Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Diplomová práce

Nástroje pro multiplatformní programování mobilních aplikací

Bc. Jiří Malina

Vedoucí práce: Ing. Miroslav Balík, Ph.D.

4. května 2015

Poděkování

Rád bych poděkoval společnosti Trigama International s.r.o. za zapůjčení zařízení pro platformy iOS a Windows Phone. Dále bych rád poděkoval všem účastníkům uživatelských testů.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 4. května 2015 .....

Odkaz na tuto práci

Malina, Jiří. *Nástroje pro multiplatformní programování mobilních aplikací.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

Abstrakt

Cílem této práce je usnadnit volbu nástroje pro multiplatformní vývoj mobilních aplikací pro operační systémy Android, iOS a Windows Phone. V první části práce se nachází porovnání těchto nástrojů a doporučení, pro jaké typy aplikací jsou jednotlivé nástroje užitečné. Druhá část obsahuje podrobnější porovnání nástrojů Appcelerator, Apache Cordova a Xamarin. Třetí část práce se zabývá vývojem demonstrační aplikace na vykazování práce v nástroji Xamarin.

Klíčová slova Multiplatformní, mobilní aplikace, srovnání, Android, iOS, Windows Phone, vykazování práce, MoSync, RhoMobile, Codename One, Appcelerator, Apache Cordova, Xamarin, Alpha Anywhere, Trigger.io, Neomade, PhoneGap, Sencha Touch, IBM MobileFirst, SAP Kapsel, Telerik, Appery.io.

Abstract

The goal of this work is to help determining the best tool for cross-platform mobile development for Android, iOS and Windows Phone operating systems. In the first part of this work is a comparison of the most commonly used tools. Second part is comprised of more detailed comparison of tools Appcelerator,

Apache Cordova and Xamarin. In the third part of this work is described the process of developing an application for time tracking in Xamarin.

Keywords Cross-platform, mobile applications, comparison, Android, iOS, Windows Phone, time tracking, MoSync, RhoMobile, Codename One, Appcelerator, Apache Cordova, Xamarin, Alpha Anywhere, Trigger.io, Neomade, PhoneGap, Sencha Touch, IBM MobileFirst, SAP Kapsel, Telerik, Appery.io.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Již v devadesátých letech dvacátého století se mobilní telefony staly nedílnou součástí našeho života. V počátcích byly používány především ke komunikaci, nicméně v současné době jsou tzv. chytré telefony používány při každodenních činnostech jako je nakupování, jízda hromadnou dopravou a mnohé další.

Rozmach chytrých mobilních telefonů s sebou přinesl mnoho příležitostí pro nezávislé vývojáře. Aplikace lze poměrně jednoduše distribuovat do celého světa přes služby, jako je Google Play nebo App Store. Každým dnem se na zmíněných službách objevují nové aplikace z celého světa. Mobilní aplikace pochopitelně nevyvíjejí jen nezávislí vývojáři, ale i softwarové společnosti.

Rychlost jakou se mobilní operační systémy rozvíjejí ovšem může vývoj aplikací komplikovat. Klasické přístupy vývoje softwaru, jako je například vodopádový model, nejsou pro vývoj mobilních aplikací vhodné. Při vývoji je nutné pružně reagovat na změny požadavků a změny v samotném operačním systému, pro který je aplikace vyvíjena. Z těchto důvodů jsou vhodnější tzv. agilní metodiky vývoje.

Další značnou překážkou je potřeba vývoje aplikace pro více než jeden operační systém. Nemuselo by se jednat o až tak velký problém, kdyby stačil jeden programovací jazyk pro všechny rozšířené mobilní platformy. V současné době jsou většinou aplikace pro jednotlivé operační systémy psány v odlišných programovacích jazycích. Pro operační systém Android se jedná převážně o jazyk Java . Aplikace pro systém iOS jsou psány v jazyce Objective C a v nedávné době byla přidána podpora také pro jazyk Swift. Pro Windows Phone jsou aplikace psány v jazyku C#.

Řešením zmíněného problému je vývoj aplikací pomocí tzv. multiplatformních nástrojů. Smyslem těchto nástrojů je sjednotit programovací jazyk pro více cílových platforem. S jednotným programovacím jazykem odpadá potřeba experta pro každý operační systém. Zároveň mohou být sdíleny části kódu mezi jednotlivými platformami, a tak je údržba kódu podstatně jednodušší. Tento způsob vývoje je zpravidla výrazně méně finančně i časově náročný.

Prvním cílem této práce je porovnat nejběžněji používané nástroje pro

multiplatformní vývoj mobilních aplikací na operační systémy Android, iOS a Windows Phone. Druhým cílem je navrhnout aplikaci na měření odpracovaného času. Třetím cílem je vybrat nejvhodnější nástroj pro použití v malých až středních společnostech a tento nástroj ověřit při implementaci navržené aplikace.

Kapitola 1

Srovnání multiplatformních nástrojů

V první části této kapitoly jsou popsány jednotlivé typy výstupů multiplatformních nástrojů pro vývoj mobilních aplikací spolu s jejich hlavními výhodami a nevýhodami. Ve druhé části kapitoly jsou rozebrány kategorie pro srovnávání. Třetí část obsahuje srovnání nejběžněji používaných nástrojů.

1.1 Úvod

V současnosti je na trhu velké množství nástrojů sloužících k vývoji mobilních aplikací pro vícero platforem. Jejich jednotným cílem je usnadnění vývoje a ušetření financí. Jejich přístup ke splnění tohoto cíle se však může podstatně lišit.

Každý vývojový tým má od multiplatformního nástroje odlišná očekávání. Některé nástroje se zaměřují na vývoj podnikových aplikací, jiné na vývoj běžných aplikací a existují i nástroje specializované na vývoj 2D a 3D her. Jednotlivé nástroje se mohou lišit nejen ve svém zaměření, ale i v rozsahu funkcí, programovacím jazyku, podporovaných zařízeních, ceně a mnohém dalším. Výběr vhodného nástroje je tak nesnadný úkol.

1.2 Dělení podle typu výstupu

Jednou ze základních kategorizací nástrojů je typ výstupu, který vytvářejí. Samotný typ výstupu podstatně ovlivňuje dostupné funkcionality, bezpečnost, výkon, možnosti distribuce a mnohé další. Základní kategorie jsou webová aplikace, nativní aplikace a webová aplikace běžící v nativním kontejneru tzv. hybridní aplikace. Některé nástroje dokonce umožňují vytvářet výstupy do více než jedné ze zmíněných kategorií.
Informace v této podkapitole byly čerpány z těchto zdrojů: [1], [2], [3], [4].

1.2.1 Webové aplikace

Nejdostupnější způsob jak vytvořit multiplatformní mobilní aplikaci je vytvořit mobilní či responzivní web. Vytvoření webové aplikace zpravidla bývá nejjednodušší způsob, jak podporovat více platforem. Lze využít rozšířených technologií, mezi něž patří především HTML, CSS A JavaScript. Kromě relativní jednoduchosti vývoje, mají webové aplikace další výhody. Jednou z nich je dostupnost téměř odkudkoliv. Aplikaci lze spustit z jakéhokoli zařízení, které má webový prohlížeč. Další výhodou je jednoduchost distribuce nové verze všem uživatelům. Jelikož jsou všechna data uložena centrálně, aktualizace se instantně, až na cache, projeví u všech uživatelů.

Webové aplikace bohužel také přinášejí mnoho nevýhod a omezení. Jednou z hlavních nevýhod je nedostupnost nativního API poskytovaného SDK dané platformy. Před příchodem respektive rozšířením podpory HTML5 v mobilních prohlížečích to znamenalo nedostupnost akcelerometru, gyroskopu, vestavěnému fotoaparátu, systému GPS atd. HTML5 přidává možnost přístupu k informacím z těchto zdrojů, nicméně má stále určité nevýhody. Jednou z nich je například potenciálně obtěžující potvrzovací dialog při přístupu k lokaci. U nativní aplikace uživateli stačí povolit přístup pouze jednou. Další poměrně značnou nevýhodou je omezenější podpora pro offline ukládání dat. HTML5 podporuje několik způsobů, jak ukládat data pro offline prohlížení. Nicméně tyto funkce zatím nejsou podporovány všemi rozšířenými webovými prohlížeči, a tak je nutno jejich použití důkladně zvážit. Dalším problémem s ukládáním dat v offline režimu s pomocí HTML5 je, že data nejsou nijak zabezpečena, a tak mohou být snáze zneužita.

1.2.2 Nativní aplikace

Pojmem nativní aplikace jsou většinou označovány aplikace napsané pro konkrétní cílovou platformu. Příkladem je aplikace na OS Android napsaná v jazyku Java s použitím SDK. V některých případech je tímto pojmem myšlena aplikace napsaná v čistě nativním kódu dané platformy. Příkladem čistě nativní aplikace je aplikace pro OS Android napsaná v jazyce C s použitím NDK.

Hlavní výhodou nativních aplikací je přístup ke všem funkcionalitám dané platformy. Aplikace mohou využívat například GPS včetně geo-fences, NFC, Bluetooth, Wi-Fi a další vymoženosti současných mobilních zařízení. Aplikace mají také přístup k nativním komponentám uživatelského rozhraní. Nativní aplikace jsou zpravidla také plynulejší, než aplikace běžící ve webovém prohlížeči. Aplikace lze mimo jiné distribuovat pomocí Google Play, App Store a Windows Store, kde lze například omezit podporované verze systému nebo dokonce vydat rozdílnou verzi aplikace pro konkrétní systém. Nativní aplikace jsou nainstalovány přímo na mobilním zařízení, a tak je oproti webové verzi jednodušší přidání offline podpory. Aplikace má přístup k nativním SDK,

které podporují pokročilé šifrovací mechanizmy, tím lze výrazně zkomplikovat případné zneužití citlivých dat.

Nevýhodou nativních mobilních aplikací je jejich podstatně komplikovanější a časově náročnější vývoj. Nicméně s příchodem multiplatformních nástrojů by se tato hlavní nevýhoda mohla postupem času ztratit.

1.2.3 Hybridní aplikace

Existuje způsob, jak získat hlavní výhody z obou předešlých přístupů k vývoji multiplatformních aplikací s jen několika málo nevýhodami. Hybridní aplikace jsou psané pomocí webových technologií, které běží v nativním kontejneru dané platformy. Díky tomu má aplikace přímý přístup k funkcionalitám SDK. Z hlediska distribuce k uživatelům jsou hybridní aplikace shodné s nativními aplikacemi.

Stejně jako u nativních aplikací může vývojář využít multiplatformních nástrojů, které se postarají o vytvoření nativního kontejneru pro příslušnou platformu. Výhodou těchto nástrojů je, že programátor nemusí být expert ve vývoji mobilních aplikací a může celou aplikaci napsat pomocí webových technologií, jako je například JavaScript. Nicméně značnou nevýhodou je nenativní „look&feel“, který může dokonce vést k zamítnutí aplikace při schvalovacím procesu v App Store.

Každý z výše zmíněných přístupů k vývoji mobilních aplikací má svá pro a proti. Neexistuje ideální přístup pro všechny typy aplikací.

Webové aplikace jsou nejjednodušší na vývoj a finančně nejméně náročné. Stačí vyvinout jednu verzi aplikace a ta běží na všech platformách, které podporují webový prohlížeč. Při příchodu nové platformy není nutné kód modifikovat. Lze ovšem vytvářet pouze aplikace s omezenou funkcionalitou a bez nativního vzhledu konkrétní platformy.

Hybridní aplikace jsou na vývoj také relativně jednoduché. Při příchodu nové platformy stačí upravit multiplatformní nástroj a aplikaci překompilovat. Kód samotné aplikace je nutno modifikovat jen minimálně nebo vůbec. Aplikace mají přístup k nativnímu SDK, a tak nemusí být funkcionalita aplikace nijak omezena oproti nativní aplikaci. Aplikace může napodobovat vzhled jednotlivých platforem, ale nemůže využívat nativních komponent uživatelského rozhraní. Některé multiplatformní frameworky se pokoušejí o vytvoření nástroje, který by umožnil vývoj s podobnou složitostí, jako mají webové aplikace, ale zároveň vzhledem a funkcemi nativní aplikace.

1.3 Základní porovnávané kategorie

Od nástrojů pro multiplatformní vývoj může mít každý odlišná očekávání. Nezávislý vývojář nebo malá firma mohou potřebovat pouze nástroj pro převod

webové aplikace na aplikace hybridní. Nemusí od nástroje vyžadovat telefonickou podporu nebo analýzu uživatelů. Jejich hlavním výběrovým kritériem může být cena nástroje. Naopak u velké společnosti nemusí jít tolik o pořizovací náklady, ale o vytvoření prvotřídní a snadno udržovatelné mobilní aplikace, se zabudovanými analytickými nástroji a jednoduchou integrací se současnými systémy ve společnosti.

1.3.1 Seznam porovnávaných kategorií

1. Programovací jazyk

   Programovací jazyk ovlivňuje jak složitost vývoje aplikace, tak dobu, za kterou se vývojář s nástrojem seznámí. K vývoji aplikací v některých nástrojích jsou dostačující znalosti běžného webového programátora. U jiných nástrojů je používán sice běžně rozšířený jazyk, jako například C#, ale je potřeba se naučit pracovat s daným frameworkem. Na trhu jsou i nástroje používající k vývoji vlastní programovací jazyk.

