokého výkonu díky využití grafických API Metal (iOS) a Vulkan (Android) [9].



Obr. 5. Výčet podporovaných platforem Unity3D [9]

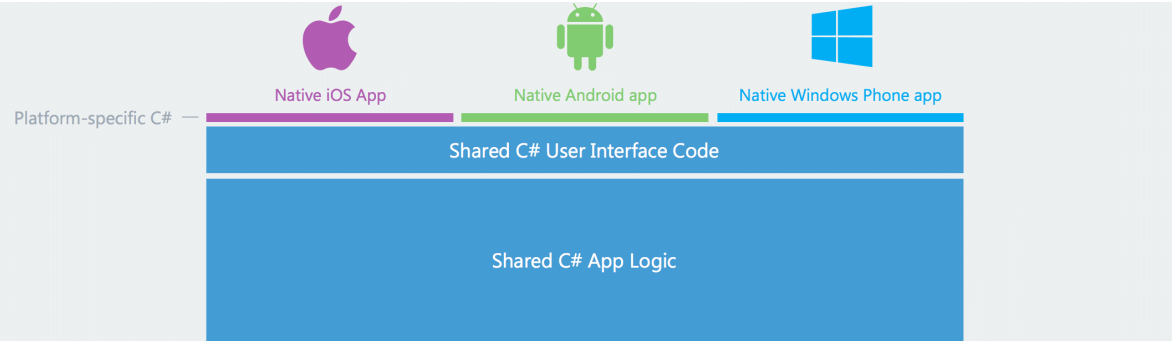
2.4 Multiplatformní vývoj

Multiplatformní vývoj aplikací se nejvíce vyplatí při vývoji business aplikací, nebo tam, kde je potřeba sdílet co nejvíce kódu a šetřit tak čas vývoje. Mezi nejznámější zástupce

multiplatformního vývoje patří Xamarin, kterému nově konkuruje open source Framework Flutter od společnosti Google [10]. Xamarin využívá jazyka C#, kdežto Flutter jazyka Dart. Dalším rozdílem je prostředí, které využívají dané frameworky. Xamarin využívá multiplatformní prostředí Mono, Flutter využívá prostředí Skia [11]. Výsledné aplikace jsou vyvíjeny rychle, mají nativní výkon a flexibilní uživatelské rozhraní.

3 XAMARIN

Xamarin je společnost kterou založil Miguel de Icaza v roce 2011, stojí za vznikem platforem Mono, Mono for Android, MonoTouch, což jsou multiplafromní implementace Common Language Infrastructure (CLI) a Common Language Specifications (.NET). V roce 2016 byla firma Xamarin zakoupena společností Microsoft a Xamarin byl plně integrován do vývojového prostředí Visual Studio [12].



Obr. 6. Princip sdílení kódu v Xamarin platformě [13]

Platforma Xamarin je určena pro vývoj aplikací, kde je potřeba sdílet co nejvíce společného kódu. Umožňuje vytvářet nativní aplikace se společnou aplikační logikou (Xamarin.iOS a Xamarin.Android), nebo aplikace se sdílenou aplikační logickou a uživatelským rozhraním (Xamarin.Forms) přičemž klade důraz na zachování výkonu proti čistě nativním aplikacím.

3.1 Mono

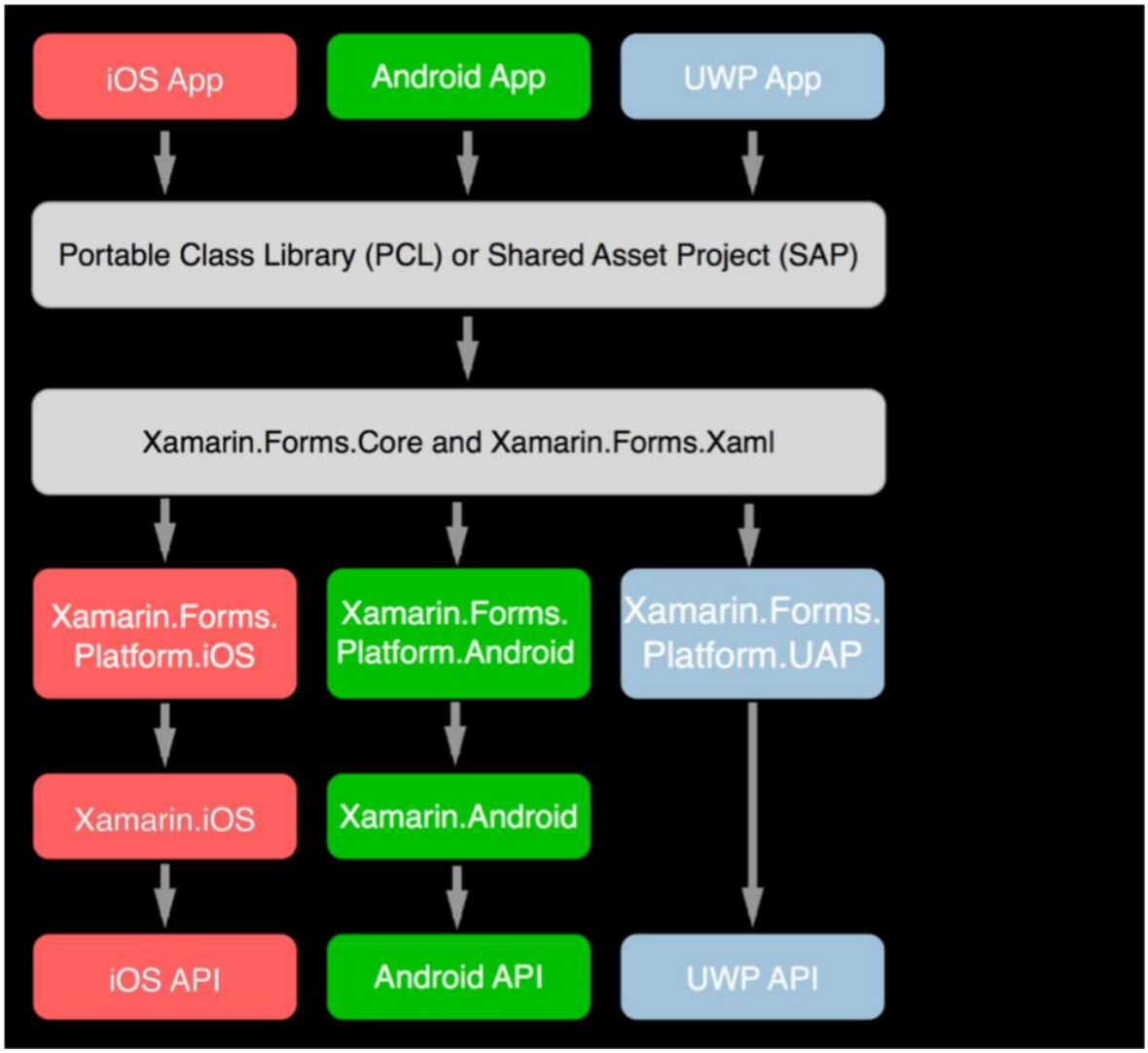
Mono je softwarová platforma vytvořená pro vývoj multiplatformních aplikací prostřednictvím .NET standart. Je vyvíjena jako open source a založená na ECMA standartu pro C# a Common Language Runtime. Smyslem této plaftormy je umožnění spouštění .NET aplikací na jiných operačních systémech než Windows. [14] V době psaní této práce podporuje Mono vše z .NET verze 4.7 vyjma WPF, WWF, omezeně WCF a ASP.NET async stack [15].

Mono je v současné době podporováno na těchto platformách [15]:

- Linux
- Mac OSX, iOS, tvOS, watchOS
- Sun Solaris
- BSD – OpenBSD, FreeBSD, NetBSD
- Microsoft Windows, XboxOne
- Sony Playstation 4

3.2 Xamarin.Forms

Xamarin.Forms umožňuje vytváření multiplatformního User Interface (dále UI), který je kompilován do nativního kódu jednotlivé platformy [16]. Podle oficiálního webu společnosti Xamarin je při vývoji v Xamarin.Forms 96% kódu sdíleného mezi platformami [17].



Obr. 7. Popis vytvoření multiplatformní Xamarin.Forms aplikace [16]

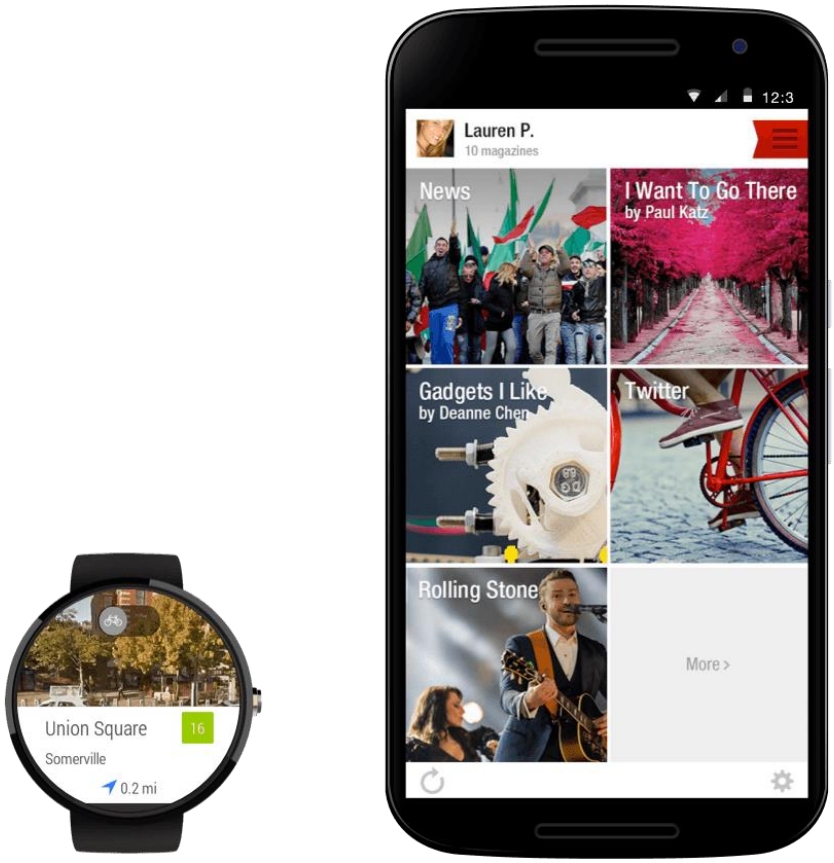
3.2.1 Výkon aplikací v Xamarin.Forms

Mezi multiplatformním UI frameworkem Xamarin.Forms, nativním Xamarin.iOS nebo Xamarin.Android a čistě nativní aplikací, jsou výkonnostní rozdíly. Podle testů webu AlexSoft jsou nejvíce výkonné nativní aplikace v iOS, nejvíce vyvážené jsou pak aplikace napsané v Xamarin.iOS a Xamarin.Android, avšak uživatel nejspíše vůbec nepozná rozdíl mezi aplikací napsanou v Xamarin.Forms, které se jeví jako nejpomalejší [16].