2. Výstup

   Pravděpodobně nejdůležitějším výběrovým kritériem je typ výstupu. V této práci dělíme výstupy nástrojů do tří kategorií – webová, hybridní a nativní aplikace. Detailnější popis kategorií společně s jejich výhodami a nevýhodami jsou popsány v kapitole 1.2 „Dělení podle typu výstupu“.

3. Podporované platformy

   Mezi nejrozšířeněji podporované mobilní platformy patří iOS a Android. Nicméně většina nástrojů už podporuje i vývoj aplikací pro Windows Phone. Počet podporovaných platforem značně ovlivňuje velikost trhu, pro který bude aplikace dostupná. I když celosvětově platformy Android a iOS mají podle IDC společně přes 95% podílu na trhu, mohou být regiony nebo společnosti, kde je podíl Windows Phone větší.[5]

4. Licence

   Jednotlivé nástroje se liší ve způsobu poskytování licence. Některé nástroje jsou zdarma. U jiných je licence poskytována na stanovený časový úsek, počet vývojářů nebo počet aplikací. Díky odlišnostem ve způsobu licencování produktů nebude tato kategorie sloužit k porovnávání, ale bude mít pouze informativní charakter.

5. Podporovaná zařízení

   Většina nástrojů podporuje vývoj aplikací jak pro mobilní telefony, tak tablety. Nicméně se zvyšující se popularitou dalších mobilních zařízení, jako jsou například chytré hodinky nebo brýle, může být toto kritérium užitečné.

6. Dokumentace

   Kvalita dokumentace může velmi ovlivnit jak rychlost seznámení se s nástrojem, tak rychlost samotného vývoje. Do této kategorie bude zahrnut také rozsah a kvalita ukázek zdrojových kódů, které jsou nedílnou součástí kvalitní dokumentace.

7. Podpora nativního UI

   Rozšířené mobilní platformy poskytují doporučení, jak by běžná aplikace měla vypadat a jak by se měla chovat. Chování a vzhled napříč operačními systémy se značně liší. Pro uživatele jednoho systému může být velmi matoucí, když je aplikace navrhnuta pro jiný operační systém. V této kategorii bude brána v potaz podpora pro odlišná UI na jednotlivých platformách a možnost využití nativních komponent uživatelského rozhraní.

8. Podpora volání nativního kódu

   Abychom mohli využít pokročilých funkcionalit dané platformy, je potřeba mít možnost volat nativní kód. Některé nástroje volání nativního kódu nepodporují vůbec. Jiné podporují nepřímé volání nativního kódu. Pro nepřímé volání nástroje využívají frameworku, který slouží jako most mezi nativním kódem a kódem, ve kterém je aplikace psána. Programátor má zpravidla možnost psát vlastní rozšíření pro daný framework, nicméně k implementaci takového rozšíření potřebuje pokročilé znalosti. Poslední kategorií jsou nástroje, které umožňují vývojáři volat části kódu napsané přímo v nativním jazyku dané platformy.

9. Aktivita komunity

   Silná komunita také může velmi usnadnit vývoj aplikací. Jelikož se tato práce nezaměřuje na produkty pro velké společnosti, mnoho nástrojů neposkytuje podporu nebo je podpora zpoplatněna. Jednou z hlavních metod porovnání jednotlivých nástrojů z hlediska aktivity komunity je počet a pravidelnost příspěvků na portálech jako je Google Groups a Stack Overflow.

1.4 Srovnání

V této sekci jsou srovnány nejrozšířenější multiplatformní nástroje používané k vývoji mobilních aplikací. Na trhu se nacházejí platformy, které poskytují převážně konzultantskou činnost a šablony do nástrojů jako je Xamarin, Apache Cordova atp. Tyto platformy, mezi něž patří např. iFactr nebo FeedHenry, nejsou ve srovnání uvedeny. Převážně z toho důvodu, že jsou určeny především pro větší společnosti. Dalším typem nástrojů, které nebyly do srovnání zahrnuty, přestože mají rozsáhlou komunitu, jsou nástroje specializované

na vývoj her. Mezi tyto nástroje patří například Adobe Air, Marmelade a Unity.

1.4.1 MoSync

Informace o nástroji MoSync byly čerpány z oficiálních webových stránek. [6]

1. Programovací jazyk

   MoSync je jedním z mála nástrojů, jehož hlavním programovacím jazykem je C/C++. Umožňuje ovšem vyvíjet aplikace i v HTML a JavaScriptu nebo dokonce v kombinaci všech třech zmíněných jazyků.

2. Výstup

   V případě aplikace naprogramované v jazyce JavaScript je výsledný výstup typická hybridní aplikace.

   V případě vývoje v jazyce C/C++ je výstupem nativní aplikace běžící ve vlastním běhovém prostředí.

   MoSync kompilátor přeloží C/C++ zdrojový do tzv. MoSync IL. Tento kód je poté zpracován tzv. Pipe-Tool, který se postará o optimalizaci a následné převedení do několika formátů. Pipe-Tool má také na starost zabalení obrázků, mediálních souborů atp. Výstup z Pipe-Tool je poté zpracován tzv. Packagerem, který má na starost buď vytvoření spustitelného souboru nebo vytvoření projektu pro vývojová prostředí Xcode nebo Visual Studio. Packager přidává ke kódu běhové prostředí, které se stará o provádění kódu na cílové platformě.

3. Podporované platformy

   MoSync podporuje iOS pouze do verze 6.x, Windows Phone do verze 7.x a Android do verze 4.x.

4. Licence

   Produkty vytvořené pomocí nástroje MoSync mohou být šířeny dvěma způsoby. První způsobem je produkt vydat pod open-source licencí GPL1v2 a přiložit oznámení o autorských právech a licencích. Druhým způsobem je získat komerční licenci. MoSync nabízí komerční licenci zdarma bez omezení na funkcionalitě. Pro získání komerční licence je potřeba pravidelná roční registrace. S placenými komerčními licencemi je navíc poskytována podpora.

5. Podporovaná zařízení

   Nástroj slouží pro vytváření aplikací pro mobilní zařízení.

6. Dokumentace

   MySync poskytuje desítky tutoriálů a ukázkových aplikací jak pro vývoj v C/C++, tak pro vývoj v JavaScriptu. Součástí je také vygenerovaná API dokumentace.

7. Podpora nativního UI

   Při vývoji aplikace pouze s použitím jazyka C/C++ MoSync poskytuje tzv. Widget API, které umožňuje vytvářet nativní komponenty uživatelského rozhraní na všechny tři cílové platformy se 100 % společného kódu. V případě vývoje aplikací v jazyce JavaScript MoSync neposkytuje nativní vzhled komponent uživatelského rozhraní. Přestože v dokumentaci je sekce o nativním uživatelském rozhraní i při vývoji v tomto jazyce, ve skutečnosti se nejedná o opravdu nativní rozhraní, ale pouze o jeho napodobování pomocí webových technologií.

8. Podpora volání nativního kódu

   MoSync poskytuje balíček knihoven, které umožňují přistupovat například k senzorům, jako je GPS a fotoaparát, ale i k API pro notifikace, práci se zvukem, in-app objednávkami, widgety atp. Jelikož je platforma open-source, pokročilý vývojář může vytvářet nové nebo upravovat stávající knihovny.

9. Aktivita komunity

   V roce 2013 byla vydána verze 4.0 Alpha, která zatím stále nebyla dokončena. Na serveru Stack Overflow je pouze pár desítek příspěvků. Na fóru na oficiálních stránkách je na první pohled aktivita poměrně vysoká, nicméně se z naprosté většiny jedná pouze o spam.

1.4.2 RhoMobile Suite

Informace o nástroji RhoMobile Suite byly čerpány z oficiálních webových stránek. [7]

Tato platforma se skládá z několika komponent, nicméně v licenci zdarma jsou dostupné pouze dvě. Open-source nástroj na vytváření hybridních mobilních aplikací Rhodes a plugin RhoStudio pro vývojové prostředí Eclipse. V placené verzi je dostupná celá řada dalších komponent. Mezi ně patří například knihovna RhoElements. Tato knihovna přidává API převážně pro Motorola Solutions zařízení s Windows Mobile. Dále přidává například podporu pro čtení čárových kódů pro iOS a Android. Další zajímavou komponentou je Cloud Build, která umožňuje sestavování aplikací v cloudu, a tak není nutné sestavovat iOS a Windows Phone aplikace v oficiálních vývojových prostředích.

1. Programovací jazyk

   Vývojář má možnost volby mezi JavaScriptem a kombinací JavaScriptu s jazykem Ruby. Pro úplný přístup ke všem funkcionalitám je ovšem Ruby zapotřebí. V čistém JavaScriptu je například omezen přístup k API pro budík, GPS, gesta a mnohé další. Pro JavaScript lze využít libovolné knihovny třetích stran, nicméně RhoStudio generuje kód v JQuery Mobile.

2. Výstup

   RhoMobile je typický zástupce nástroje vytvářejícího hybridní aplikace.

3. Podporované platformy

   Mezi podporované mobilní platformy patří Android, Windows Phone, Windows Mobile a iOS. Řada API není dostupná při vývoji v jazyce JavaScript, nicméně pro Windows Phone není řada API dostupná ani v jazyce Ruby. Chybí například podpora pro čtení čárových kódů, fotoaparát, GPS a notifikace.

4. Licence

   RhoMobile Suite nabízí dva typy licencí. Prvním typem je licence zdarma, ovšem dostupné komponenty jsou omezeny na RhoStudio a Rhodes. Cena druhého typu licence začíná na 299 dolarech za měsíc. Kompletní seznam podporovaných API, včetně indikace potřebné úrovně licence viz http://docs.rhomobile.com/en/5.0.25/guide/apisummary.

5. Podporovaná zařízení

   Mezi podporovaná zařízení patří mobilní telefony, nicméně i mnoho Motorola Solutions zařízení. Motorola Solutions zařízení jsou používána především v průmyslu. Jedná se například o zařízení WT41N0, které slouží jako mini počítač na zápěstí pro pracovníky ve skladu.

6. Dokumentace

   Součástí dokumentace jsou základní postupy, jak s nástrojem pracovat. Dále je v dokumentaci popis jednotlivých API s jednoduchými příklady. Nicméně ukázky kódu s řešením komplexnějších problémů jsou zastaralé nebo chybějí úplně.

7. Podpora nativního UI

   Součástí frameworku je balíček CSS souborů, pomocí kterých lze napodobit nativní vzhled jednotlivých platforem.

8. Podpora volání nativního kódu

   Vývojář může využít knihoven, které jsou součástí RhoMobile nebo využít knihoven třetích stran. Poměrně jednoduše lze vytvořit vlastní knihovnu v nativním SDK dané platformy.

9. Aktivita komunity

   Na Stack Overflow je celkem necelých 400 příspěvků z toho 50 příspěvků v posledních 30 dnech. Na oficiálním fóru pravidelně diskutuje několik desítek aktivních vývojářů. Aktivita na Google Groups je v posledním půl roce téměř nulová.

1.4.3 Codename One

Informace o nástroji Codename One byly čerpány z oficiálních webových stránek. [8]

1. Programovací jazyk

   Vše, včetně uživatelského rozhraní, je psáno v jazyce Java.

2. Výstup

   Výstupem nástroje jsou nativní aplikace. Sestavování aplikací probíhá na Codename One serveru. Lokální sestavování aplikací není oficiálně podporováno, nicméně existují komunitní tutoriály s postupem, jak aplikace sestavovat na vlastním zařízení. U firemní licence je poskytována podpora pro vytvoření serverů pro sestavování na vlastním hardwaru.

3. Podporované platformy

   Hlavní podporované platformy jsou iOS, Android a BlackBerry. Podpora pro Windows Phone byla přidána v roce 2014 a podle oficiálních příspěvků na stránkách není ještě kompletně odladěna.

4. Licence

   Nástroj je open-source a pro vývoj aplikací jak pro soukromé, tak pro komerční užití si není potřeba pořizovat placený účet. Nicméně s placeným účtem má uživatel přístup k více video tutoriálům, k pokročilým službám a e-mailové či telefonní podpoře. Mezi tyto služby patří mimo jiné push notifikace a reportování pádů aplikace.

5. Podporovaná zařízení

   Codename One je zaměřen na vývoj aplikací pro mobilní telefony.

6. Dokumentace

   Součástí dokumentace je několik málo video tutoriálů, JavaDocs a komunitní tutoriál obsahující krátké ukázky jednotlivých funkcí. Poslední aktualizace dokumentace byla provedena v prosinci 2013.

7. Podpora nativního UI

   U základních komponent uživatelského rozhraní je podporován nativní vzhled a chování. Uživatel má možnost vytvářet uživatelské rozhraní buď

přímo v kódu nebo může využít WYSIWYG editoru. Framework poskytuje přímou podporu i pokročilejších UI komponent, jako je například, v terminologii platformy Android, tzv. drawer.

8. Podpora volání nativního kódu

   Pro vytvoření nativního kódu pro příslušnou platformu stačí vytvořit interface, který dědí od NativeInterface. Vytvořením interface je automaticky vygenerován kód v jazycích daných platforem. Nicméně parametry a návratové typy metod jsou omezeny na primitivní typy, pole primitivních typů a Pair komponentu.

9. Aktivita komunity

   Komunita je relativně aktivní. Hlavním komunikačním kanálem je jednoznačně skupina Codename One na portále Google Groups, kde se objevuje 5 - 10 dotazů denně. Na portále StackOverFlow je aktivita výrazně nižší, nicméně také se zde pravidelně objevují nové dotazy.

1.4.4 Appcelerator

Informace o nástroji Appcelerator byly čerpány z oficiálních webových stránek. [9]

1. Programovací jazyk

   Hlavním programovacím jazykem pro platformu Appcelerator je JavaScript. Pro využití nativních funkcionalit dané platformy je nutné se seznámit s tzv. Titanium SDK. Součástí platformy je Alloy framework. Jedná se o JavaScriptový framework založený na MVC architektuře s vestavěnou podporou pro JavaScriptové knihovny Bacbbone.js a Underscore.js. Nástroj není přizpůsoben pro jiné JavaScriptové knihovny.

2. Výstup

   Vývojář má možnost zvolit si mezi vytvořením hybridní aplikace běžící v nativním webovém kontejneru a vytvořením nativní aplikace.

3. Podporované platformy

   V současné chvíli jsou plně podporovány mobilní platformy iOS a Android. Podpora pro Windows Phone je zatím pouze v beta verzi, nicméně v blízké budoucnosti je plánováno vydání release verze.

4. Licence

   Titanium Studio i Titanium SDK využívají celou řadu open-source nástrojů. Jejich kompletní výčet s příslušnými open-source licencemi lze nalézt na adrese http://www.appcelerator.com/legal/open-source-attribution/. Součástí platformy jsou také služby v cloudu, mezi něž patří například

analytické nástroje a podpora vytvoření vlastního API v jazyce node.js. V licenci zdarma je omezen denní počet dotazů na API a také jsou omezeny funkce vývojového prostředí.

5. Podporovaná zařízení

   Platforma je zaměřena na vývoj pro mobilní telefony a tablety.

6. Dokumentace

   Součástí platformy je rozsáhlá, ale zároveň velmi přehledná dokumentace. Obsahuje několik kompletních ukázkových aplikací a databázi nahrávek videí z konferencí.

7. Podpora nativního UI

   Appcelerator podporuje velkou část nativních komponent uživatelského rozhraní.

8. Podpora volání nativního kódu

   Titanium SDK obsahuje API pro volání základních nativních funkcí daných platforem z jazyka JavaScript. Nicméně v případě potřeby má vývojář možnost stáhnout nebo vytvořit tzv. modul pro volání jím požadovaného nativního kódu. Mezi dostupné moduly ke stažení patří například modul pro podporu čtení čárových kódů nebo modul pro tzv. in-app platby.

9. Aktivita komunity

   Podle oficiálních webových stránek má Appcelerator více něž 650 tisíc registrovaných vývojářů. Hlavním komunikačním kanálem je sekce Q&A na oficiálních stránkách, kde se denně objevuje 5 - 10 dotazů. Na portále StackOverFlow jsou také pravidelně pokládány nové dotazy.

1.4.5 Apache Cordova

Informace o nástroji Apache Cordova byly čerpány z oficiálních webových stránek.[10]

Apache Cordova je open-source nástroj ve vlastnictví společnosti Apache. Jedná se o most mezi jazykem JavaScript a nativním SDK jednotlivých mobilních platforem. Nástroj je interně využíván několika frameworky, mezi něž patří například PhoneGap, Sencha Touch, IBM MobileFirst, Telerik, Appery.io nebo SAP Kapsel.

1. Programovací jazyk

   Pro implementaci aplikací v nástroji Apache Cordova je potřeba znalostí běžného webového vývojáře. Kód je psán v jazycích JavaScript, CSS a

HTML. Vývojář má možnost použít libovolnou JavaScriptovou knihovnu třetí strany.

2. Výstup

   Výstupem je typická hybridní aplikace.

3. Podporované platformy

   Apache Cordova podporuje všechny rozšířené mobilní platformy mezi něž patří Android, iOS, Windows Phone a BlackBerry.

4. Licence

   Samotný framework Apache Cordova je open-source pod Apache 2.0 licencí. Nicméně nadstavby jako PhoneGap, IBM Worklight atp. mají vlastní licenční podmínky.

5. Podporovaná zařízení

   Podporována jsou všechna mobilní zařízení s webovým prohlížečem podporujícím JavaScript tzn. převážně telefony a tablety.

6. Dokumentace

   Samotný Apache Cordova je pouze kontejner, který umožňuje přidávání modulů. Společnost Apache poskytuje sadu modulů pro napojení na základní funkce nativních SDK. Dokumentace kontejneru a sady základních modulů je dostačující. Nicméně se objevují moduly třetích stran, které dokumentaci úplně postrádají.

7. Podpora nativního UI

   Podobně jako u frameworku RhoMobile lze pouze napodobovat nativní vzhled komponent uživatelského rozhraní pomocí CSS stylů. Existují nadstavby, mezi něž patří například Sencha Touch, které obsahují CSS styly napodobující nativní vzhled.

8. Podpora volání nativního kódu

   Na oficiálních stránkách je komunitní repositář s více než 700 moduly pro volání nativního kódu. Relativně snadno lze vytvořit vlastní modul.

9. Aktivita komunity

   Komunita je velmi rozsáhlá a aktivní. Na portále StackOverFlow se týdně objevuje stovky nových dotazů. Mnoho odpovědí na otázky ohledně Apache Cordova lze nalézt pod tagem „PhoneGap“.

1.4.6 Xamarin

Informace o nástroji Xamarin byly čerpány z oficiálních webových stránek. [11]

1. Programovací jazyk

   Kód v nástroji Xamarin lze kompletně psát v jazyce C#. Nicméně pro tvorbu uživatelského rozhraní je preferovanějším způsobem psaní kódu v jazyku XAML.

2. Výstup

   Výstupem je plně nativní aplikace. Na platformách Android a iOS je k aplikačním balíčkům připojen .NET framework Mono. Pro Windows Phone lze aplikace sestavovat přímo ve Visual Studiu bez zvláštního sestavovacího nástroje poskytovaného společností Xamarin, jelikož jsou aplikace psány v jazyce C#.

3. Podporované platformy

   Nástroj slouží k vývoji aplikací pro operační systémy Android, iOS, Windows Phone a Mac OS.

4. Licence

   V licenci zdarma je omezena velikost výsledné aplikace a dostupnost knihoven třetích stran. Xamarin nabízí několik typů placených licencí, které se liší dostupnou funkcionalitou. Cena pro nezávislé vývojáře je 25 dolarů měsíčně. Pro společnosti začíná cena na 83 dolarech za měsíc. Kompletní ceník a seznam dostupných funkcionalit lze nalézt na adrese https://store.xamarin.com/.

5. Podporovaná zařízení

   Do verze Xamarin 5.0 byly podporovány pouze mobilní telefony a tablety, nicméně v současné době lze vytvářet aplikace i pro wearables.

6. Dokumentace

   Xamarin poskytuje velmi rozsáhlou a kvalitní dokumentaci. Pro začínající vývojáře jsou dostupné 4 relativně rozsáhlé vzorové aplikace spolu s desítkami ukázkových projektů.

7. Podpora nativního UI

   Nativní uživatelské rozhraní lze vytvářet dvěma způsoby. Prvním způsobem je vytvoření UI pomocí nativního SDK. Tento způsob má tu nevýhodu, že se vývojář musí seznámit s vývojem UI v nativních SDK všech platforem, pro které chce aplikaci vytvářet. Další nevýhodou je větší množství kódu specifického pro konkrétní platformu. Druhým relativně novým způsobem je vytvoření jednoho UI pomocí tzv. Xamarin Forms.

8. Podpora volání nativního kódu

   Xamarin SDK je namapován na téměř všechna API nativních SDK. Pokud vývojář přesto potřebuje volat nativní kód, například protože má již implementovanou knihovnu, Xamarin umožňuje napojení knihovny psané v nativním kódu na C# projekt.

9. Aktivita komunity

   Xamarin má přes 900 000 registrovaných vývojářů a jejich počet každý měsíc roste. Hlavní komunikačním kanálem je fórum na oficiálních stránkách, nicméně i na portále StackOverFlow je pravidelně posíláno více jak 10 dotazů denně.

1.4.7 Alpha Anywhere

Informace o nástroji Alpha Anywhere byly čerpány z oficiálních webových stránek. [12]

1. Programovací jazyk

   Jedná se o nástroj, který se snaží minimalizovat nutnost psaní kódu. Většina kódu je generována, nicméně vývojář má možnost kód modifikovat. Webové aplikace jsou vyvíjeny v jazycích XBasic, HTML, CSS a JavaScript.

2. Výstup

   Nástroj neslouží jen k implementaci klientské aplikace, ale i k implementaci serverového rozhraní a databáze. Výstupem klientské části je běžná webová aplikace. Vývojář má možnost propojit Alpha Anywhere s frameworkem PhoneGap a vytvářet aplikace hybridní.

3. Podporované platformy

   Podporovány jsou všechny platformy s webovým prohlížečem s podporou jazyka JavaScript. V případě integrace s nástrojem PhoneGap jsou podporovány stejné platformy, jako u frameworku Apache Cordova.

4. Licence

   Alpha Anywhere nabízí zkušební verzi na 30 dnů zdarma. Cena plné verze nástroje začíná na 999 amerických dolarech za rok.

5. Podporovaná zařízení

   Výstupem je webová aplikace s responzivním designem cílená především na mobilní telefony, tablety a PC.

6. Dokumentace

   Kvalita dokumentace je na velmi nízké úrovni. Velká část byla naposledy upravována před více jak třemi roky. Chybí přehledné oddělení návodů pro jednotlivé verze nástroje. Na oficiálních webových stránkách je dostupná databáze video tutoriálů. Nicméně bez kvalitní psané dokumentace je jejich užitek nízký.

7. Podpora nativního UI

   Stejně jako u jiných čistě webových nebo hybridních aplikací není nativní uživatelské rozhraní podporováno.

8. Podpora volání nativního kódu

   Volání nativního kódu je podporováno pouze v případě integrace s nástrojem PhoneGap.

9. Aktivita komunity

   Jediným aktivním komunikačním kanálem je message board na oficiálních webových stránkách. Objevuje se zde několik příspěvků týdně. Na portálech StackOverFlow a Google Groups je aktivita nulová.

1.4.8 Trigger.io

Informace o nástroji Trigger.io byly čerpány z oficiálních webových stránek. [13]

1. Programovací jazyk

   Trigger.io je dalším zástupcem nástrojů, u kterých jsou používány převážně webové technologie JavaScript, HTML a CSS. Součástí nástroje je tzv. Forge framework, který zpřístupňuje nativní funkcionality z JavaScriptového kódu.

2. Výstup

   Výstupem je webová nebo hybridní aplikace. Nástroj umožňuje vývojáři vidět změny v kódu na emulátoru nebo připojeném zařízení v reálném čase. Dalším zajímavostí je tzv. Trigger.io Reaload, pomocí kterého je možné aktualizovat aplikaci v klientských zařízeních bez nutnosti nahrávání nové verze do Google Play nebo App Store. V případě vývoje hybridních aplikací probíhá sestavování v cloudu podobně jako u nástroje RhoMobile. Na serveru je sestavován pouze hybridní kontejner a samotný zdrojový kód tak není nikdy odesílán z lokálního zařízení.

3. Podporované platformy

   V případě vývoje hybridní aplikace jsou podporovány pouze platformy Android a iOS. Nicméně pro ostatní platformy lze použít webovou verzi.

4. Licence

   Cena nástroje se odvíjí od počtu vývojářů a počtu vyvíjených aplikací. Cena pro jednoho vývojáře začíná na 39 amerických dolarech měsíčně při ročním úvazku. Dále je také nabízena 14-ti denní zkušební verze zdarma.

5. Podporovaná zařízení

   Nástroj je zaměřen především na vývoj aplikací pro mobilní telefony. Nicméně stejně jako u jiných nástrojů pro vývoj hybridních aplikací lze minimálně část kódů použít i pro vývoj webových stránek.

6. Dokumentace

   Psaná dokumentace je stručná, ale velmi výstižná. Veškeré API je popsáno a v případě pokročilých funkcionalit opatřeno krátkou ukázkou zdrojového kódu. Součástí dokumentace je několik ukázkových aplikací.

7. Podpora nativního UI

   Trigger.io poskytuje několik málo předpřipravených CSS stylů pro nativní vzhled. Nicméně opravdu nativní uživatelské rozhraní není podporováno.

8. Podpora volání nativního kódu

   Vývojář má možnost vytvořit si vlastní modul v nativním kódu příslušné platformy, případně využít již hotové moduly. Oficiální databáze obsahuje pouze několik položek, které zpřístupňují jen základní nativní funkce.

9. Aktivita komunity

   Na portále StackOverFlow bylo položeno přibližně sto dotazů za posledních 365 dnů. Na serveru Google Groups se v posledním roce neobjevil žádný dotaz týkající se nástroje Trigger.io.

1.4.9 Neomade

Informace o nástroji Neomade byly čerpány z oficiálních webových stránek. [14]

1. Programovací jazyk

   Aplikace jsou programovány v jazyce Java. Uživatelské rozhraní lze vytvářet přímo pomocí Java kódu nebo v přehlednější formě v XML dokumentu. Struktura kódu se velmi podobá struktuře kódu na platformě Android.

2. Výstup

   Stejně jako v případě Codename One je nástroj určen k vývoji nativních aplikací.