3.3 Xamarin.Android

Xamarin.Android je framework, který umožňuje vývoj nativních Android apliakcí s pomocí Android SDK v jazyce C# nebo F# za použití knihoven .NET a vývojového prostředí Visual Studio. Ke svému chodu potřebuje Windows 7 a vyšší, Visual Studio 2015 nebo 2017 a Xamarin for Visual Studio. Instalace obstará stažení dalších potřebných částí jako Xamarin Workbooks, Xamarin Profiler, Xamarin Remoted Simulator, Android NDK, Android SDK, Java SE Develpment Kit, Google Android Emulator a další [18].

Kompilace v Xamarin.Android využívá principu Just In Time. C# kód je zkompilován do assemblies a ty jsou součástí výsledného APK souboru. Spuštění aplikace potom probíhá pomocí Mono virtuálního stroje, který vykonává instrukce z assemblies stejně jako by virutální stroj Dalvik zpracovával nativní aplikaci napsanou v jazyku Java [18].



Obr. 8. Xamarin.Android aplikace pro Android a Wear OS [15]

3.4 Xamarin.iOS

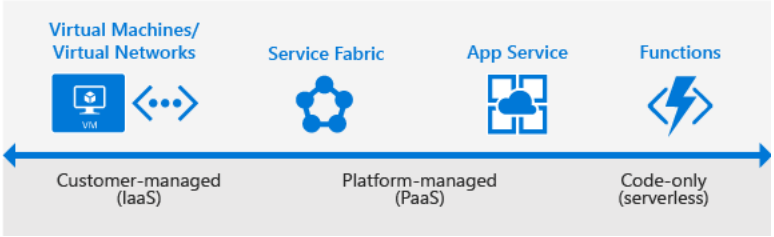
Xamarin.iOS je framework, který umožňuje vývojářům vytvářet nativní iOS aplikace pomocí SDK iOS, přičemž lze psát aplikaci v jazyce C# za použití knihoven .NET Base Class Library (BCL) a vývojového prostředí Visual Studio. Mezi systémové požadavky patří iOS SDK, poslední verze Xcode a macOS Seirra 10.12 nebo vyšší, takže je k vývoji nativních iOS aplikací v Xamarinu potřeba zařízení od společnosti Apple [19].

Kompilace v Xamarin.iOS využívá principu Ahead of Time Compilation, kde je C# kód převeden do nativního ARM kódu a výsledkem je balík .APP s aplikací. V Xamarin.iOS není dostupná dynamická generace kódu, takže za chodu programu musí být všechen kód již zkompilován [19].

Obr. 9. Xamarin.iOS aplikace pro iPhone a WatchOS [13]

4 AZURE A .NET CORE

Azure je sada cloudových služeb, která slouží k vytváření, hostování a škálování webových aplikací na cloudových serverech společnosti Microsoft, k virtualizaci, nasazení databází a dalších služeb. Umožňuje nasazení, testování, monitoring a správu služeb. Cloudová služba funguje na run-time operačním systému Windows Azure, který zprostředkovává řadu služeb, které není potřeba instalovat a jejich konfigurace probíhá z prostředí webového portálu [20].



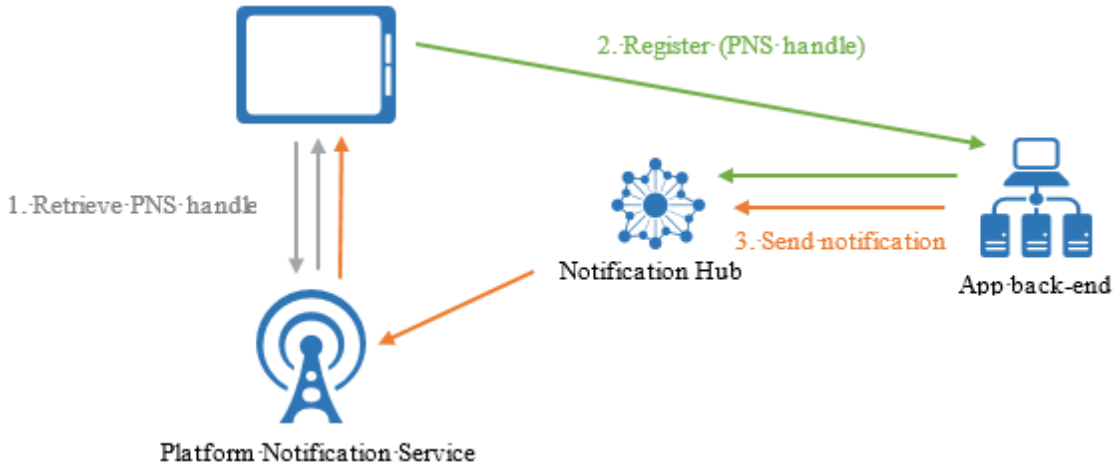
Obr. 10. Popis Azure

4.1 Azure functions

Jedná se o tzv. bezserverovou výpočetní platformu, která umožňuje vytváření jednoduchých API funkcí bez potřeby celé serverové aplikace a infrastruktury. Vyvíjené API funkce je možné nasazovat přímo do cloudu a rychleji tak integrovat nové služby. Je možno psát kód v jazycích C#, F#, Node.js, Java nebo PHP [21].

4.2 Azure Notification Hubs

Notification Hubs je modul, který umožňuje odesílání push notifikací na zařízení se systémy iOS, Android, Windows, nebo Kindle. Využívá k tomu služby Apple Push notification Servuce, Google Cloud Messaging, Windows Push Notification Service, Microsoft Push Notification Service a další. Dá se připojit k webovým službám založených na platformách .NET, PHP, Java nebo Node.js.



Obr. 11. Způsob zpracování push notifikací pomocí Notification Hub [22]

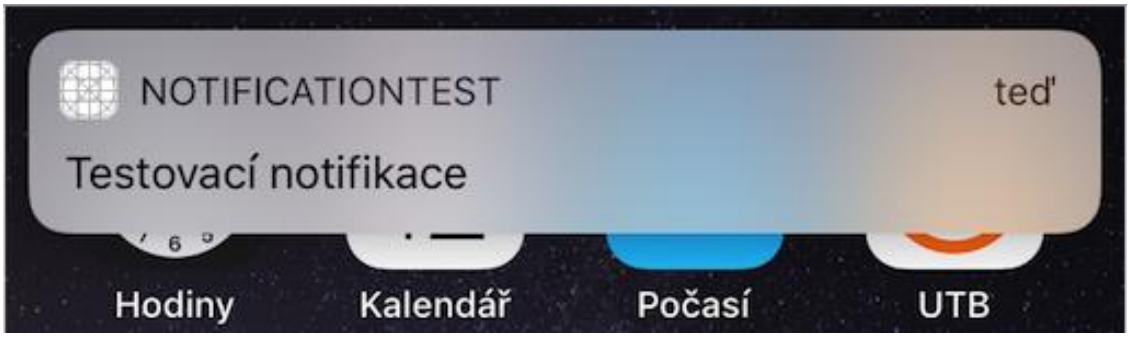
Výhodou této služby je jednoduchá implementace v .NET a správa v Azure portálu. Alternativou je služba Firebase od společnosti Google [23].

4.3 .NET Core

Jedná se o open source framework pro systémy Windows, maxOS a Linux který podporuje multiplatformní vývoj webových aplikací ASP.NET Core, command-line aplikací, knihoven a UWP aplikací. .NET Core neobsahuje Windows Forms a WPF, které renderují GUI pro desktopové aplikace na OS Windows. Typicky se hodí na vývoj multiplatformních micro-services.

5 NOTIFIKACE

Notifikace jsou jednou z důležitých částí (nejen mobilních) operačních systémů. Slouží k upozornění uživatele na aktivitu aplikace. Může jít například o upozornění na příchozí zprávu, zmeškaný hovor, nebo o připomínku v kalendáři, která dává prostřednictvím notifikací uživateli vědět, že se něco stalo [24].



Obr. 12. Příklad zobrazení notifikace v horní části obrazovky telefonu

Notifikace lze z pravidla spravovat v notifikačním centru, což je součást operačního systému, která sdružuje chronologicky seřazené notifikace ze všech aplikací a programů v zařízení [25].

Notifikace dělíme na dvě základní kategorie. Lokální notifikace a Push notifikace.

5.1 Lokální notifikace

U lokálních notifikací aplikace předává detaily o notifikaci operačnímu systému uvnitř daného zařízení, který následně zpracovává zobrazení notifikace. Děje se tak i v momentě kdy aplikace, nebo program, nejsou vůbec spuštěny [24]. Může jít například o událost v kalendáři, která v určitý čas upozorní uživatele prostřednictvím notifikace.

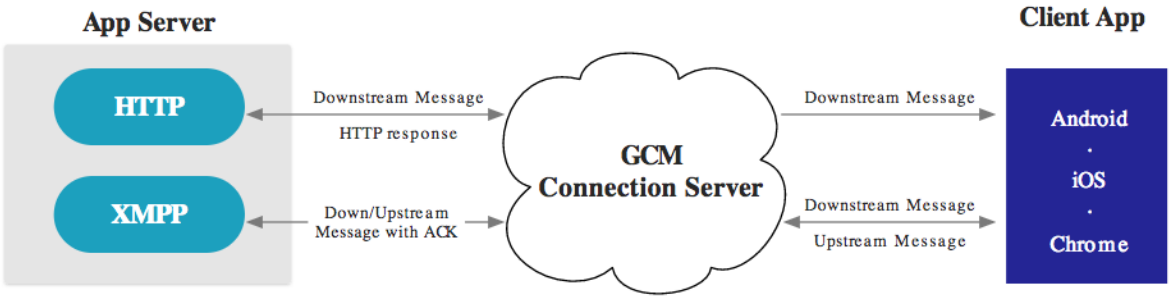
5.2 Push notifikace

U push notifikací operační systém přijímá a zpracovává notifikace na základně vnějšího podnětu ze strany serverové notifikační služby. Serverová aplikace komunikuje se zařízením přes serverové služby, které si každá platforma řeší zvlášť.