3. Podporované platformy

   Neomade slouží k vytváření aplikací pro mobilní operační systémy Android, iOS, Windows Phone a Blackberry.

4. Licence

   Ve verzi zdarma lze vyvíjet aplikace pouze pro operační systémy Android a iOS. Plná licence pro firmu o maximálně 10 zaměstnancích stojí 999 EUR na jeden rok pro jednoho vývojáře. Pro větší společnosti je nabízena licence za 4999 EUR za vývojáře.

5. Podporovaná zařízení

   Nástroj se specializuje na vývoj aplikací pro mobilní telefony a tablety.

6. Dokumentace

   Na oficiálních webových stránkách se nachází přehledná a pravidelně aktualizovaná dokumentace funkcionalit, které nástroj nabízí. Nicméně obsahuje pouze několik ukázkových úryvků kódu a neobsahuje žádné kompletní ukázkové projekty. Součástí dokumentace je i vygenerovaný JavaDoc pro Generic API.

7. Podpora nativního UI

   Nástroj poskytuje abstraktní vrstvu, pomocí které má vývojář možnost vytvářet nativní uživatelské rozhraní pro všechny cílové platformy obdobně, jako je tomu u Xamarin.Forms. Podporovány jsou základní komponenty uživatelského rozhraní, jako např. Button, TextField a Layouts.

8. Podpora volání nativního kódu

   Součástí tzv. Generic API je rozhraní pro práci s lokálním úložištěm, fotoaparátem, SMS, senzory, ale i například REST API. Vývojář má možnost implementovat požadovanou funkcionalitu v nativním kódu příslušné platformy.

9. Aktivita komunity

   Na portálech StackOverflow a Google Groups je aktivita komunity nulová. Na oficiálních stránkách se nenachází fórum pro vývojáře.

1.4.10 Ostatní

V této sekci jsou popsány platformy, které využívají jednoho ze zmíněných nástrojů, ale mají přidanou hodnotu, a tak stojí za zmínku.

1.4.10.1 IBM MobileFirst

IBM MobileFirst, dříve označován jako IBM Worklight, je platforma zaměřena na usnadnění vývoje aplikací pro podniky. Při vývoji hybridních aplikací je využíván nástroj Apache Cordova. Platformu lze využít i při vývoji plně nativních aplikací. V říjnu 2014 byla vydána první verze komponenty pro platformu Xamarin.[15]

Platforma poskytuje perzistentní úložiště, které je pomocí jazyka JavaScript šifrováno s použitím šifry AES256. Další zajímavou funkcí je vzdálené vynucení instalace kritických aktualizací. Nástroj přidává také například podporu pro SSO, Push notifikace, automatickou detekci jailbreaku a mnohé další. [1][16]

1.4.10.2 SAP Kapsel

SAP Kapsel je balíček modulů pro Apache Cordovu, který přidává podobné funkcionality jako IBM MobileFirst. Jedná se například o podporu pro bezpečné přihlašování uživatelů, čtení čárových kódu, šifrované úložiště a mnohé další. [17]

1.4.10.3 Telerik AppBuilder a Appery.io

Telerik AppBuilder (Icenium) i Appery.io jsou balíčky nástrojů a služeb, které jsou zaměřeny na usnadnění a zrychlení vývoje multiplatformních aplikací s použitím nástroje Apache Cordova. Poskytují WYSIWYG editory pro tvorbu uživatelského rozhraní, nástroje pro napojení aplikace na REST API a databázi, sestavování aplikací v cloudu a mnohé další.[18]

1.4.10.4 Sencha Touch

Sencha Touch je HTML a JavaScriptový framework uzpůsobený pro vývoj hybridních aplikací, které se svým vzhledem blíží vzhledu aplikacích vyvinutých v nativním prostředí. Obsahuje balíček kaskádových stylů pro běžně používané komponenty uživatelského rozhraní přizpůsobených příslušným platformám. Pro sestavování aplikací je podporována integrace s nástroji Apache Cordova a PhoneGap.[19]

1.4.10.5 PhoneGap

Název PhoneGap je původním názvem nástroje Apache Cordova. Nicméně po darování open-source nástroje společností Adobe/Nitobi společnosti Apache byl projekt přejmenován na Apache Cordova. PhoneGap je tak často zaměňován za samotný sestavovací nástroj Apache Cordova. Jedná se však pouze o jednu z jeho distribucí.[20]

PhoneGap přidává několik rozšíření jako například službu PhoneGap Build. Ta podobně jako Cloud Build u nástroje RhoMobile slouží k sestavování aplikací na serveru. [21]

1.5 Shrnutí

Každý z porovnávaných nástrojů má určité výhody i nevýhody. Vhodný nástroj je třeba vybrat na základě požadavků na cílovou aplikaci. V tabulkách 1.1 a 1.2 jsou uvedeny nejdůležitější kategorie pro výběr. U každé kategorie je uvedeno „A“ pokud nástroj v dané kategorii vyhovuje, „N“ pokud nevyhovuje a „A-“ pokud vyhovuje s výhradami.

	MoSync	RhoMobile	Codename One	Appcelerator
Hl. jazyk	C/C++	JS/Ruby	Java	JS
Nativní apl.	A	N	A	A
Hybridní apl.	A	A	N	A
Kvalita dokumentace	A	N	N	A
Nativní UI	A	N	A	A
Aktivita komunity	N	A	A	A
Android	A(4.x)	A	A	A
iOS	A(6.x)	A	A	A
Windows Phone	A(7.x)	A-	A(beta)	A(beta)

Tabulka 1.1: Srovnání nástrojů 1

	Apache Cordova	Xamarin	Alpha Anywhere	Trigger.io	Neomade
Hl. jazyk	JS	C#	XBasic/JS	JS	Java
Nativní apl.	N	A	N	N	A
Hybridní apl.	A	N	A*	A	N
Kvalita dokumentace	A	A	N	A	N
Nativní UI	N	A	N	N	A
Aktivita komunity	A	A	A-	N	N
Android	A	A	A*	A	A
iOS	A	A	A*	A	A
Windows Phone	A	A	A*	N	A

Tabulka 1.2: Srovnání nástrojů 2

- Nástroj MoSync má dva zásadní nedostatky. Prvním nedostatkem je velmi nízká aktivita komunity a v posledním roce i nízká aktivita samotných vývojářů nástroje. Druhým nedostatkem je chybějící podpora pro nové verze cílových platforem.
- Nástroj RhoMobile je relativně kvalitní nástroj v případě, že cílové platformy jsou iOS a Android. Podpora Windows Phone je značně omezená a nic nenapovídá tomu, že by společnost Motorola v budoucnu plánovala podporu rozšířit.

- Velkou výhodou nástroje Codename One je využití jednoho z nejrozšířenějších programovacích jazyků. Nicméně pouze komunitní a navíc neaktuální dokumentace, spolu s omezenou podporou vývoje pro Windows Phone, činí nástroj takřka nepoužitelným.

- Appcelerator se jednoznačně řadí mezi nejlepší nástroje současnosti. Jeho jedinou zásadní nevýhodou byla chybějící podpora vývoje pro Windows Phone, nicméně po vydání beta verze je jen otázka času, kdy bude vývoj plně podporován.

- Apache Cordova je velmi vhodný nástroj, pokud není kritický nativní vzhled a chování aplikace. Velkou výhodou při použití tohoto nástroje je velikost jeho komunity.

- Xamarin je dalším nástrojem bez jakýchkoli větších nedostatků. Jeho velkou výhodou, obdobně jako u Codename One, je využití rozšířeného programovacího jazyka C# pro tvorbu nativních aplikací.

- Alpha Anywhere je nástroj, který se snaží minimalizovat nutnost psaní kódu. V tabulce 1.2 jsou uvedeny „*“ indikující podporu vývoje hybridních aplikací v případě integrace s nástrojem PhoneGap. Nástroj má dva klíčové nedostatky. Prvním z nich je nepřehlednost, malý rozsah a neaktuálnost dokumentace. Druhým je složitost implementace funkcionalit, které nejsou přímo dostupné. Jednou z příčin této složitosti je jazyk XBasic, který nepatří mezi běžně používané programovací jazyky.

- Trigger.io je nástroj podobající se nástroji Apache Cordova. Jelikož jsou aplikace sestavovány na serveru, není nutné mít pro vývoj na iOS operační systém Mac OS. Nicméně je oproti nástroji Apache Cordova podstatně méně rozšířen a má několik nedostatků. Hlavní z nich jsou chybějící podpora pro platformu Windows Phone a relativně nízká aktivita komunity.

- Neomade je nástroj, který se v mnoha ohledech podobá nástroji Codename One. Jeho hlavní výhodou je aktuální přehledná dokumentace. Bohužel aktivita anglicky mluvící komunity je téměř nulová, a tak je použití nástroje pro komerční vývoj značně riskantní.

Ve srovnání jsou tři nástroje, které rozsahem funkcí a velikostí komunity jednoznačně vyčnívají. Jedná se o nástroje Apache Cordova, Appcelerator a Xamarin. V následující kapitole se nachází jejich podrobnější porovnání.

Kapitola 2

Srovnání nástrojů Apache Cordova, Appcelerator a Xamarin

V této kapitole jsou podrobněji popsány architektury, možnosti persistentního ukládání dat a možnosti sdílení kódu třech nástrojů vybraných v předchozí kapitole. V závěru kapitoly se nachází doporučení, pro který případ užití jsou nástroje vhodné.

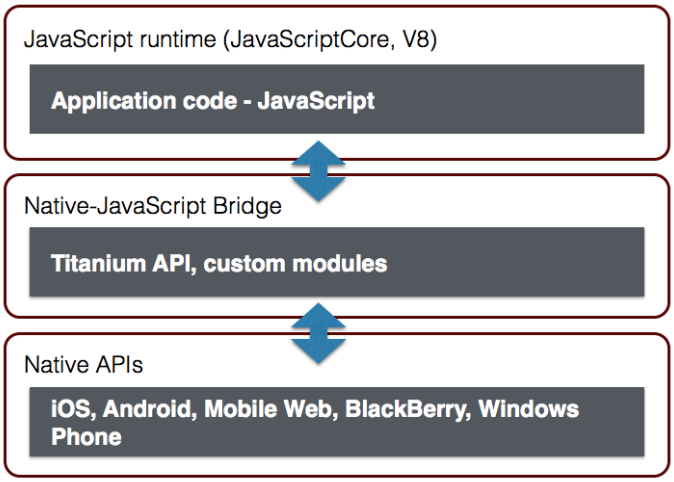
2.1 Úvod

Ze srovnání v předchozí kapitole vyplynulo, že nejvhodnější nástroje pro multiplatformní vývoj mobilních aplikací jsou Apache Cordova, Appcelerator a Xamarin. Pro výběr nejvhodnější nástroje je třeba provést podrobnější porovnání z hlediska chování na cílových platformách, možností perzistentního ukládání dat, výkonu a nutnosti psaní specifického kódu pro jednotlivé platformy.

2.2 Appcelerator

Společnost Appcelerator nabízí produkty jak pro nezávislé vývojáře či společnosti, tak pro velké podniky. Hlavními produkty cílenými na nezávislé vývojáře a společnosti jsou Titanium Studio a Titanium SDK. Veškeré služby a produkty pro podniky se skrývají pod názvem Appcelerator Platform.

Jak již bylo zmíněno v úvodní kapitole, podpora pro operační systém Windows Phone je zatím pouze v Beta verzi. Dokumentace zatím není kompletně aktualizována, proto nelze z jistotou tvrdit, že všechny funkce zmíněné v této kapitole budou na tomto operačním systému dostupné.



Obrázek 2.1: Titanium architektura
[23]

Informace v této podkapitole byly čerpány z oficiální dokumentace nástroje. [22]

2.2.1 Appcelerator Titanium

Jedná se o platformu sloužící k vývoji multiplatformních nativních aplikací v jazyku JavaScript. Tato platforma je z velké části zdarma a většina nástrojů je vydána pod open-source licencí.

2.2.2 Architektura

Na obrázku 2.1 je znázorněn zjednodušený pohled na architekturu Appcelerator Titanium. Při spuštění aplikace se vytvoří běhové prostředí, znázorněné v horní části obrázku, které se stará o vyhodnocení JavaScriptového kódu. Tento kód je uložen v Java/Objective-C souboru jako řetězec. V prostřední části obrázku je znázorněn most mezi JavaScriptovým kódem a nativním kódem platformy tzv. Bridge. Každý JavaScriptový objekt má vytvořen svůj protějšek v nativním kódu. Například při vytváření komponenty uživatelského rozhraní je předán požadavek výše zmíněnému mostu. Most zavolá příslušný nativní kód dané platformy a zároveň se postará o vytvoření JavaScriptového objektu, se kterým je možno dále pracovat v rámci JavaScriptového kódu. Ve spodní části obrázku je znázorněn operační systém platformy, na kterém aplikace běží.[24][25]
Appcelerator Titanium se skládá z následujících komponent.

- Titanium SDK je balíček nástrojů založených převážně na jazyku Node.js. Tyto nástroje, mezi něž patří například výše zmíněný Titanium Bridge,

slouží k sestavení aplikace a následnému běhu na cílové platformě.

- Appcelerator CLI je aplikace pro příkazovou řádku pro ovládání nástrojů z Titanium SDK.

- Titanium APIs se skládá ze stovek rozhraní, pomocí kterých lze přistupovat z JavaScriptového kódu k nativním funkcím příslušné platformy.