5.2.1 Push notifikace v operačním systému Android

V mobilním operačním systému Android se zasílání a příjem push notifikací řeší pomocí služby Google Cloud Messaging (dále GCM). Serverová aplikace vyhodnotí požadavek na

zaslání push notifikace na konkrétní zařízení, pomocí HTTP a XMPP protokolu předá požadavek GCM serveru. Ten následně odešle notifikace ke konkrétnímu zařízení [26].

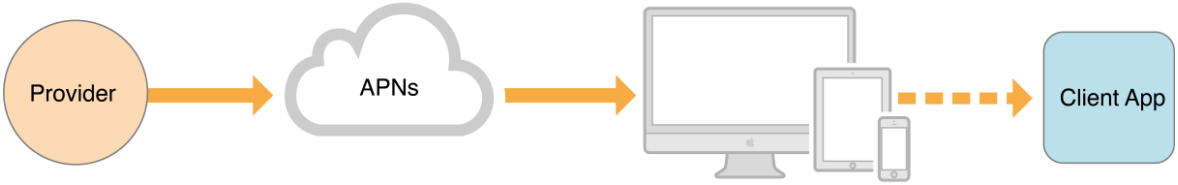


Obr. 13. Architektura Google Cloud Messaging [26]

Přes GCM je možná i komunikace od zařízení (tedy aplikace) k aplikačnímu serveru – například pro odesílání zprávy v chatu. Zařízení musí být registrováno pro příjem notifikací. Při registraci dostane unikátní identifikátor nazvaný registrační token. Pomocí tohoto tokenu pak lze zasílat notifikace na konkrétní zařízení [26].

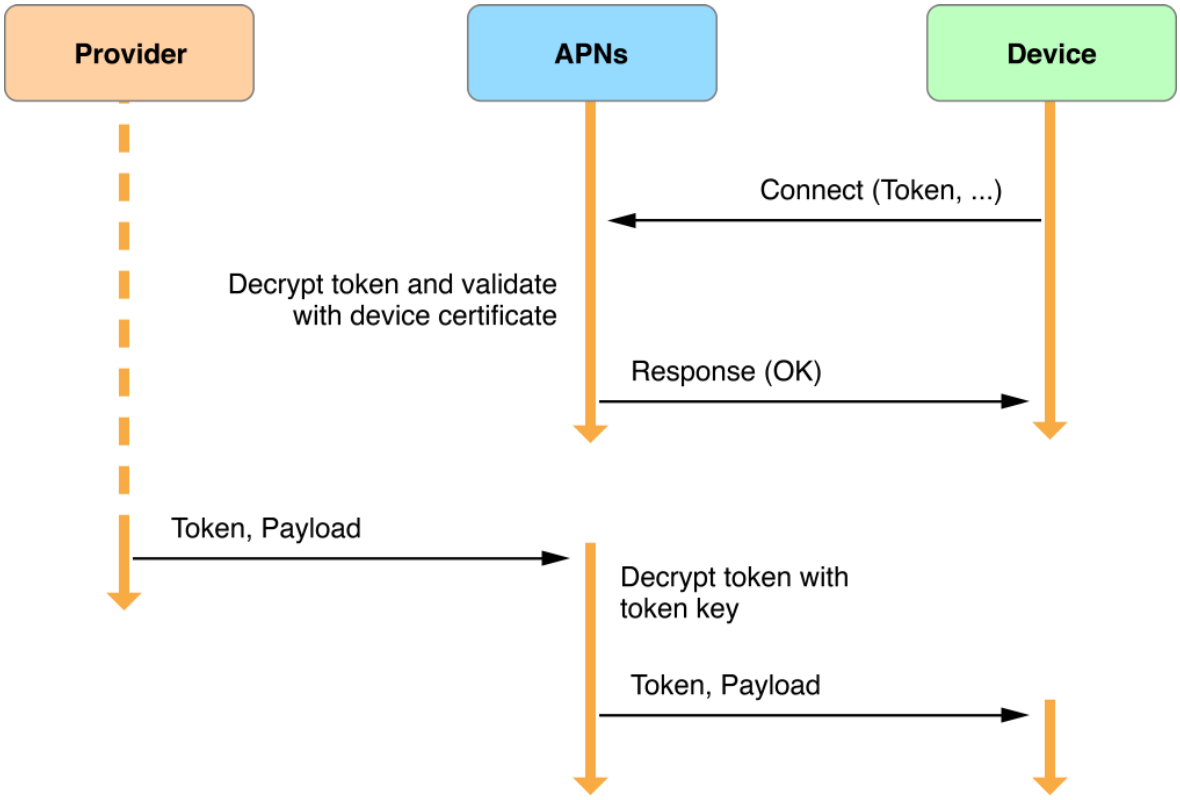
Od roku 2017 společnost Google doporučuje přechod na nový systém multiplatformního zpracování notifikací s názvem Firebase [27].

5.2.2 Push notifikace v operačním systému iOS



Obr. 14. Příjem push notifikací od odesílatele do aplikace [28]

Zpracování push notifikací v operačních systémech společnosti Apple probíhá přes službu Apple Push Notification Service (dále APN), což je serverová služba, která zprostředkovává zabezpečené a efektivní doručení notifikace do systémů iOS, tvOS a macOS [28]. Při prvním spuštění aplikace, která podporuje push notifikace, se systém přihlásí na APN k odběru notifikací, které přijímá ve formátu JSON s maximální velikostí 4KB [29]. K příjmu push notifikací aplikace potřebuje Apple Push Notification Authentification Key certifikát, který vygeneruje autor aplikace pomocí svého vývojářského účtu. Aplikace odebírající push notifikace v konkrétním zařízení je identifikována pomocí Device Tokenu, který získá po ověřeném připojení k APN. Jakmile mobilní aplikace získá Device Token od APN, může jej předat serverové aplikaci jako unikátní identifikátor pro zasílání push notifikací na dané zařízení [29].



Obr. 15. Zaslání push notifikace od serverové aplikace do mobilní aplikace [28]

II. PRAKTICKÁ ČÁST

6 ROZBOR ARCHITEKTURY PROJEKTU

Architektura projektu je základním kamenem celé práce. Nevhodně zvolená architektura by mohla vést k řadě problémů jako je například zdlouhavá implementace, špatná udržitelnost kódu, nebo problémy chodu jednotlivých částí projektu. Návrh architektury vychází z požadavků projektu.

6.1 Požadavky projektu

Požadavkem projektu je multiplatformní mobilní aplikace pro rychlou komunikaci se studenty. Jako rychlou komunikaci rozumíme odesílání a příjem zpráv v reálném čase. K tomu slouží v mobilním světě push notifikace, které jsou popsány v teoretické části. Aplikace musí umožňovat dva následující scénáře:

1. Odeslání rychlé zprávy studentům
2. Odeslání požadavku na hodnocení rozvrhové akce

Pro odeslání rychlé zprávy studentům učitel v aplikaci vybere rozvrhovou akci, u které vybere zaslání zprávy, následně přidá svoji zprávu, případně vybere seznam studentů, kterým má být zpráva doručena a zároveň mají tuto rozvrhovou akci v rozvrhu, a odešle zprávu. Studentům následně dorazí zpráva formou push-notifikace, takže doručení je téměř okamžité.

Pro odeslání požadavku na hodnocení rozvrhové akce učitel vybere rozvrhovou akci, u které zvolí požadavek na hodnocení, následně nastaví heslo k hodnocení a případně upraví seznam studentů, kterým má být požadavek na hodnocení doručen a zároveň mají tuto rozvrhovou akci v rozvrhu. Studentům následně dorazí zpráva formou push-notifikace, v jejímž detailu naleznou formulář pro odeslání hodnocení. Učitel poté uvidí hodnocení v detailu hodnocení výuky.

Z požadavků plyne, že aplikace potřebuje back end s API rozhraním pro posílání a příjem dat. Back end musí umožňovat zasílání push notifikací. Některá data, jako například rozvrhy, je potřeba brát ze STAG API. Serverový back end nakonec potřebuje DB k ukládání dat.

6.1.1 Multiplatformní mobilní aplikace

Pro vývoj multiplatformních mobilních aplikací existuje několik frameworků, které jsou popsány v teoretické části. Jako nejlepší volba se jeví framework Xamarin, ve kterém je možno využít maximum sdíleného kódu jako například celé sdílené GUI. Jedinou věcí, která

je potřeba implementovat pro každou platformu zvlášť, jsou push-notifikace. Alternativou by mohl být například Flutter.

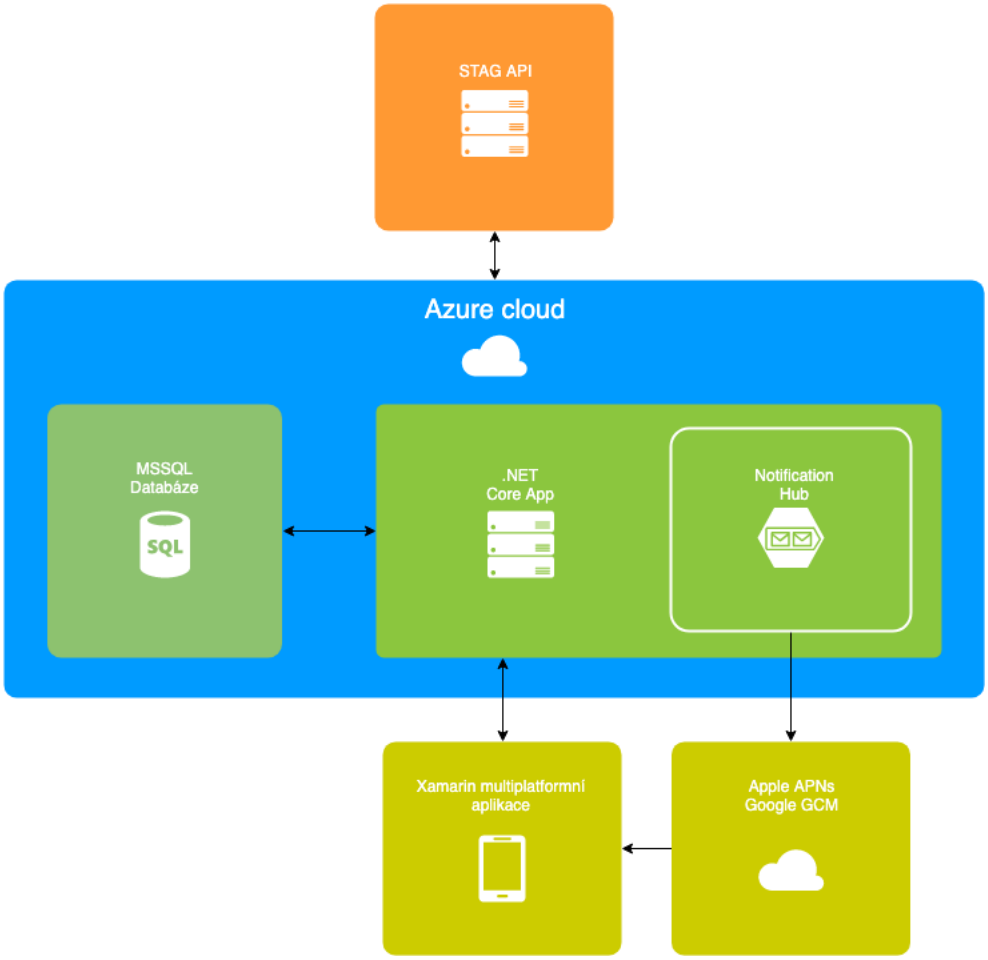
6.1.2 Cloudový back end

Implementovat serverovou části projektu by bylo možné v libovolném jazyce jako například PHP, C++ nebo python, ale je potřeba brát ohled na implementaci push-notifikací, komunikaci s DB, nebo taky hosting serveru. Těmto požadavkům nejlépe vyhovuje .NET Core serverová služba, kterou lze implementovat v jazyce C#. Databázi MSSQL lze jednoduše navázat ke službě. O push notifikace se postará framework Notification Hub, který lze použít taktéž v jazyce C#. Pro hosting celé této služby se nabízí cloudová platforma Azure, jelikož poskytuje všechny tyto možnosti na jednom místě s okamžitou instalací a online podporou.

6.2 Návrh architektury

Serverová část bude postavena na technologii .NET Core s využitím jazyka C#, která bude hostována na cloudové službě Azure. To umožní implementovat v serverové aplikaci Notification Hub, který se stará o zasílání push notifikací. Notification Hub zajistí komunikaci s notifikačními servery společnosti Google a Apple. Dále Azure nabízí jednoduchou správu MSSQL databáze pro uchování dat. Komunikace s IS STAG bude probíhat pomocí STAG REST API, kde jednotlivé příkazy budou volány a zpracovány přímo z .NET Core aplikace.

Multiplatformní mobilní aplikace bude postavena na technologii Xamarin taktéž s využitím jazyka C#. Aplikace bude komunikovat přes jednotné REST API implementované v .NET Core aplikaci. Požadavky na STAG zařídí toto API.



Obr. 16. Návrh architektury projektu

6.3 Popis navrhnuté serverové služby

6.3.1 Serverová služba

Serverová služba se stará o:

- Komunikaci s klientem (Xamarin mobilní aplikace)
- Komunikace se STAG API (prvotní ověřování uživatele, stahování rozvrhů)
- Ukládání dat do vlastní DB (ukládání rozvrhů svázaných s identifikátorem pro push notifikace)
- Odesílání push notifikací pomocí Notification Hub

Pro vytvoření serverové části projektu byl zvolen Framework .NET Core, který je určen pro serverovou činnost. Zvolen byl z důvodu snadného nasazení ve službě Azure [30] a vývoji v jazyce C# a podpoře multiplatformního nástroje pro odesílání push notifikací – Notification Hub. Alternativou by mohla být serverová aplikace běžící na serveru s OS Linux

napsaná v jazyce PHP, nebo python a pro odesílání push notifikací by mohla sloužit služba Firebase [27].

6.3.2 Hosting serveru

Pro hosting serverové služby byl zvolen cloud Azure od společnosti Microsoft. Nasazení služby probíhá z vývojového prostředí Visual Studio.

Důvody výběru hostingu:

- jednoduché nasazení serverové aplikace na cloud
- provázanost cloudu s vývojovým prostředím Visual Studio, které umožňuje testování a nasazení služby přímo z GUI
- rychlá konfigurace výkonu serveru
- snadné nasazení Notification Hub z prostředí Azure
- financování Pay-As-You-Go

Nevýhody hostingu:

- majitel nemá chod serveru pod fyzickou kontrolou, aplikace běží v cloudu
- závislost na cenové politice Azure

6.3.3 Komunikace se STAG API

Komunikace mezi serverovou službou a STAG probíhá pomocí REST API [31]. Data jsou zpracována ve formátu JSON. Probíhá zde ověřování studentů, učitelů, stahování jejich rozvrhů a stahování informací o rozvrhových akcích.

Seznam použitých služeb:

getRozvrhByStudent

Adresa: /ws/services/rest2/rozvrhy/getRozvrhByStudent

Vrátí rozvrh studenta jako rozvrhové akce podle jeho osobního čísla.

getRozvrhByUcitel

Adresa: /ws/services/rest2/rozvrhy/getRozvrhByUcitel

Vrátí rozvrh učitele jako rozvrhové akce podle jeho ID učitele.

getRozvrhovaAkceInfo

Adresa: /ws/services/rest2/rozvrhy/getRozvrhovaAkceInfo

Vrátí rožšířené info o rozvrhové akci podle ID rozvrhové akce.

getUciteleRoakce

Adresa: /ws/services/rest2/rozvrhy/getRozvrhByStudent

Vrátí rozvrh studenta jako rozvrhové akce podle jeho osobního čísla.

6.3.4 Notification Hub

O zasílání push notifikací se stará služba Notification Hub, která běží na cloudové službě Azure. Serverová služba používá knihovny Notification Hub pro odesílání push notifikací nezávisle na cílové platformě (iOS, Android, Windows, Amazon).

Výhodou Notification Hub:

- sdružení zasílání push notifikací na různé cílové platformy pod jednu službu
- jednoduchá a rychlá implementace
- možnost zasílání push notifikací jak konkrétním zařízením, tak broadcast
- statistiky odesílání/přijímání push notifikací v prostředí portálu Azure

Nevýhody:

- závislost na cloudové službě Azure a na jazyku C#
- nepraktické pro zasílání push notifikací na jednu cílovou platformu

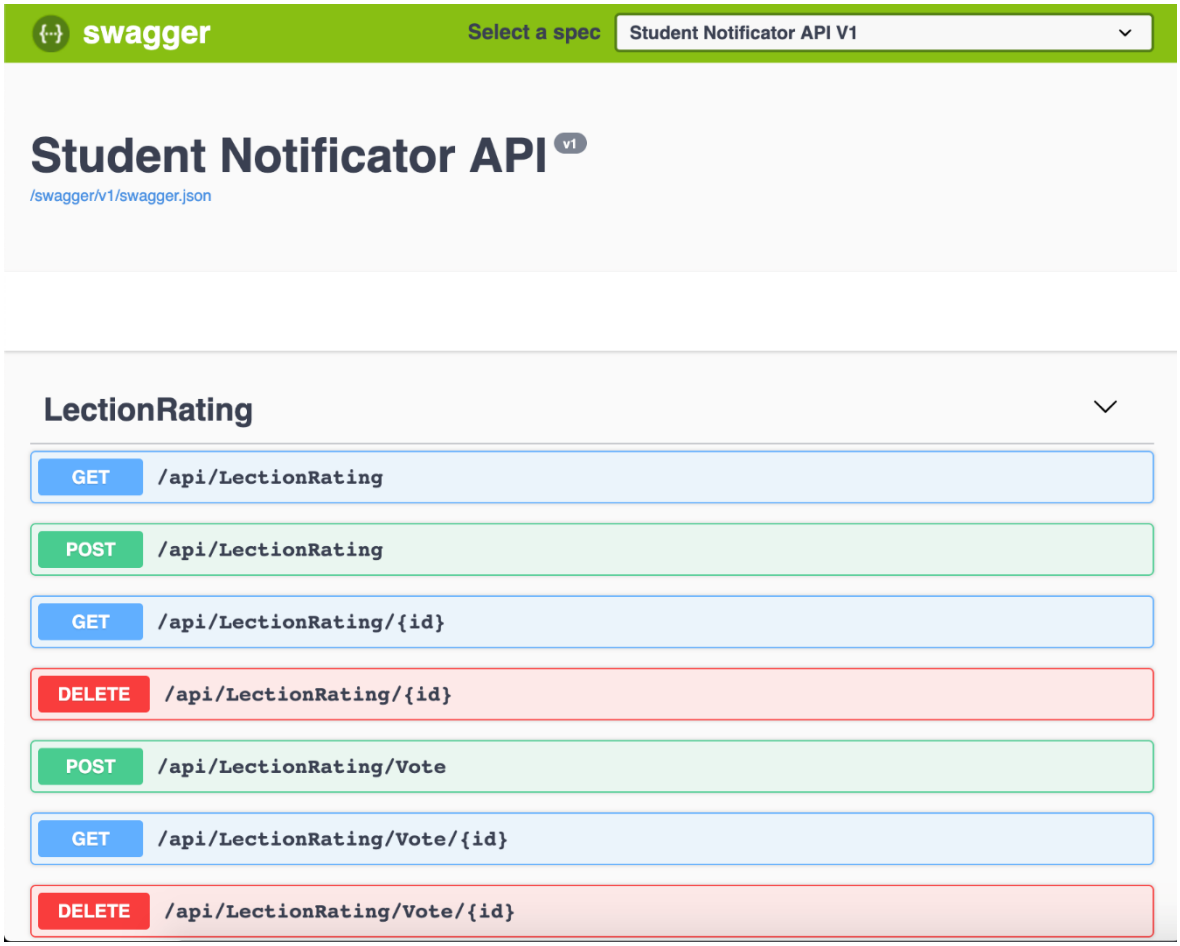
Jako alternativa pro zasílání push notifikací by mohla být použita služba Firebase od společnosti Google, případně implementace pro jednotlivé platformy zvlášť, která by ale byla velmi náročná a zdlouhavá.