- Titanium Studio je vývojové prostředí založené na prostředí Aptana Studio. Využití tohoto prostředí není k samotnému vývoji zapotřebí, nicméně se stará o aktualizace Titanium SDK, integraci s Appcelerator Cloud Services a mnohé další. Prostředí Aptana je rozšířením open-source prostředí Eclipse o podporu převážně webových technologií.

- Moduly slouží k rozšiřování základních rozhraní o nové funkce. Již v samotném Titanium SDK jsou některé funkcionality implementovány jako moduly. Jedná se o moduly pro prostřední vrstvu z obrázku 2.1 popsanou výše. Appcelerator poskytuje market, kde lze moduly stahovat a případně i publikovat. Některé jsou poskytovány zdarma, ale u jiných je třeba pro volné stažení mít předplacenou licenci Appcelerator Platform. Moduly třetích stran nejsou ze strany Appcelerator nijak kontrolovány.

- Appcelerator Cloud Services jsou backendové služby sloužící ke sběru a následné analýze dat. Bez předplatného pro Appcelerator Platform jsou dostupné služby omezeny. Některé nejsou dostupné vůbec a u jiných je nastaven například denní limit na počet požadavků.

2.2.3 Appcelerator Platform

Platforma Appcelerator neslouží pouze k samotné implementaci aplikace, ale k zjednodušení práce v průběhu celého životního cyklu vývoje. Platforma je postavena na Appcelerator Titanium a rozšiřuje ho převážně o funkce vhodné pro vývoj aplikací ve velkých podnicích.

Platforma je placená a je nabízena ve třech plánech. Nejnákladnějším plánem je „Enterprise“, který je stavěn danému podniku na míru. Plán určený pro týmy stojí 259 dolarů měsíčně za jednoho vývojáře. Tento plán má omezený denní počet požadavků, neposkytuje analýzu výkonnosti a stability aplikace a má mnoho dalších omezení. Nejlevnější je plán „Indie“ určený pro nezávislé vývojáře, který stojí 39 dolarů měsíčně. Oproti plánu pro týmy má mimo jiné snížen limit pro počet požadavků a není dostupná služba Arrow Builder. Kompletní seznam omezení pro jednotlivé plány lze nalézt na adrese http://www.appcelerator.com/pricing/.

Platforma se mimo Titanium skládá z následujících komponent.

- Appcelerator Studio je vývojové prostředí postavené na Titanium Studio. Podporuje například profilování aplikace na reálném zařízení, umožňuje automaticky publikovat aplikaci na Google Play a App Store atp. [26]
- Appcelerator Arrow je balíček nástrojů sloužící k jednoduchému vývoji rozhraní pro přístup a úpravu dat z klientských aplikací.
- Appcelerator Insights je aplikace zaznamenávající metriky klientských aplikací.
- Appcelerator Platform Services jsou další služby obdobně jako Cloud Services, které slouží například k automatizovanému testování aplikace.

2.2.4 Perzistentní úložiště dat

Pro perzistentní ukládání dat nabízí platforma tři různé přístupy. Každý z těchto přístupů je vhodný pro jiný případ užití.

2.2.4.1 Properties

Nejjednodušším úložištěm dat jsou Titanium.Properties. Jedná se o úložiště typu klíč/hodnota. Slouží ke vkládání dat, která lze serializovat do formátu JSON. Je vhodné především pro ukládání menších objemů dat. Jedná se o ekvivalent k SharedPreferences z platformy Android.

2.2.4.2 Databáze

Pro velké objemy strukturovaných dat je vhodným úložištěm databáze. Součástí operačních systémů iOS a Android je SQLite3 engine, který Appcelerator využívá, a tak nedochází k žádnému zásadnímu omezení výkonu.

2.2.4.3 Filesystém

Poslední možností je využít lokální filesystém. Ukládání souborů na filesystém je vhodné především v případě nutnosti ukládání obrázků nebo pokud jsou data již v souboru přijata.

2.2.5 Uživatelské rozhraní

Pro tvorbu uživatelského rozhraní je doporučováno využívat tzv. Alloy framework. Tento framework je založen na návrhovém vzoru MVC. Umožňuje vytvářet uživatelské rozhraní pomocí značkovacího jazyka XML. Model Alloy frameworku je postaven na Backbone.js. Navíc Alloy poskytuje vestavěnou podporu pro framework Underscore.js.

Stylování uživatelského rozhraní je prováděno pomocí Titanium Style Sheets. Jedná se o soubory s koncovkou „tss“, které se velmi podobají kaskádovým stylů.

Titanium SDK podporuje desítky multiplatformních komponent uživatelského rozhraní. Dokumentace u všech komponent obsahuje popis komponenty, výčet podporovaných atributů a ukázku zdrojového kódu. Jediný nedostatek je chybějící obrázek komponenty na každé podporované platformně. Kompletní výčet podporovaných komponent lze nalézt na adrese http://docs.appcelerator.com/titanium/3.0/#!/api/Titanium.UI.

Pro pozicování komponent uživatelského rozhraní slouží tři layouty. Jedná se o Absolute, Vertical a Horizontal layout. Jak již vyplývá z názvu, komponenty u Vertical layoutu jsou uspořádány pod sebou a u Horizontal layoutu vedle sebe. U Absolute layoutu jsou komponenty pozicovány relativně vůči levému hornímu rohu nebo pravému spodnímu rohu rodiče. Pro podporu rozdílných velikostí a hustot displejů lze využívat tzv. Density-independent pixelu.

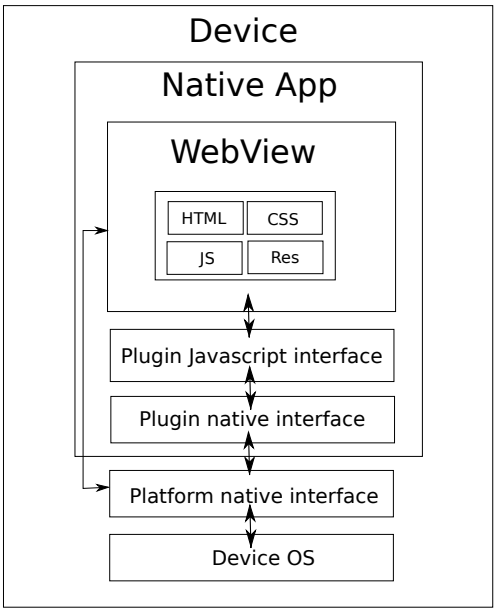
2.2.5.1 Kód specifický pro platformu

Pro oddělení sdíleného kódu od kódu pro příslušnou platformu lze využít dvou metod. První metodou je oddělování pomocí podmínek a druhou metodou je uložení souboru do podsložky s názvem dané platformy. Celkově se platforma snaží upřednostňovat konvenci nad konfigurací. Adresářová struktura je předdefinovaná, a vývojář je tak oproštěn od nutnosti psaní konfiguračních souborů.

Některé komponenty uživatelského rozhraní jsou specifické pro konkrétní platformu. Tyto komponenty jsou od verze 1.4 dávány do vlastního jmenného prostoru „iOS/Android“. Nicméně z důvodu zpětné kompatibility nebyly komponenty odstraněny ze sdíleného jmenného prostoru, ale pouze označeny jako „Deprecated“.

Na platformně Android je běžným prvkem uživatelského rozhraní tzv. Action Bar. Tento prvek byl přidán do systému Android ve verzi 3.0. Pro zobrazení Action Baru je používána knihovna AppCompact. Díky tomu lze používat moderní Material téma i na zařízeních s verzí systému Android 4.x. Na platformně iOS je ekvivalentem pro Action Bar tzv. Navigation Bar. Bohužel framework neposkytuje abstrakci těchto dvou komponent, a tak je pro jejich použití třeba psát kód specifický pouze pro jednu platformu.

Úplně chybí podpora pro navigaci pomocí tzv. Draweru. Jedná se o vyjíždějící menu, které se objevuje u mnoha aplikací na systému Android. Tento navigační vzor má svůj ekvivalent i na systému iOS.



Obrázek 2.2: Cordova architektura

2.3 Apache Cordova

Apache Cordova je nástroj pro vývoj multiplatformních mobilních aplikací s použitím běžně rozšířených technologií určených převážně pro vývoj webu. Od října roku 2012 patří produkt společnosti Apache Software Foundation a je licencován pod Apache 2.0 licencí.

Aplikace je implementována jako webové stránka s použitím technologií HTML, CSS a JavaScript. Výstupem nástroje je aplikační balíček, který lze šířit pomocí marketů daných platforem. Uživatelské rozhraní výsledné aplikace je zobrazováno v nativním webview. Aplikace má přístup k nativním funkcím dané platformy.
Informace v této podkapitole byly čerpány z oficiální dokumentace nástroje. [27]

2.3.1 Architektura

Na obrázku 2.2 je znázorněna architektura platformy Cordova. Při sestavování je pro každou platformu vytvořen projekt, ve kterém je automaticky do webview vložena úvodní stránka aplikace. Samotné Webview se stará zobrazení uživatelského rozhraní a vyhodnocování JavaScriptového kódu. Nicméně WebView je na každé platformně rozšířeno tak, aby podporovalo práci s Cordova pluginy.

Cordova plugin se skládá z JavaScriptového a nativního rozhraní. JavaScriptové rozhraní odstiňuje aplikace od konkrétní implementace na dané

platformně. Nativní část pluginu je implementována v jazyce příslušné platformy, a tak má přístup ke všem funkcionalitám, které platforma a zařízení nabízí.

2.3.2 Perzistentní úložiště dat

Nástroj Apache Cordova nabízí metody perzistentního ukládání dat dostupné v prohlížeči na cílovém zařízení. Jedná se o LocalStorage, WebSQL a IndexedDB.

LocalStorage lze použít na všech rozšířených verzích operačních systému Android, iOS i Windows Phone. Údržba WebSQL byla ukončena na konci roku 2010. [28] Webové prohlížeče na platformách Android a iOS WebSQL podporují, nicméně není zaručeno, že bude podpora zachována v dalších verzích prohlížečů. [29] IndexedDB je plně dostupné na OS Android od verze 4.4, nicméně na iOS a Windows Phone jsou jeho funkce omezeny. [30]

Další možností je použít pluginy a pro ukládání dat využít úložiště dostupné na konkrétních platformách. Příkladem je plugin dostupný na adrese https://github.com/litehelpers/Cordova-sqlite-storage, který zpřístupňuje SQLite na všechny tři platformy.

2.3.3 Uživatelské rozhraní a kód specifický pro platformu

Jak již bylo zmíněno, uživatelské rozhraní je implementováno pomocí technologií HTML, CSS popřípadě JavaScript. Lze použít jakýkoli HTML i JavaScriptový framework. Velkou nevýhodou je nenativní „look&feel“ a zároveň výkonnost omezená schopnostmi webview. Nativní vzhled lze napodobit pomocí kaskádových stylů. Je důležité poznamenat, že je nutné ošetřovat nekompatibility prohlížečů, stejně jako u vývoje webů.

Množství sdíleného kódu mezi platformami se odvíjí od způsobu přístupu k implementaci. Prvním způsobem je vytvoření jednoho uživatelského rozhraní pro všechny tři platformy. V tomto přístupu je sdíleno až 100% kódu. Druhou variantou je upravení uživatelského rozhraní pro každou platformu.

2.4 Xamarin

Společnost Xamarin poskytuje produkt Xamarin Platform sloužící k tvorbě nativních aplikací s použitím jazyka C#. Společnost byla založena roku 2011. Pomocí serveru CrunchBase získala investice v hodnotě 82 milionů dolarů. Počet registrovaných vývojářů dosahuje téměř jednoho milionu. [31] [11] [32]

Další produkty slouží převážně k usnadnění vývoje podnikových aplikací. Jedná se o produkty Xamarin Test Cloud, Xamarin Insights a Xamarin University.

Xamarin Test Cloud je framework určený na testování mobilních aplikací. Umožňuje pustit automatizované testy v cloudu na desítkách zařízení

najednou. Výsledný report obsahuje nejen informace o stabilitě aplikace, ale i o slabých místech z pohledu výkonu.

Xamarin University je portál pro vzdělávání. Na portále se lze účastnit online video tutoriálů, domluvit si sezení s jedním z expertních Xamarin vývojářů a mnohé další.

Xamarin Insights je nástroj sloužící ke sběru dat přímo ze zařízení koncových uživatelů. Vývojář má tak možnost získat nejen informace o stabilitě a výkonnosti aplikace, ale i o běžném chování uživatelů. Díky tomu lze například odhalit nedostatky v návrhu uživatelského rozhraní.

Cenová politika jednotlivých produktů se liší. Cena Xamarin Test Cloud začíná na 1000 dolarech za měsíc. Xamarin Insights jsou zatím v beta verzi a jsou poskytovány bez poplatku. Xamarin University je dostupný za roční poplatek, umožňující neomezený přístup, v hodnotě 1995 dolarů. Hlavní produkt Xamarin Platform je nabízen ve 4 verzích. Ve verzi zdarma lze vytvářet pouze aplikace s omezenou velikostí. Druhá verze je určena pro nezávislé vývojáře a stojí 25 dolarů měsíčně. Další verze za 83 dolarů měsíčně je určena pro malé a střední společnosti. Poslední verze cílená především pro velké podniky stojí 158 dolarů měsíčně. Uvedené ceny pro Xamarin Platform jsou vždy za jednoho vývojáře a platformu. Podrobný popis dostupných funkcí v jednotlivých verzích je uveden na adrese https://store.xamarin.com/. [11]

Informace v této podkapitole byly čerpány z oficiální dokumentace nástroje. [33]

2.4.1 Architektura

Pro vývoj mobilních aplikací Xamarin poskytuje dva produkty. Jedná se o Xamarin.Android a Xamarin.iOS. Oba tyto produkty jsou založeny na prostředí Mono.