7 NÁVRH SERVEROVÉ ČÁSTI PROJEKTU

Serverová část projektu se stará o ověřování uživatelů, stahování rozvrhů, rozesílání zpráv pomocí push notifikací a vyhodnocování hodnocení výuky. Mobilní aplikaci poskytuje rozhraní pomocí REST API. Serverová část je postavena na technologii .NET Core, která slouží právě pro serverové účely [32].

7.1 .NET Core služba

.NET Core služba slouží jako API rozhraní pro mobilní aplikaci a zprostředkovává rozesílání push notifikací uživatelům. Je napsána pomocí MVC architektury, tedy Model-View-Controller, kde Model reprezentuje data, View zobrazení dat a Controller interakci mezi View a Controllerem. View je generován automaticky frameworkem Swagger, který generuje HTML stránku nad REST API, kde je možno přímo spouštět API příkazy a pracovat s výsledky v otevřeném textu. Jednotlivé části API rozhraní jsou přidány v projektu mezi služby jako singleton [33].



Obr. 17. Rozhraní Swagger pro vizualizaci REST API

7.2 REST API StudentsNotifier AppService

REST API StudentsNotifier je jednotné API, které poskytuje veškeré potřebné služby pro mobilní aplikaci. Všechny API příkazy jsou implementovány asynchronně, tak aby bylo možné obsloužit více požadavků ve stejném čase [34]. Poskytuje rozhraní pro inicializaci uživatelů a jejich dat ze STAG API, dále poskytuje rozhraní pro odesílání a příjem zpráv, odesílání a příjem hodnocení výuky a nakonec rozhraní pro inicializaci a odesílání push notifikací.

Seznam služeb, které poskytuje .NET Core služba StudentsNotifier:

1. **Message** – posílání a čtení zpráv

   [GET] /api/Message/{id}

   - Vrátí zprávu podle ID.

   [POST] /api/Message

   - Vloží zprávu do kolekce zpráv a odešle zprávu formou push notifikace příslušným studentům podle jejich ID.

   [GET] /api/Message

   - Vrátí seznam všech zpráv. Použito pouze pro správu a ladění.

   [DELETE] /api/Message/{id}

   - Smaže zprávu podle ID. Použito pouze pro správu a ladění.

   [GET] /api/Message/UserMessages/{id}

   - Vrátí seznam všech zpráv daného uživatele podle jeho ID.

Tělo JSON zprávy pro odeslání zprávy uživatelům obsahuje jedinečné ID, čas odeslání, text zprávy, jméno odesílatele a pole notifikačních tokenů, na které se má zpráva odeslat.

```
{
   "id" : "12345678",
   "dateTime" : "2019-05-12T09:29:16.925Z",
   "messageText" : "Zde je tělo zprávy",
   "userIds:" : [
      " APA91bHun4MxP5egoKMwt2KZFBaFUH-1RYqx...“,
      " APA91bHun4MxP5egoKMwt2KZFBaFUH-2ABw4..."
   ],
   "messageFrom" : "Satoshi Nakamoto",
   "sendMessageResult:" : true
}
```

Kód 1. Příklad JSON těla pro odesílání zpráv uživatelům

2. **User** – vytváření a čtení uživatelů

   [GET] /api/User/{id}

   - Vrátí uživatele podle jeho ID.

   [POST] /api/User

   - Vloží nového uživatele včetně tokenu pro odběr push notifikace.

   [GET] /api/User

   - Vrátí všechny uživatele. Použito pouze pro správu a ladění.

   [DELETE] /api/User/{id}

   - Smaže zprávu podle ID. Použito pouze pro správu a ladění.

   [GET] /api/User/RozvrhoveAkce/{id}

   - Vrátí rozvrhové akce uživatele podle ID. Na rozvrhové akce se dotáže ze STAG API podle osobního čísla uživatele, které je nastaveno v instanci uživatele.

Tělo JSON zprávy pro vložení nového uživatele obsahuje jeho jedinečné ID, řetězec s jeho celým jménem, roli (odpovídá rolím na IS STAG, tedy student, nebo vyučující), STAG ID neboli osobní číslo a notifikační token vygenerovaný z fyzického zařízení daného uživatele pro příjem push notifikací.

```
{
   "id" : "12345678",
   "name" : "Satoshi Nakamoto",
   "role" : "ST",
   "stagID:" : "A12345",
   "notificationToken" : "APA91bHun4MxP5egoKMwt2KZFBaFUH-1RYqx..."
}
```

Kód 2. Příklad JSON těla zprávy pro vložení nového uživatele

3. **LectionRation** – hodnocení výuky

   [GET] /api/LectionRating

   - Vrátí všechny hodnocení výuky.

   [POST] /api/LectionRating

   - Vytvoří nové hodnocení výuky

   [GET] /api/LectionRating/{id}

   - Vrátí hodnocení výuky podle ID.

   [DELETE] /api/LectionRating/{id}

   - Odstraní hodnocení výuky podle ID.

   [POST] /api/LectionRating/Vote

   - Vloží nové hodnocení pro konkrétní instanci hodnocení výuky.

   [GET] /api/LectionRating/Vote/{id}

   - Vrátí konkrétní hodnocení podle ID.

   [DELETE] /api/LectionRating/Vote/{id}

   - Odstraní konkrétní hodnocení podle ID.

   [POST] /api/LectionRating/SendVoteRequest

   - Odešle vybraným uživatelům notifikaci s požadavkem na hodnocení výuky.

Tělo JSON zprávy pro vložení nového hodnocení výuky obsahuje jedinečné ID, ID lekce neboli rozvrhové akce, kód pro hlasování (jeli potřeba), pole s hlasováním a čas v sekundách, který určuje životnost tohoto hodnocení, které po vypršení zmizí.

```
{
   "id" : "12345678",
   "lectionID" : "123456789",
   "voteCode" : "ahoj",
   "votes:" : [
      {}
   ],
   "durationInSec" : "120"
}
```

Kód 3. Příklad JSON těla zprávy pro požadavek na hodnocení výuky

Tělo JSON zprávy pro vložení nového hodnocení poté obsahuje jedinečné ID, ID lekce které hodnocení patří a hodnocení uživatele. Samotné hodnocení je tedy anonymní.

```
{
   "id" : "12345678",
   "lectionRatingID" : "123456789",
   "userVote" : 1
}
```

Kód 4. Příklad JSON těla zprávy pro hodnocení výuky

4. **PushNotifications** – zasílání push notifikací

   [GET] /api/Notifications/Register

   - Zaregistruje zařízení k odběru push notifikací a vytvoří mu ID.

   [DELETE] /api/Notifications/Unregister

   - Odebere zařízení možnost odebírat push notifikace podle ID.

   [PUT] /api/Notifications/Enable/{id}

   - Nastaví odběr push notifikací podle ID.

   [POST] /api/Notifications/Send

   - Odešle push notifikaci na konkrétní zařízení.

Tělo JSON zprávy pro odeslání notifikace obsahuje prostor pro zprávu, volbu cílové platformy, token konkrétního zařízení, které má přijmout notifikaci a tagy pro hromadné odesílání notifikací.

```
{
   "content" : "{"aps":{"alert":"Test notifikace"}}",
   "platform" : "apns",
   "handle" : "APA91bHun4MxP5egoKMwt2KZFBaFUH-1RYqx...",
   "tags:" : [
      "studenti“,
      "vyucujici"
   ]
}
```

Kód 5. Příklad JSON těla zprávy pro odeslání push notifikace

8 NÁVRH XAMARIN APLIKACE A UŽIVATELSKÉHO PROSTŘEDÍ

Uživatelské rozhraní využívá návrhový vzor MVVM, tedy Model-View-ViewModel. Model se stará o uchování dat, View se stará o zobrazení dat a ViewModel o interakci mezi uživatelským rozhraním a daty. View je v drtivé většině případů realizován pomocí XAML kódu, který značně připomíná značkovací jazyky jako například HTML [35].

Pro práci se serverovou částí slouží rozhraní IDataStore, které popisuje všechny metody potřebné pro přístup k datům, jako je například požadavek na rozvrhy uživatele, nebo zaslání zprávy. Konkrétní implementaci tohoto rozhraní poté implementuje třída AzureDataStore, která pak volá konkrétní API příkazy ze serverové části běžící na cloudu Azure.

8.1 Návrh uživatelského prostředí

V aplikaci je potřeba zobrazit několik věcí:

- Stránka pro přihlášení uživatele
- Seznam rozvrhových akcí
- Seznam zpráv

Klíčovým prvkem uživatelského prostředí je jednoduchost a rychlost předání informace od učitele ke studentům. Učitel po přihlášení do aplikace uvidí svůj rozvrh, kde kliknutím na detail rozvrhové akce může odeslat studentům buďto rychlou zprávu, nebo požadavek na hodnocení výuky. Pro zobrazení seznamu rozvrhových akcí a seznamu zpráv poslouží komponenta TableView, která umí zobrazit seznam prvků [16] [36].

8.1.1 Stránka pro přihlášení uživatele

Stránka pro přihlášení uživatele obsahuje pole, kde uživatel zadá svoje Osobní číslo a tlačítko pro stažení dat z IS STAG. O všechno ostatní už se postará serverová část, která stáhne rozvrhy a vrátí je aplikaci.

8.1.2 Seznam rozvrhových akcí

Stránka se seznamem rozvrhových akcí zobrazuje všechny rozvrhové akce uživatele, které si po přihlášení stáhla z IS STAG. Rozvrhové akce jsou zobrazeny podle času se zobrazením typu rozvrhové akce (přednáška, cvičení, seminář, zkouška). Po kliknutí na rozvrhovou akci se zobrazí detail s možnostmi pro zaslání rychlé zprávy, nebo požadavku na hodnocení

výuky. Dále se zobrazí seznam studentů, kteří jsou přihlášení v aplikaci a mají tuto rozvrhovou akci také v rozvrhu.

8.1.3 Seznam zpráv

Stránka se seznamem zpráv zobrazuje seřazené přijaté zprávy od nejnovější po nejstarší. Jednotlivé zprávy ve svém těle v seznamu zobrazují odesílatele, čas přijetí a náhled textu. Pokud by byl text příliš dlouhý, tak lze zobrazit v detailu zprávy. V detailu zprávy je vidět odesílatel, čas odeslání a text zprávy.