Mono je open-source platforma skládající se z několika komponent. Hlavními komponentami jsou kompilátor jazyka C#, běhové prostředí a balíček knihoven. Jedná se o podobný produkt jako Microsoft .NET framework. Nicméně výhodou produktu Mono je podpora běhu nejen na systémech Microsoft Windows, ale i na systémech založených na Unixu. [34]

Způsob sestavování a běhu na systémech Android, iOS a Windows Phone se liší. Na operačním systému iOS je kód AOT kompilován do nativního strojového kódu. Na operačním systému Android je kód zkompilován do IL. Při samotném běhu programu je používán Just-In-Time kompilátor. U obou systémů je součástí aplikačního balíčku Mono framework starající se například o správu paměti. Aplikace pro operační systém Windows Phone jsou nativně psány v jazyce C#. Z toho důvodu nejsou vyžadovány žádné externí nástroje pro jejich kompilaci ani běh. Je důležité poznamenat, že od všech zmíněných odlišností je koncový vývojář naprosto odstíněn.

Xamarin framework obaluje nativní SDK platforem iOS a Android a umožňuje vývojáři přistupovat k jejich funkcím přímo ze C# kódu. Do Xamarin projektu lze vložit libovolné knihovny pro C#. Na adrese http://scan.xamarin.com lze ověřit kompatibilitu příslušné knihovny s Xamarin.Android, Xamarin.IOS i Windows Phone. Xamarin také poskytuje databázi knihoven určených přímo pro Xamarin.Android a Xamarin.IOS dostupných na adrese https://components.xamarin.com. Mnoho z těchto knihoven poskytuje rozhraní pro použití v multiplatformních aplikacích. Od verze 4.2 lze také do projektu přidat existující Java nebo Objective C knihovny.

2.4.2 IDE

V závislosti na cílových platformách si lze zvolit mezi Xamarin Studiem a Microsoft Visual Studiem. V tabulce 2.1 lze vidět, pro které platformy je možné vyvíjet v příslušných OS a vývojových prostředích.

OS	Mac OS	MS Windows	
IDE	Xamarin Studio	Xamarin Studio	Visual Studio
Android	A	A	A
iOS	A	N	A
Windows Phone	N	N	A

Tabulka 2.1: Podpora vývoje pro platformy v závislosti na OS a IDE

2.4.3 Perzistentní úložiště dat

Stejně jako u předchozích dvou produktů porovnávaných v této kapitole lze data ukládat do databáze, úložiště typu klíč/hodnota nebo do souboru. Jelikož systém Windows Phone neobsahuje databázi SQLite, je nutné ji přidat do aplikačního balíčku. Pro jednotný přístup k filesystému ze všech tří platforem lze použít například komponentu PCLStorage.

2.4.4 Uživatelské rozhraní

Uživatelské rozhraní lze vytvářet dvěma způsoby. Prvním způsobem je implementace uživatelského rozhraní nativním způsobem pro každou platformu. Výhodou tohoto přístupu je možnost využít všech dostupných komponent uživatelského rozhraní příslušné platformy. Jednou z nevýhod je pochopitelně výrazně menší podíl sdíleného kódu. Další nevýhodou je potřebná znalosti nativního způsobu vývoje pro každou platformu. Druhým způsobem je implementace uživatelského rozhraní s použitím Xamarin.Forms. Jedná se o multiplatformní nástroj zprostředkovávající společné komponenty uživatelského rozhraní ve sdíleném kódu. Jedna se pouze o abstrahující vrstvu. Na cílové

platformně je pomocí třídy dědící od třídy Renderer vytvořena nativní komponenta. Vývojář má navíc možnost jednoduše vytvořit vlastní Renderer pro libovolnou nativní komponentu uživatelského rozhraní. Projekt využívající Xamarin.Forms běží pouze na zařízeních se systémem Android 4+, iOS 6.1+ a Windows Phone 8.

Při použití Xamarin.Forms lze vytvářet uživatelské rozhraní přímo v C# kódu nebo pomocí jazyka XAML. Výhodou vývoje v jazyku XAML je okamžité zobrazení rozložení jednotlivých prvků přímo v IDE. Naopak nevýhodou je chybějící podpora pro nastavování například šířky prvků v závislosti na hodnotách známých pouze za běhu aplikace. Pro implementaci běžných vzorů pro navigaci v aplikaci slouží tzv. Pages. Podporována je například navigace pomocí záložek, carousel, boční vyjíždějící menu atp. Pro rozložení prvků na stránce slouží AbsoluteLayout, Grid, RelativeLayout a StackLayout. Pomocí těchto layoutů lze relativně jednoduše vytvořit i komplikované uživatelské rozhraní. Xamarin.Forms poskytuje společné prvky uživatelského rozhraní, mezi něž patří například Label, Button, Entry, Image, ProgressBar, ListView, DatePicker a mnohé další.

2.4.5 Kód specifický pro platformu

Xamarin nabízí dva způsoby přístupu ke sdílení kódu mezi jednotlivými platformami.

2.4.5.1 Sdílený projekt

Jedná se o nejjednodušší způsob sdílení kódu. V solution je vytvořena složka se sdíleným kódem a projekty pro každou cílovou platformu. Při kompilaci je sdílená složka přikompilována do kódu každé platformy. Výhodou tohoto přístupu je možnost využít direktiv kompilátoru pro oddělení kódu, který je specifický pro platformu. Pro sdílenou složku není vytvořen samostatný DLL soubor, a tudíž se tento přistup nehodí pro vytváření knihoven.

2.4.5.2 PCL

U tohoto způsobu je i pro sdílený kód vytvořen samostatný projekt. Tento projekt je následně referencován z projektů platforem. Výhodou je, že i pro sdílený kód je vytvořen DLL soubor. Nevýhodou je, že kód specifický pro konkrétní platformu musí být v projektu dané platformy. Jinými slovy nelze využívat direktivi kompilátoru. Tento problém se řeší vytvořením rozhraní, které je následně implementováno v příslušných projektech. Další nevýhodou je dostupnost pouze podmnožiny .NET frameworku.

2.5 Podporované verze OS

Všechny tři nástroje podporují OS Android od verze 2.3. Nicméně při použití nástroje Xamarin.Forms je minimální podporovaná verze 4.0. Nižší verzi systému používá necelých 7% zařízení.

Nástroj Apache Cordova podporuje operační systém iOS od verze 5.1, zbylé dva nástroje od verze 7.1. Podle statistik App Store je 98% zařízeních vybaveno verzí 7.1 nebo vyšší.

U všech třech nástrojů je podporován Windows Phone 8. Starší verzi OS používá méně než 5% zařízení. Jak již bylo zmíněno, u nástroje Appcelerator je podpora v beta verzi.

Xamarin podporuje naprostou většinu API nativních SDK. Velkou výhodou je, že podpora pro nové verze OS bývá zpravidla přidána ve stejný den, kdy je systém oficiálně zveřejněn.

2.6 Shrnutí

U všech třech nástrojů je sestavování pro operační systém iOS nutné provádět na zařízení s Mac OS. Nicméně nástroj Xamarin umožňuje spustit tzv. Build Host na systému Mac OS a následně se na něj po síti napojit ze systému Windows. Díky tomu lze vyvíjet aplikace pro všechny tři platformy ve vývojovém prostředí Microsoft Visual Studio. Tento proces má dvě podmínky. První je vlastnictví Proffesional licence Microsoft Visual Studia. Druhou je připojení obou zařízení na jedné lokální síti.

Z hlediska výkonu je na tom nejlépe nástroj Xamarin. Kód pro platformu Windows Phone není nijak upravován oproti nativnímu vývoji, kód pro platformu iOS je kompilován do strojového kódu a kód pro platformu Android je kompilován do IL. Naopak nejhůře je na tom nástroj Apache Cordova, ve kterém je uživatelské rozhraní vykreslováno do WebView.

Další výhodou nástroje Xamarin je podpora vývoje pro wearables. V rámci této práce nebude aplikace pro wearables implementována, nicméně s nástrojem Xamarin je možno aplikaci o podporu v budoucnosti rozšířit.

Ze srovnání jednoznačně vychází nejlépe nástroj Xamarin. Jeho jedinou zásadní nevýhodou je nutnost pořízení placeného předplatné. Nicméně pro akademické účely společnost Xamarin poskytuje Bussiness licenci zdarma. V případě nutnosti vývoje pro operační systém Android ve verzi 2.3 je třeba zvážit použití nástroje Appcelerator. V nástroji Xamarin je tato verze podporována pouze v případě implementace uživatelského rozhraní nezávisle pro každou cílovou platformu. Výhodou nástroje Apache Cordova je rychlejší a levnější vývoj, nicméně nástroj je vhodný pouze pro vývoj aplikací bez nativního „look&feel“.

Kapitola 3

Analýza a návrh

V první části této kapitoly jsou sepsány funkční a nefunkční požadavky na cílovou aplikaci. Ve druhé části jsou podrobně rozebrány jednotlivé případy užití. Třetí část se zabývá návrhem uživatelského rozhraní a porovnáním s konkurenčními aplikacemi. V poslední části je popsána architektura řešení a databázový model.

3.1 Úvod

Cílem první části této práce byl výběr vhodného nástroje pro tvorbu multiplatformních aplikací v komerční sféře. Ze srovnání vyplynulo, že nejvhodnějším nástrojem je Xamarin. Je běžné, že se nástroj zdá být pro nějaký účel ideální, ale při praktickém použití se objeví závažné nedostatky. Z toho důvodu je nutné nástroj ověřit v praxi.

Ve druhé části této práce je nástroj použit pro tvorbu netriviální aplikace pro tři nejrozšířenější mobilní operační systémy. Jedná se o aplikaci pro zaznamenávání odpracovaného času.

3.2 Analýza požadavků

Nejprve je nutné jasně definovat požadavky na cílovou aplikaci. Aplikace je cílena především na vývojáře, kteří potřebují měřit a reportovat čas strávený nad jednotlivými úkoly. Vy zbytku této kapitoly je výrazem „ticket“ míněn jeden úkol a výrazem „záznam“ je míněno jedno měření.

3.2.1 Funkční požadavky

Registrace – Aplikace musí uživateli umožnit vytvořit si vlastní účet.

Přihlášení – Aplikace musí uživateli umožnit přístup pouze po zadání platných přihlašovacích údajů.

Odhlášení – Aplikace musí uživateli umožnit odhlásit se a smazat veškerá citlivá data.

Vytvoření ticketu – Aplikace musí uživateli umožnit vytvořit nový ticket s vlastním názvem. Ticketu může být přiřazen maximálně jeden projekt.

Smazání ticketu –Aplikace musí uživateli umožnit smazat ticket a všechny příslušné záznamy.

Vytvoření záznamu – Aplikace musí umožnit vytvořit nový záznam. Záznam se vždy váže právě na jeden ticket. Záznam musí jít vytvořit jak k novému, tak k existujícímu ticketu.

Měření času – Aplikace musí umožňovat měřit čas.

Uložení lokace – Při vytváření záznamu musí aplikace umožnit uložit lokaci, na které se uživatel právě nachází. Nalezení lokace musí být prováděno automaticky s využitím vestavěných senzorů cílového zařízení.

Smazání záznamu – Aplikace musí uživateli umožnit smazat libovolný záznam.

Vytvoření projektu – Aplikace musí uživateli umožnit vytvořit nový projekt s vlastním názvem.

Zobrazení ticketů – Aplikace musí být schopna zobrazit seznam ticketů.

Zobrazení záznamů – Aplikace musí být schopna zobrazit seznam záznamů.

Nastavení e-mailové adresy – Aplikace musí uživateli umožnit nastavit e-mailovou adresu pro export dat.

Export do CSV – Aplikace musí být schopna exportovat data do formátu CSV a následně je odeslat e-mailem.

Export do XLS – Aplikace musí být schopna exportovat data do souboru ve formátu XLS a následně soubor odeslat e-mailem.

Nastavení URL pro Redmine – Aplikace musí uživateli umožnit nastavit vlastní URL Redmine serveru.

Zadání přihlašovacích údajů pro Redmine – Aplikace musí uživateli umožnit zadat vlastní přihlašovací údaje k nástroji Redmine.

Import dat z nástroje Redmine – Aplikace musí umožnit importovat projekty, tickety a záznamy z nástroje Redmine.

Export dat do nástroje Redmine – Aplikace musí umožnit exportovat projekty, tickety a záznamy do nástroje Redmine.

Synchronizace se serverem – Aplikace musí být schopna průběžně synchronizovat projekty, tickety a záznamy se serverem.

3.2.2 Nefunkční požadavky

Kompatibilita s OS Android – Aplikace musí podporovat běh na systému Android. Přípustná jsou omezení funkcionalit dané platformou.

Kompatibilita s OS iOS – Aplikace musí podporovat běh na systému iOS. Přípustná jsou omezení funkcionalit dané platformou.

Kompatibilita s OS Windows Phone – Aplikace musí podporovat běh na systému Windows phone. Přípustná jsou omezení funkcionalit dané platformou.

Intuitivnost – Aplikace musí dodržovat zvyklosti uživatelského rozhraní na každé cílové platformně.

Podpora více zařízení – Aplikace musí umožnit registraci více zařízení jednoho uživatele.

Dostupnost připojení k internetu – Při přihlašování musí mít aplikace přístup k internetu.

Offline režim – Aplikace musí být schopna fungovat bez připojení k internetu. Bez připojení není pochopitelně vyžadováno fungování požadavků, které ze své podstaty potřebují připojení k internetu.

3.3 Případy užití

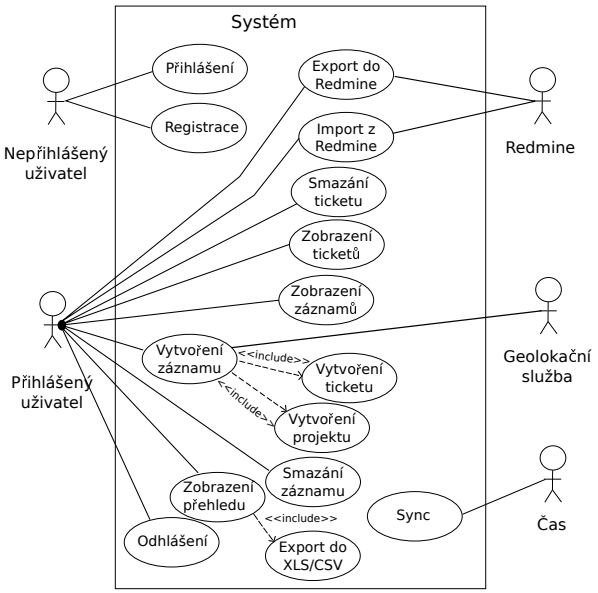
Na diagramu 3.1 jsou zobrazeny hlavní případy užití aplikace.

3.3.1 Případ užití – Přihlášení

Hlavní aktéři: nepřihlášený uživatel
Vedlejší aktéři: žádní
Vstupní podmínky: Připojení k Internetu je dostupné.

1. Případ užití je zahájen spuštěním aplikace.
2. Aplikace zobrazí pole pro vyplnění přihlašovacích údajů.
3. Uživatel vyplní své přihlašovací údaje a stiskne tlačítko „Přihlásit se“.
4. Aplikace odešle na server požadavek pro ověření přihlašovacích údajů.
5. Server ověří přihlašovací údaje a informuje aplikaci.
6. Aplikace přihlásí uživatele.

Alternativní scénáře:



Obrázek 3.1: Diagram případů užití

(a) Viz kroky 1-4.

(b) Uživatelské údaje jsou nesprávné. Aplikace informuje uživatele o neúspěšnosti přihlášení. Pokračuje krokem č.2.

Výstupní informace: Uživatel je informován o (ne)úspěšnosti přihlášení.

Případ užití registrace je téměř identický.

3.3.2 Případ užití – Odhlášení

Hlavní aktéři: přihlášený uživatel
Vedlejší aktéři: žádní
Vstupní podmínky: žádné

1. Případ užití je zahájen stiskem tlačítka „Odhlásit“.
2. Aplikace zobrazí potvrzovací dialog.
3. Uživatel potvrdí odhlášení.
4. Aplikace odhlásí uživatele a smaže veškerá data vázaná k tomuto uživateli.

Alternativní scénáře:

(a) Viz kroky 1 a 2.

(b) Uživatel dialog nepotvrdí.

(c) Aplikace schová dialog.

Výstupní informace: V případě průchodu hlavním scénářem je uživatel přesměrován na přihlašovací obrazovku a veškerá data vázaná k tomuto uživateli jsou smazána.

3.3.3 Případ užití – Vytvoření záznamu

Hlavní aktéři: přihlášený uživatel
Vedlejší aktéři: Geolokační služba
Vstupní podmínky: V případě žádosti o uložení lokace je nutné připojení k internetu.

1. Případ užití je zahájen přechodem na obrazovku „Nový záznam“.
2. Aplikace uživateli zobrazí pole pro zadání názvu ticketu.
3. Uživatel vyplní název ticketu.
4. Aplikace uživateli zobrazí pole pro výběr existujícího nebo vytvoření nového projektu.
5. Uživatel vybere projekt.
6. Aplikace uživateli zobrazí možnost získání a uložení lokace.
7. Nepovinný krok:
   a) Uživatel zvolí možnost uložení lokace.
   b) Aplikace získá souřadnice pomocí senzorů zařízení.
   c) Aplikace kontaktuje geolokační službu a požádá o převedení souřadnic na adresu.
   d) Geolokační služba informuje aplikaci o adrese odpovídající poskytnutým souřadnicím.
8. Aplikace zobrazí tlačítko pro zahájení měření času.
9. Kroky 9 - 13 se mohou opakovat 0-N krát. Uživatel stiskne tlačítko pro zahájení měření času.
10. Aplikace uživatele každou sekundu informuje o uběhnutém čase.
11. Aplikace zobrazí tlačítko pro pozastavení měření.

12. Uživatel stiskne tlačítko pro zastavení měření.
13. Aplikace pozastaví měření času.
14. Uživatel stiskne tlačítko uložit.
15. Aplikace uloží záznam do persistentního uložiště.

Alternativní scénáře:

- Scénář vytvoření projektu
  (a) V kroku 5 uživatel zvolí možnost vytvoření nového projektu.
  (b) Aplikace zobrazí pole pro zadání názvu projektu.
  (c) Uživatel zadá název projektu.
  (d) Scénář pokračuje krokem 6.
- Scénář vytvoření záznamu k existujícímu ticketu
  (a) Případ užití je zahájen kliknutím na existující ticket nebo záznam.
  (b) Aplikace uživateli zobrazí neaktivní pole s názvem ticketu a projektu.
  (c) Scénář pokračuje krokem 6.
- Scénář nedostupný senzor pro získání lokace
  (a) V kroku 7 uživatel zvolí možnost uložení lokace.
  (b) Aplikace nemá povolený přístup k senzorům zařízení.
  (c) Aplikace informuje uživatele o nedostupnosti senzorů.
  (d) Scénář pokračuje krokem 8.
- Scénář nedostupnost geolokační služby
  (a) V kroku 7c se nepodaří kontaktovat geolokační službu.
  (b) Aplikace informuje uživatele o nedostupnosti služby.
  (c) Scénář pokračuje krokem 8.

Výstupní informace: Nový záznam je uložen v persistentním úložišti.

3.3.4 Případ užití – Zobrazení ticketů/Smazání ticketu

Hlavní aktéři: přihlášený uživatel
Vedlejší aktéři: žádní
Vstupní podmínky: žádné

1. Případ užití je zahájen přechodem na obrazovku „Seznam ticketů“.
2. Aplikace zobrazí seznam všech ticketů.
3. Nepovinný krok:
   a) Uživatel vyvolá kontextovou nabídku pro prvek seznamu. Způsob vyvolání nabídky se odvíjí od platformy.
   b) Aplikace zobrazí kontextovou položku pro smazání ticketu.
   c) Uživatel stiskne kontextovou položku.
   d) Aplikace smaže daný ticket a všechny navazující záznamy.
4. Nepovinný krok:
   a) Uživatel klikne na položku seznamu.
   b) Aplikace uživatele přesměruje na případ užití „Vytvoření záznamu“.

Alternativní scénáře:

(a) Případ užití je zahájen přechodem na obrazovku „Seznam ticketů“.

(b) V úložišti nejsou zatím žádné tickety. Aplikace informuje uživatele, že ještě nevytvořil žádný ticket.

Výstupní informace: V případě mazání ticketu je ticket smazán z persistentního uložiště.

3.3.5 Případ užití – Zobrazení záznamů/Smazání záznamu

Hlavní aktéři: přihlášený uživatel
Vedlejší aktéři: žádní
Vstupní podmínky: žádné

1. Případ užití je zahájen přechodem na obrazovku „Poslední záznamy“.
2. Aplikace zobrazí posledních 10 záznamů. Záznamy jsou řazeny od nejnovějšího k nejstaršímu.
3. Nepovinný krok:

   a) Uživatel vyvolá kontextovou nabídku pro prvek seznamu. Způsob vyvolání nabídky se odvíjí od platformy.
   b) Aplikace zobrazí kontextovou položku pro smazání záznamu.
   c) Uživatel stiskne kontextovou položku.
   d) Aplikace zobrazí potvrzovací dialog.
   e) Pokud uživatel dialog potvrdil, aplikace smaže daný záznam. V opačném případě pouze schová kontextovou nabídku.

4. Nepovinný krok:
   a) Uživatel klikne na položku seznamu.
   b) Aplikace uživatele přesměruje na případ užití „Vytvoření záznamu“.

Alternativní scénáře:

(a) Případ užití je zahájen přechodem na obrazovku „Poslední záznamy“.

(b) V úložišti nejsou zatím žádné záznamy. Aplikace informuje uživatele, že ještě nevytvořil žádný záznam.

Výstupní informace: V případě mazání záznamu je záznam smazán z persistentního uložiště.

3.3.6 Případ užití – Export do XLS/CSV

Hlavní aktéři: přihlášený uživatel
Vedlejší aktéři: žádní
Vstupní podmínky: Systém podporuje přikládání příloh k e-mailové zprávě. Připojení k internetu je dostupné nebo systémová aplikace pro odesílání e-mailů podporuje běh v offline režimu.

1. Případ užití je zahájen stiskem tlačítka „Odeslat“.
2. Aplikace zobrazí seznam dostupných cílových formátů na dané platformě.
3. Uživatel zvolí požadovaný formát.
4. Aplikace exportuje uložiště do požadovaného formátu. Aplikace požádá operační systém o odeslání e-mailové zprávy s přiloženým souborem v požadovaném formátu.
5. Systém nalezne implicitní aplikaci pro odesílání e-mailů.
6. Zbytek scénáře se odvíjí od cílové platformy.

Alternativní scénáře: Žádné

Výstupní informace: V případě úspěchu je zpráva odeslána na zvolenou e-mailovou adresu. Uživatel je informován o stavu operace.

3.3.7 Případ užití – Zobrazení přehledu

Hlavní aktéři: přihlášený uživatel
Vedlejší aktéři: žádní
Vstupní podmínky: žádné

1. Případ užití je zahájen přechodem na obrazovku „Přehled“.
2. Aplikace zobrazí informace o počtu odpracovaných hodin v tomto měsíci. Dále zobrazí seznam projektů s počtem odpracovaných hodin u každého projektu.
3. Aplikace zobrazí dvě pole pro výběr data. Pole slouží pro výběr období zobrazeného přehledu.
4. Uživatel klikne na pole pro výběr data.
5. Aplikace zobrazí komponentu pro výběr data.
6. Uživatel vybere datum.
7. Aplikace aktualizuje záznamy pro zadané rozmezí.
8. Nepovinný krok: «include» Export do XLS/CSV

Alternativní scénáře:

(a) Viz kroky 1-7.

(b) Datum „do“ předchází datu „od“. Aplikace informuje uživatele o chybném výběru a ponechá v poli původní datum.

Výstupní informace: V případě úspěchu při importu dat jsou data ze server Redmine uložena v lokálním úložišti. Při importu/exportu je uživatel informován o výsledku operace.

3.3.8 Případ užití – Export do Redmine/Import z Redmine

Hlavní aktéři: přihlášený uživatel
Vedlejší aktéři: žádní
Vstupní podmínky: Aktivní připojení k internetu. V systému Redmine musí být povolena komunikace přes REST API.

1. Případ užití je zahájen přechodem na obrazovku „Nastavení“.
2. Aplikace zobrazí pole pro URL systému Redmine.
3. Uživatel vyplní URL systému Redmine
4. Aplikace zobrazí pole pro přihlašovací údaje.
5. Uživatel vyplní své přihlašovací údaje k systému Redmine.
6. Aplikace zobrazí tlačítka „Import“/„Export“.
7. Uživatel stiskne jedno z tlačítek.
8. Aplikace se pokusí přihlásit uživatele do systému Redmine.
9. Aplikace v závislosti na operaci importuje data ze systému Redmine nebo Exportuje data do systému Redmine.
10. Aplikace informuje uživatele o výsledku komunikace.

Alternativní scénáře:

- Chybné přihlašovací údaje
  (a) Viz kroky 1-8.
  (b) Údaje pro přihlášení do systému Redmine jsou neplatné. Aplikace informuje uživatele o neplatnosti přihlašovacích údajů.
  (c) Scénář pokračuje krokem 4.
- Chybná URL systému Redmine
  (a) Viz kroky 1-8.
  (b) Na zadané URL se nenachází systém Redmine. Aplikace uživatele informuje o neplatnosti poskytnuté URL.
  (c) Scénář pokračuje krokem 2.

Výstupní informace: V případě úspěchu jsou data importována z nebo exportována do systému Redmine a uživatel je informován o výsledku operace.

3.3.9 Případ užití – Synchronizace

Hlavní aktéři: Čas
Vedlejší aktéři: žádní
Vstupní podmínky: Aktivní připojení k internetu. Aplikace se nachází v popředí. Uživatel je přihlášen.