8.1.4 Seznam hodnocení výuky

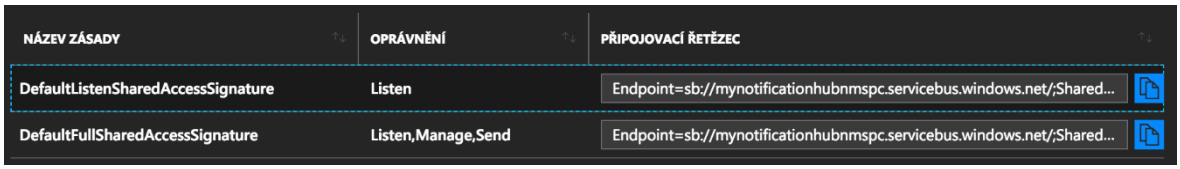
Stránka se seznamem hodnocení výuky zobrazuje probíhající hodnocení výuky. Jednotlivé hodnocení výuky jsou pojmenovány podle předmětu, na kterém hodnocení probíhá. V detailu hodnocení lze najít průměrné hodnocení, maximální a minimální udělené hodnocení a počet udělených hodnocení.

9 NOTIFIKACE

V projektu je potřeba využít pouze push notifikace pro real-time příjem zpráv. Lokální notifikace nejsou v této aplikaci potřeba, protože by umožňovaly pouze periodickou kontrolu, jestli uživatele nečeká nová zpráva. Požadavkem aplikace je vše okamžitá komunikace, proto je potřeba využít push notifikací. Pro fungování push notifikací je potřeba vyřešit několik věcí. Push notifikace je potřeba implementovat jak na serverové části, tak na klientské – tedy v mobilní aplikaci.

9.1 Notification Hub

Na portálu cloudu Azure je potřeba vytvořit Notification Hub, tedy vytvořit novou instanci softwarového balíčku, který bude zpracovávat push notifikace. Pro aktivaci služby pro iOS je potřeba v nastavení vložit vygenerovaný certifikát, u androidu Server Key vygenerovaný z Firebase Messaging Cloud pro odběr push notifikací [37]. Dále je potřeba nastavit zásady přístupu. Připojovací řetězec se následně používá v mobilní aplikaci při prvním spuštění, kde probíhá konfigurace odběru push notifikací.



Obr. 18. Připojovací řetězec Notification Hub pro odběr push notifikací v iOS

9.2 Xamarin aplikace

V Xamarin.iOS projektu je potřeba pro iOS přidat aplikaci provisioning profil, který umožňuje vývojáři využívat služby k přijímání push notifikací na vývojářském zařízení. Dále je potřeba přidat do projektu NuGet balíček NotificationHub a implementovat metody pro zaregistrování push notifikací, příjem a zpracování push notifikací. Každé zařízení pak má unikátní device token, který slouží k identifikaci konkrétního zařízení. Pro konfiguraci Notification Hubu je potřeba vložit připojovací řetězce vygenerované v Azure.

Pro Xamarin.Android je potřeba vygenerovat konfigurační balíčky přes Firebase Messaging Center (FMC) službu společnosti Google. Taktéž je potřeba přidat do projektu NuGet balíček Xamarin.GooglePlayServices.Base a vytvořit notifikační kanál. Notification Hub se propojí stejným připojovacím řetězcem, jako v případě iOS [38].

9.2.1 Formát obsahu notifikace pro iOS

Obsah notifikace je definován ve formátu JSON, který nese několik důležitých informací pro systém. Některé informace se přímo zobrazují, jako například titulek, nebo text těla notifikace, je možné však využít například kategorií pro kategorizaci notifikace, nebo vlastních informací až do velikosti 4KB [39].

Příklad notifikace pro iOS:

```
{
   "aps" : {
      "alert" : {
         "title" : "Nová zpráva",
         "subtitle" : "Od: Satoshi Nakamoto",
         "body" : "Dobrý den, hodina odpadá."
      },
      "category" : "MESSAGE"
   },
   "messageID" : "12345678"
}
```

Kód 6. Příklad JSON těla notifikace pro iOS

9.2.2 Formát obsahu notifikace pro Android

Obsah notifikace je definován ve formátu JSON stejně jako u iOS, který je rozdělen na dvě části – tělo notifikace a data. Tělo notifikace má maximální velikost 2KB a data mohou být mít velikost až 4KB. Notifikací pro Android lze nastavit priorita, která ovlivňuje rychlost zpracování notifikace na GCM serverech. Priority se typicky nastavuje vyšší pro aplikace sloužící jako zprostředkování hovorů, nebo chat, kde je potřeba doručovat zprávy bez odezvy [40].

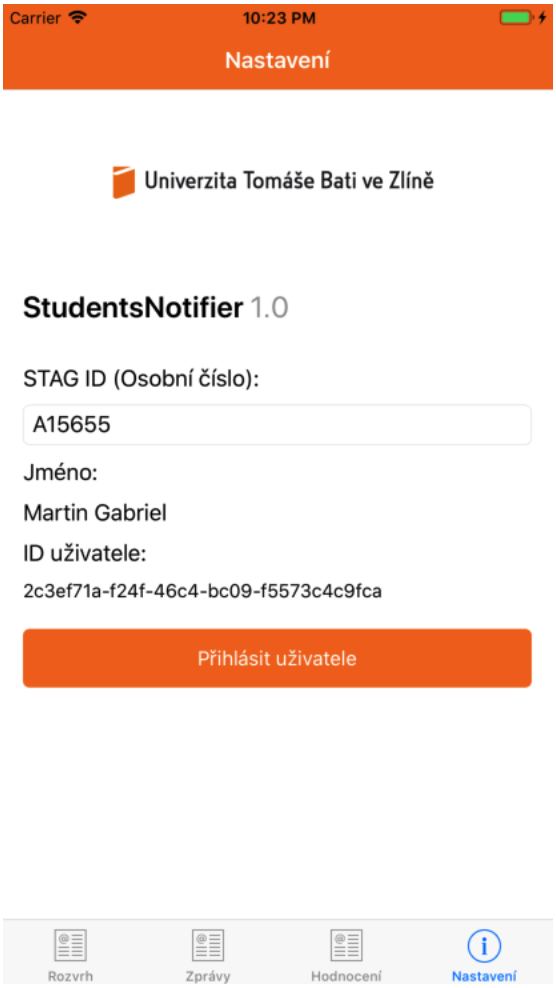
Příklad notifikace pro Android:

```
{
   "to" : "APA91bHun4MxP5egoKMwt2KZFBaFUH-1RYqx...",
   "notification" : {
      "body" : "Dobrý den, hodina odpadá.",
      "title" : "Nová zpráva",
      "icon" : "mojeikona"
   },
   "data" : {
      "From" : "Satoshi Nakamoto",
      "MessageID" : "12345678",
      "Date" : "2019-05-11"
   }
}
```

Kód 7. Příklad JSON těla notifikace pro Android

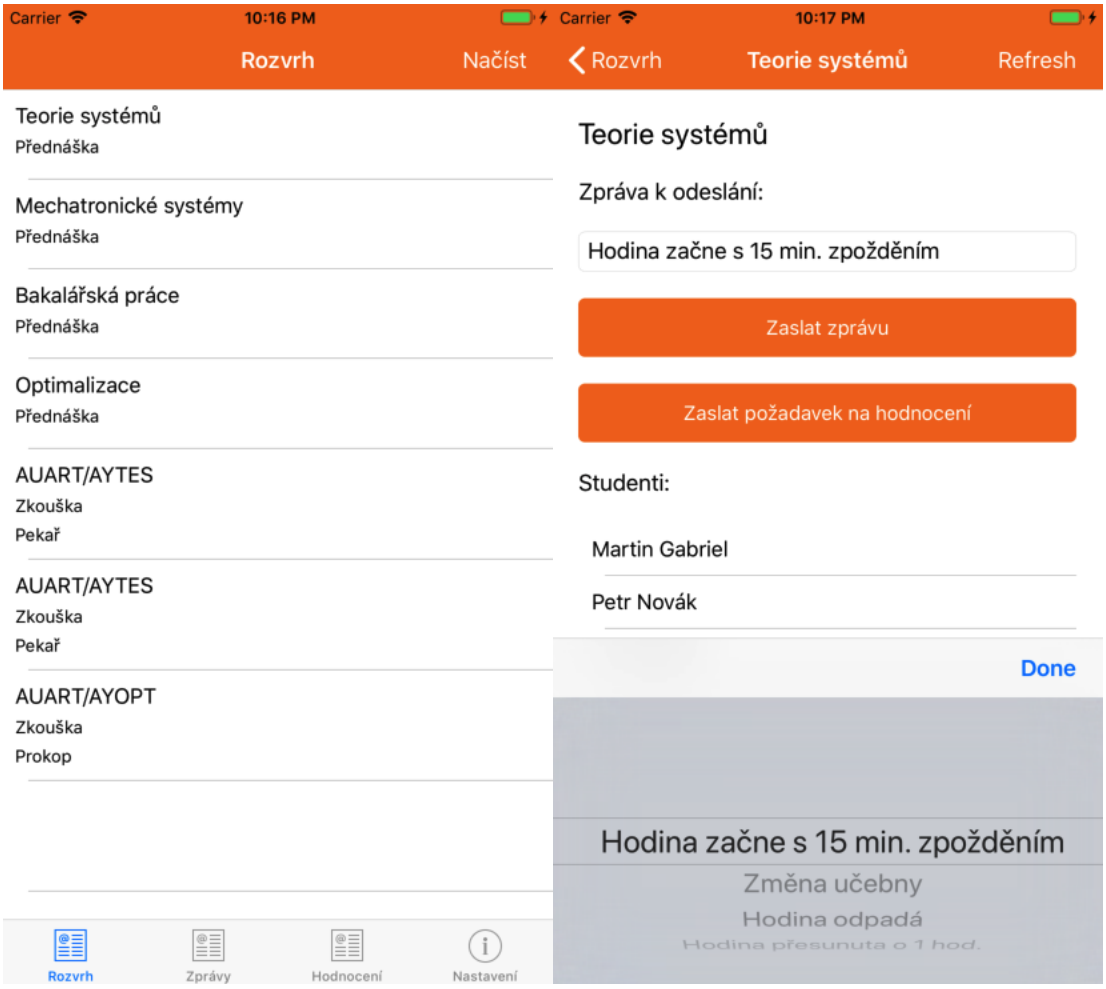
10 PŘÍKLAD VYUŽITÍ APLIKACE

Po spuštění aplikace je potřeba, aby uživatel nastavil jeho STAG ID, neboli osobní číslo, díky kterému si aplikace stáhne rozvrh. Student i učitel se přihlašují stejným způsobem – zadáním svého osobního čísla ze STAGu a kliknutím na Přihlásit uživatele. Server uživateli vygeneruje jedinečné ID a stáhne potřebná data. Následující obrázky zobrazují pohled přihlášeného učitele. Student i učitel mají v aplikaci stejné možnosti zobrazení, student pouze nemůže odesílat zprávy ostatním studentům a nemůže odesílat požadavek na hodnocení výuky – to může pouze učitel.



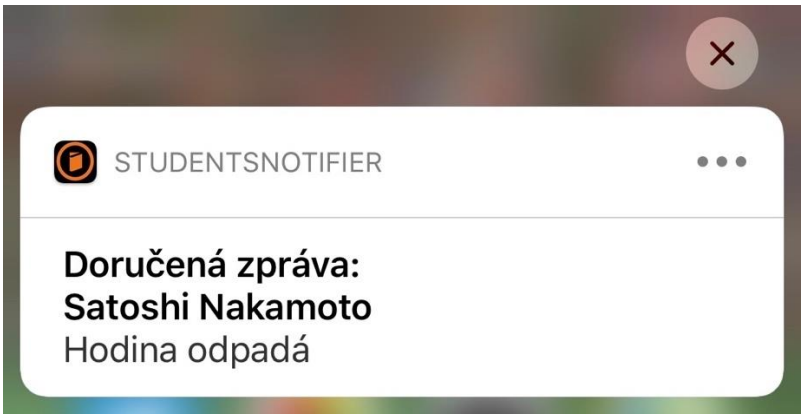
Obr. 19. Stránka nastavení

Po kliknutí na tlačítko pro stažení dat z IS STAG zavolá aplikace API příkaz, který z IS STAG stáhne rozvrh daného uživatele, zároveň zajistí synchronizaci zařízení pro odběr push notifikací ze serverové části. Aplikace spuštěná v režimu emulátoru nebude přijímat push notifikace, protože ty jsou vázány na reálné zařízení. Uživatel poté může přejít na záložku Rozvrh, kde uvidí svoje aktuální rozvrhové akce.



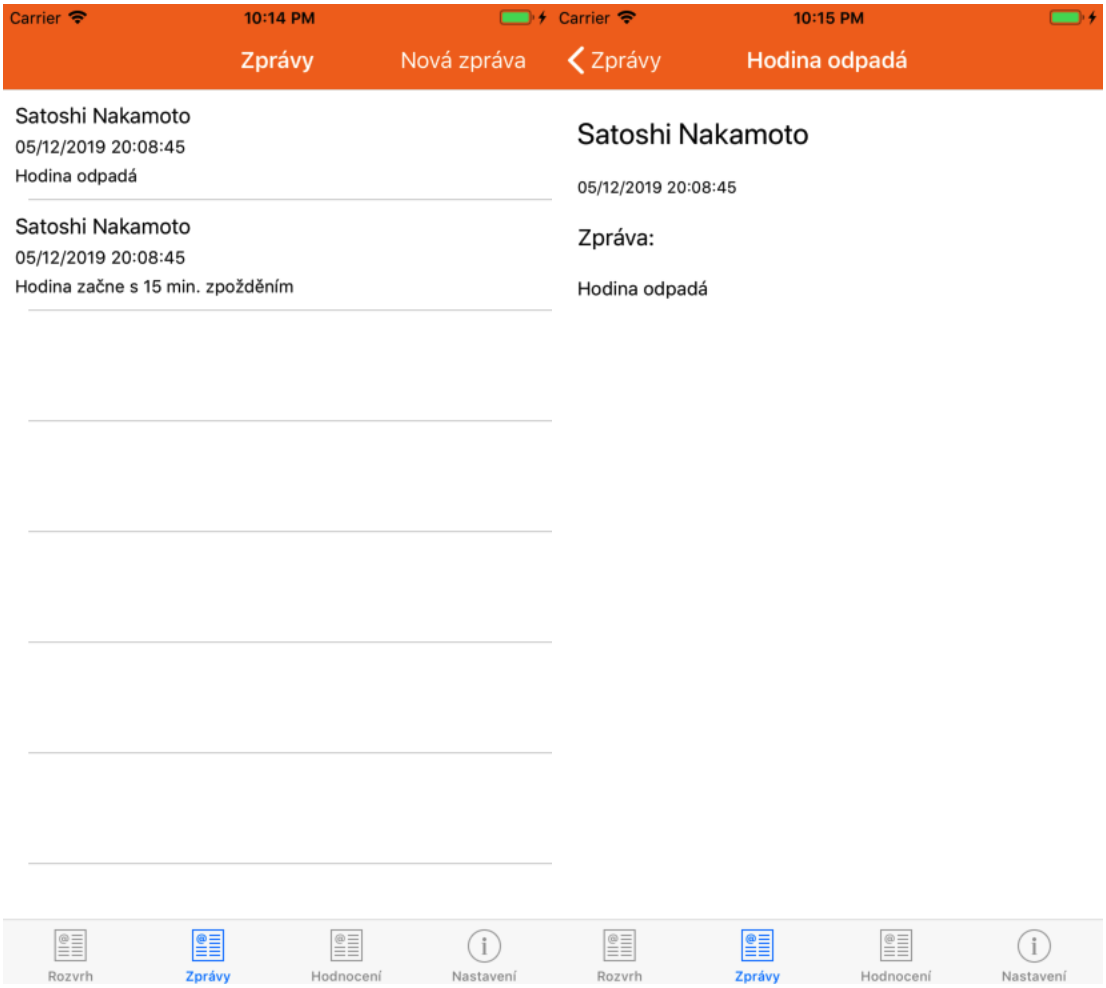
Obr. 20. Seznam rozvrhových akcí s detailem akce

Při zobrazení detailu rozvrhové akce může učitel vybrat a odeslat rychlou zprávu všem uživatelům ze seznamu, případně odebrat ty, kterým nechce zprávu odeslat. V seznamu se zobrazují pouze uživatelé, kteří jsou přihlášení v aplikaci, respektive mají v aplikaci synchronizovaný rozvrh. Kliknutím na odeslání zprávy dojde k odeslání push notifikací na cílová zařízení. Na cílovém zařízení se příjem zprávy projeví v notifikačním centru.



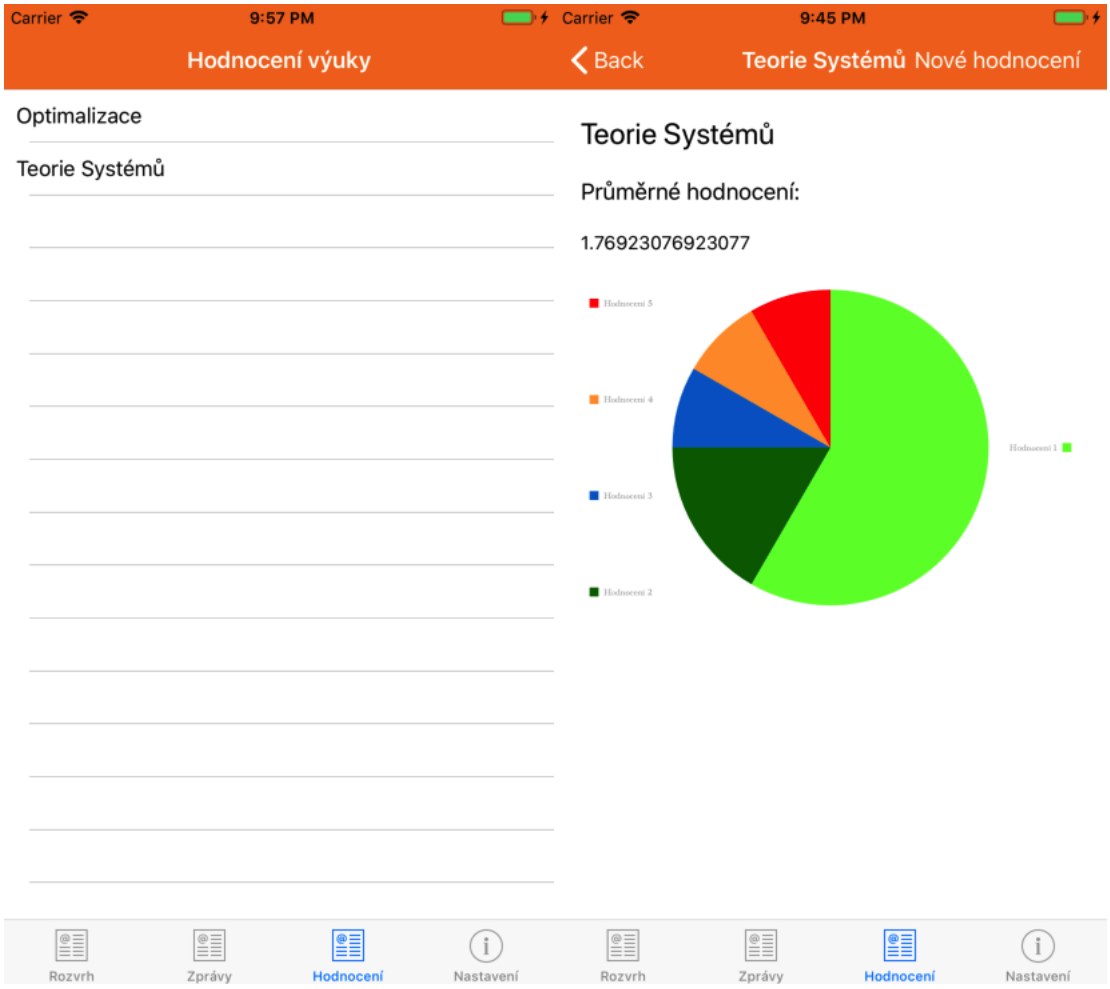
Obr. 21. Doručení push notifikace s detailem zprávy

V záložce Zprávy najde uživatel seznam přijatých zpráv seřazených podle času přijetí, takže nejnovější budou na vrcholu. Po kliknutí na konkrétní zprávu se zobrazí její detail.



Obr. 22. Seznam zpráv s detailem zprávy

V záložce Hodnocení výuky najde uživatel seznam všech probíhajících hodnocení. V detailu hodnoceného předmětu najde možnost hlasovat, přičemž hlasování je anonymní a lze hlasovat vícekrát.



Obr. 23. Hodnocení výuky s detailem vyhodnocení

Detail hodnocení výuky zobrazuje název hodnoceného předmětu, průměrné hodnocení a četnost hodnocení v grafu. Dále umožňuje odeslat nové hodnocení výuky s jednoduchým výběrem hodnocení 1 až 5, kde 1 je nejlepší hodnocení a 5 nejhorší.

ZÁVĚR

Vytvořená multiplatformní aplikace splňuje všechny body zadání. V teoretické části byly popsány mobilní platformy Android, iOS a některé další. Dále byl popsán multiplatformní vývoj, vývoj v multiplatformním frameworku Xamarin, Azure cloud s nasazením .NET Core služby a nakonec notifikace na mobilních platformách. V praktické části se podařilo navrhnout architekturu pro multiplatformní mobilní aplikaci a její serverovou část. V ukázkové mobilní aplikace byly zprovozněny push notifikace pro okamžité odesílání zpráv, bylo implementováno zasílání zpráv na základě rozvrhových akcí a hodnocení výuky. Zvolená architektura, postavená na technologiích společnosti Microsoft, má velkou výhodu v jednoduchosti a rychlosti implementace, dále také ve vývojářské podpoře, dokumentaci, nebo bezproblémovosti jejich cloudového řešení.

V průběhu práce na serverové části jsem narazil jenom na několik úskalí s cloudem Azure. Jedním z problému bylo automatické vypínání serverové části po určitém čase i přes běžící vlákna. Dalším problémem byly problémy s fakturací nevyužitých služeb, které ale byly rychle vyřešeny s podporou. Nakonec nedošlo ani k použití Azure Functions, jelikož serverová část potřebuje být více provázaná a nenašel jsem využití pro tuto službu, i když jsem se o to prvotně snažil.

Aplikaci je možné dále rozšiřovat z pohledu uživatelského rozhraní, lepších možností na hodnocení výuky, přidání anglické lokalizace, šifrování zpráv a celkového zlepšení zabezpečení, nebo provázanosti s oficiální UTB aplikací.

SEZNAM POUŽITÉ LITERATURY

[1] Mobile Operating System Market Share Worldwide. *Statscounter* [online]. [cit. 2019-05-13]. Dostupné z: http://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-200901-201905

[2] *Platform Architecture* [online]. [cit. 2019-05-13]. Dostupné z: https://developer.android.com/guide/platform

[3] *Android platform versions* [online]. [cit. 2019-05-10]. Dostupné z: https://developer.android.com/about/dashboards/index.html

[4] *What is the architecture of iOS* [online]. [cit. 2019-05-13]. Dostupné z: https://intellipaat.com/tutorial/ios-tutorial/ios-architecture/

[5] *Apple iOS* distribution [online]. [cit. 2019-05-13]. Dostupné z: https://developer.apple.com/support/app-store/

[6] Mozilla definitivně končí s Firefox OS, zaměří se na prohlížeč. *Root.cz* [online]. 2016 [cit. 2019-05-14]. Dostupné z: https://www.root.cz/clanky/mozilla-definitivne-konci-s-firefox-os-zameri-se-na-prohlizec/

[7] Responsive Web Design. *A LIST APART* [online]. [cit. 2018-03-15]. Dostupné z: http://alistapart.com/article/responsive-web-design

[8] *PhoneGap* [online]. [cit. 2018-03-15]. Dostupné z: https://phonegap.com/

[9] Unity for Mobile. *Unity3D* [online]. [cit. 2018-03-15]. Dostupné z: https://unity3d.com/solutions/mobile?utm_source=unity3d-multiplatform&utm_medium=cta

[10] *Flutter* [online]. [cit. 2018-03-15]. Dostupné z: https://flutter.io/

[11] *Flutter vs Xamarin: A Developer's Perspective* [online]. [cit. 2019-05-12]. Dostupné z: https://blog.codemagic.io/flutter-vs-xamarin-a-developer-s-perspective/

[12] *Microsoft to acquire Xamarin and empower more developers to build apps on any device* [online]. [cit. 2018-03-15]. Dostupné z: https://blogs.microsoft.com/blog/2016/02/24/microsoft-to-acquire-xamarin-and-empower-more-developers-to-build-apps-on-any-device/

[13] *Deliver native Android, iOS, and Windows apps, using existing skills, teams, and code. Xamarin* [online]. [cit. 2018-03-14]. Dostupné z: https://www.xamarin.com/platform

[14] *About Mono* [online]. [cit. 2018-03-15]. Dostupné z: https://www.mono-project.com/docs/about-mono/

[15] *Compatibility. Mono* [online]. [cit. 2018-03-15]. Dostupné z: https://www.mono-project.com/docs/about-mono/compatibility/

[16] PETZOLD, Charles. Creating Mobile Apps with Xamarin.Forms [online]. Redmond, Washington 98052-6399: Microsoft Press, 2016 [cit. 2018-03-15]. ISBN 978-1-5093-0297-0. Dostupné z: https://developer.xamarin.com/guides/xamarin-forms/creating-mobile-apps-xamarin-forms/

[17] *Xamarin.Forms. Xamarin* [online]. [cit. 2018-03-15]. Dostupné z: https://www.xamarin.com/forms

[18] *Xamarin.Android* [online]. [cit. 2018-03-14]. Dostupné z: https://docs.microsoft.com/en-us/xamarin/android/

[19] Get Started with Xamarin.iOS*. Xamarin* [online]. 2018 [cit. 2019-05-13]. Dostupné z: https://docs.microsoft.com/en-us/xamarin/ios/get-started/index

[20] *Co je Azure?* [online]. [cit. 2018-03-16]. Dostupné z: https://azure.microsoft.com/cs-cz/overview/what-is-azure/

[21] *An introduction to Azure Functions* [online]. [cit. 2018-03-15]. Dostupné z: https://docs.microsoft.com/en-us/azure/azure-functions/functions-overview

[22] *Notification Hubs* [online]. [cit. 2018-03-16]. Dostupné z: https://azure.microsoft.com/cs-cz/services/notification-hubs/

[23] *Firebase Cloud Messaging* [online]. [cit. 2018-03-16]. Dostupné z: https://firebase.google.com/docs/cloud-messaging/

[24] Local and Remote Notifications Overview. *Apple Developer Documentation* [online]. [cit. 2019-05-13]. Dostupné z: https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/index.html#//apple_ref/doc/uid/TP40008194-CH3-SW1

[25] *Use notifications on your iPhone, iPad, and iPod touch* [online]. [cit. 2018-03-12]. Dostupné z: https://support.apple.com/en-us/HT201925

[26] Google Cloud Messaging: Overview. *Google Cloud Messaging* [online]. [cit. 2019-05-13]. Dostupné z: https://developers.google.com/cloud-messaging/gcm

[27] *GCM and FCM Frequently Asked Questions* [online]. [cit. 2018-03-14]. Dostupné z: https://developers.google.com/cloud-messaging/faq

[28] APNs Overview. *Apple Developer Documentation* [online]. [cit. 2019-05-13]. Dostupné z: https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW1

[29] *Creating the Remote Notification Payload* [online]. [cit. 2018-03-14]. Dostupné z: https://developer.apple.com/library/content/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/CreatingtheNotificationPayload.html

[30] *Portál Azure* [online]. [cit. 2018-05-05]. Dostupné z: https://portal.azure.com/

[31] *Webové služby nad IS/STAG* [online]. [cit. 2018-05-05]. Dostupné z: https://stag-ws.utb.cz/ws/web/

[32] SMITH, Steve. Architecting Modern Web Applications with ASP.NET Core and Microsoft Azure [online]. Redmond, Washington 98052-6399: Microsoft Corporation, 2019 [cit. 2019-05-12].

[33] SINGLETON, James. ASP.NET Core 1.0 High performance. Birmingham: Packt Publishing, 2016. ISBN 9781785881893.

[34] MAREŠ, Amadeo. 1001 tipů a triků pro C#. Brno: Computer Press, 2008. ISBN 978-80-251-2125-2

[35] DAJBYCH, Václav. Mvvm: model-view-viewmodel [online]. 21.04.2009 [cit. 2019-05-10]. Dostupné z: https://www.dotnetportal.cz/clanek/4994/MVVM-Model-View-ViewModel

[36] *Xamarin.Forms TableView* [online]. 14. 12. 2018 [cit. 2019-05-10]. Dostupné z: https://docs.microsoft.com/cs-cz/xamarin/xamarin-forms/user-interface/tableview

[37] *Tutorial: Push notifications to Xamarin.Android apps using Azure Notification Hubs* [online]. 28. 03. 2019 [cit. 2019-05-09]. Dostupné z: https://docs.microsoft.com/cs-cz/azure/notification-hubs/xamarin-notification-hubs-push-notifications-android-gcm

[38] *Tutorial: Push notifications to Xamarin.Android apps using Azure Notification Hubs* [online]. 01. 05. 2019 [cit. 2019-05-12]. Dostupné z: https://docs.microsoft.com/cs-cz/azure/notification-hubs/xamarin-notification-hubs-push-notifications-android-gcm

[39] *Generating a Remote Notification: Send notifications to the user's device with a JSON payload.* [online]. [cit. 2019-05-11]. Dostupné z:

https://developer.apple.com/documentation/usernotifications/setting_up_a_remote_notification_server/generating_a_remote_notification

[40] *Messaging Concepts and Options* [online]. [cit. 2019-05-11]. Dostupné z: https://developers.google.com/cloud-messaging/concept-options

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

C#	Programovací jazyk
.NET	Framework pro vývoj software
iOS	Mobilní operační systém od společnosti Apple.
macOS	Desktopový operační systém od společnosti Apple.
GCM	Google Cloud
REST	Representational State Transfer
API	Aplikační interface
SQL	Structured Query Language
LINQ	Language Integrated Query
XAML	Extensible Application Markup Language
STAG	Školní informační systém

SEZNAM OBRÁZKŮ

SEZNAM KÓDŮ

SEZNAM PŘÍLOH