1. Případ užití je automaticky zahájen při přihlášení a následně každou minutu.
2. Aplikace požádá server o veškerá nová data od poslední synchronizace.
3. Server aplikaci pošle data.
4. Aplikace data uloží do persistentního uložiště.
5. Aplikace vybere všechna data, která byla lokálně aktualizovaná od poslední synchronizace.
6. Aplikace pošle data na server.
7. Server informuje aplikaci o přijetí dat.
8. Aplikaci aktualizuje datum poslední synchronizace.

Alternativní scénáře:

- Server je nedostupný
  (a) Viz krok 1.
  (b) Server neodpovídá.
  (c) Scénář je ukončen.

Výstupní informace: V případě úspěchu jsou data synchronizována.

3.4 Návrh uživatelského rozhraní

Aplikace musí být podporována třemi různými platformami. Uživatelé každé platformy mohou být zvyklí na rozdílné chování. Pro minimalizaci kódu specifického pro každou platformu by bylo dobré používat pouze základní prvky uživatelského rozhraní, které jsou dostupné na všech cílových platformách.

Aplikace se skládá z 8 obrazovek. Jelikož je většina těchto obrazovek na stejné úrovni, nejvhodnější navigační prvek je běžné menu. Navigace pomocí menu je vhodná především pro aplikace s více jak 4 obrazovkami na stejné úrovni. Pro 4 a méně obrazovek je zpravidla používána navigace pomocí záložek.

Obrázek 3.2: Wireframe přihlášení

Obrázek 3.3: Wireframe nový záznam

Navigaci pomocí menu lze jednoduše přizpůsobit zvyklostem na danou platformu. Pro systém iOS a Android bude menu vyjíždět z levé části displeje. Pro systém Windows Phone bude menu samostatná obrazovka spustitelná z nativní kontextové nabídky.

Veškeré wireframe jsou vytvořeny pouze pro iOS. Pro ostatní operační systémy budou prvky uživatelského rozhraní nahrazeny svými ekvivalenty.

3.4.1 Přihlášení/Registrace

Na obrázku 3.2 je zobrazen wireframe přihlašovací obrazovky. Obrazovka obsahuje pole pro přihlašovací jméno a pole pro heslo. Dále obsahuje tlačítko pro zobrazení registračního formuláře. Obrazovka registrace je velmi obdobná, pouze je změněna textace tlačítek.

Obrazovka naplňuje tyto požadavky:

- registrace,
- přihlášení.

3.4.2 Vytvoření záznamu

Na obrázku 3.3 je zobrazen wireframe obrazovky pro vytvoření nového záznamu. Obrazovka obsahuje pole pro název ticketu a název projektu. Dále obsahuje Switch indikující, zda uživatel chce tento záznam synchronizovat se serverovým úložištěm nebo mít data uložena pouze lokálně. Druhý Switch indikuje, zda chce uživatel uložit k záznamu současnou polohu. Úplně vespodu se nachází tlačítko pro spuštění měření odpracovaného času. Čas je zobrazován ve vrchní části obrazovky. V kontextovém menu se nachází tlačítko pro uložení záznamu.

Obrazovka naplňuje tyto požadavky:

Obrázek 3.4: Wireframe poslední záznamy

Obrázek 3.5: Wireframe všechny tickety

- vytvoření ticketu,
- vytvoření záznamu,
- vytvoření projektu,
- měření času,
- uložení lokace.

3.4.3 Zobrazení ticketů

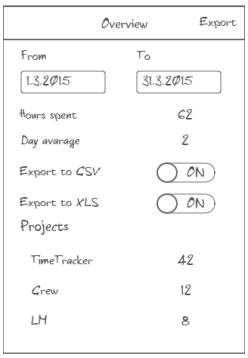
Na obrázku 3.5 je zobrazen wireframe se seznamem ticketů. Každá položka seznamu znázorňuje jeden ticket. U každého ticketu je uveden jeho název, projekt a počet odpracovaných hodin. Při vyvolání kontextové nabídky může uživatel smazat daný ticket a všechny navázané záznamy. Způsob vyvolání a vzhled kontextové nabídky je specifický pro každou platformu.

Obrazovka naplňuje tyto požadavky:

- zobrazení ticketů,
- smazání ticketu.

3.4.4 Zobrazení záznamů

Na obrázku 3.5 je zobrazen wireframe se seznamem záznamů. Položky jsou rozděleny do skupin podle dnů. Nad každou skupinou je napsáno datum a součet naměřených časů pro daný den. Položky jsou řazeny od nejnovější k nejstarší. Každá položka seznamu zobrazuje jeden záznam. U každé položky je uveden název ticketu, název projektu a naměřený čas. Podobně jako u seznamu ticketů lze vyvolat kontextovou nabídku a libovolný záznam smazat.



Obrázek 3.6: Wireframe přehled

Obrázek 3.7: Wireframe nastavení

Obrazovka naplňuje tyto požadavky:

- zobrazení záznamů,
- smazání záznamu.

3.4.5 Zobrazení přehledu

Na obrázku 3.6 je zobrazen wireframe s přehledem odpracovaných hodin. Navrchu jsou zobrazeny dvě komponenty pro výběr data. Pomocí těchto komponent uživatel může filtrovat zobrazená data. Na obrazovce je dále zobrazen počet odpracovaných hodin a průměrný počet odpracovaných hodin za jeden den. V prostřední části obrazovky se nachází dva Switche indikující formát, do kterého chce uživatel exportovat data. Ve spodní části se nachází seznam projektů, ke kterým uživatel v daném časovém úseku vytvořil alespoň jeden záznam. V kontextové nabídce se nachází tlačítko pro export dat do cílového formátu a následné odeslání e-mailem.

Obrazovka naplňuje tyto požadavky:

- export do CSV,
- export do XLS.

3.4.6 Nastavení

Na obrázku 3.7 se nachází wireframe s nastavením aplikace. V první části je pole pro změnu e-mailové adresy pro export dat. Ve druhé části se nachází pole pro zadání údajů k systému Redmine. Konkrétně se jedná o URL, přihlašovací jméno a heslo. V poslední části se nachází tlačítka pro zahájení exportu a importu dat. V kontextové nabídce je tlačítko pro odhlášení uživatele z aplikace.

Obrazovka naplňuje tyto požadavky:

- odhlášení,
- nastavení e-mailové adresy,
- nastavení URL pro Redmine,
- zadání přihlašovacích údajů pro Redmine,
- import dat z nástroje Redmine,
- export dat do nástroje Redmine.

3.5 Konkurenční produkty

3.5.1 Toggle iOS

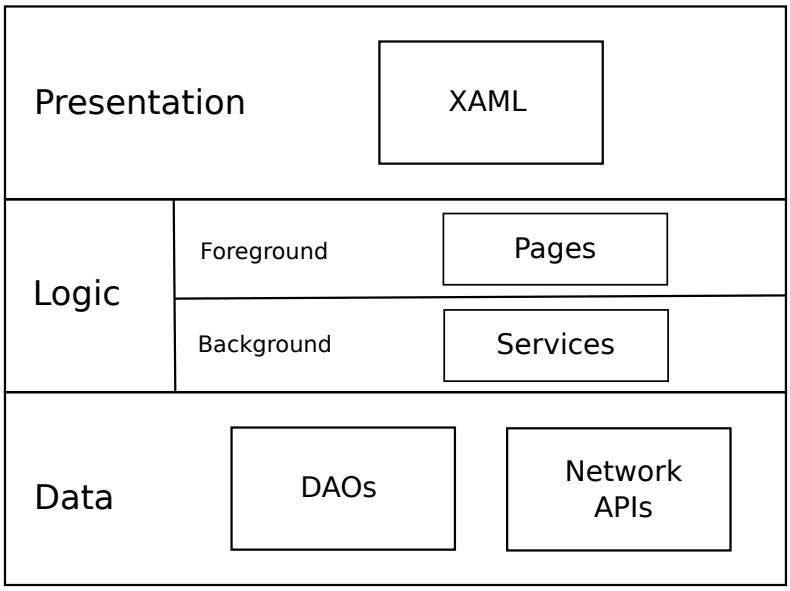
Aplikace Toggle je přímou konkurencí vyvíjené aplikace.

+ Data jsou zobrazena v přehledných grafech.
– Kontextové akce v seznamech se nechovají nativně.
– V přehledu nelze zobrazit data pro libovolný časový úsek.
– Hlavní menu se nechová nativně. Menu nepodporuje tzv. OnEdge gesto, což může mít za následek neúmyslnou aktivaci kontextové nabídky v seznamu položek.

3.5.2 Jiffy Android

Aplikace Jiffy je pravděpodobně největším konkurentem. Na Google Play má přes 100 000 stažení. [35]

+ Data jsou zobrazena v přehledných grafech.
+ Podporuje aktivaci měření pomocí NFC.
– Nepodporuje synchronizaci pro více zařízení.
– V základním nastavení nejsou data nijak zálohována. Není podporována záloha do cloudu. Data lze zálohovat pouze do souboru.
– Při měření času nejsou zobrazeny sekundy. Uživatel tak může mít dojem, že měření vůbec neprobíhá.



Obrázek 3.8: Architektura

3.5.3 RescueTime Android

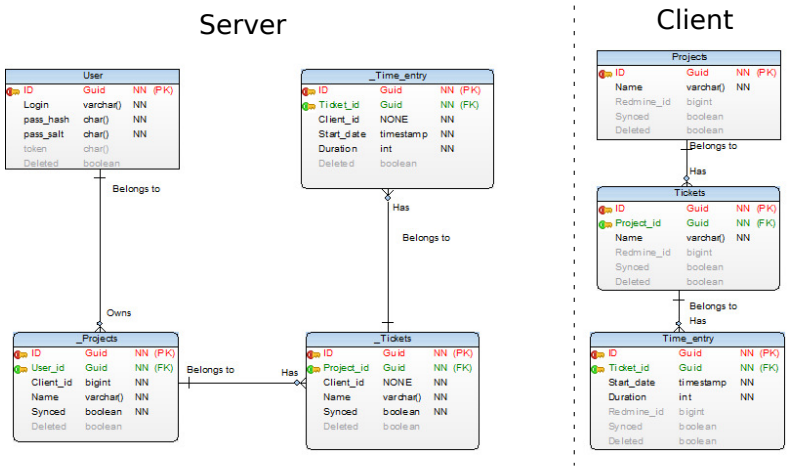
Aplikace RescueTime slouží k zaznamenávání času stráveného v aplikacích. Ze vzhledu a chování aplikace je na první pohled patrné, že je aplikace vyvinuta v multiplatformním nástroji.

- Při přihlašování není zobrazen indikátor aktivity, a tak uživatel může mít dojem, že se nic neděje.
- Komponenty uživatelského rozhraní nejsou nativní.
- Chybí podpora přecházení mezi záložkami pomocí gest.
- Aplikace chybně pracuje se zásobníkem pro tlačítko zpět. Lze se dostat do nekonečného cyklu.

Hlavní nedostatky aplikací Toggle a RescueTime vycházely z nenativního chování a vzhledu komponent uživatelského rozhraní. Cílem tedy bude pokusit se pomocí multiplatformního nástroje vyvinout aplikaci s nativním vzhledem a chováním na všech třech platformách.

3.6 Architektura aplikace

Jednoduchý pohled na architekturu lze vidět na obrázku 3.8. Aplikace je tvořena několika oddělenými vrstvami. Každá z vyobrazených vrstev komunikuje pouze se sousedícími vrstvami.



Obrázek 3.9: Databázové schéma

Ve vrchní části obrázku se nachází vrstva prezentační. Jedná se o vrstvu uživatelského rozhraní, se kterým interaguje uživatel aplikace. V této vrstvě jsou definovány komponenty uživatelského rozhraní a jejich vzájemné rozložení.

V prostředí části obrázku se nachází vrstva s byznys logikou. Pro mobilní aplikace je kritické, aby časově náročné operace nebyly prováděny na UI vlákně aplikace. Při provádění těchto operací na UI vlákně aplikace nereaguje na vstupy od uživatele a je výrazně zvýšeno riziko násilného ukončení aplikace operačním systémem. Z toho důvodu je vrstva rozdělena do dvou částí: foreground a background. Ve foreground části se nachází operace, které běží na UI vlákně aplikace. Naopak veškeré operace v background části jsou prováděny na pracovním vlákně aplikace.

Ve spodní části obrázku se nachází datová vrstva. Cílem této vrstvy je poskytovat data byznys vrstvě, ale zároveň skrýt způsob jejich uložení. V této vrstvě je tedy například zajištěn přístup k lokální databázi, ale i přístup k databázi serverové.

3.7 Databáze

Na obrázku 3.9 je znázorněno databázové schéma. Na straně klienta jsou tři entity. Jedná se o projekty, tickety a časové záznamy. Na straně serveru je navíc entita uživatel.

Databázové schéma aplikace je na první pohled velmi jednoduché. V podstatě jedinou komplikací je vyřešení přidělování identifikátoru jednotlivým entitám. Jelikož mohou být entity vytvářeny bez připojení k internetu, nelze přidělovat identifikátor na straně serveru ve chvíli, kdy je entita vytvářena.

Byly zváženy následující přístupy k přidělování identifikátoru.

GUID – Globální unikátní identifikátor, který lze generovat pseudonáhodným generátorem. Pravděpodobnost vygenerování dvou stejných identifikátorů je tak nízká, že lze předpokládat jejich unikátnost. Každý klient tak může generovat identifikátory na své straně.

Identifikátory generované na serveru – U klienta jsou vytvářeny pouze dočasné identifikátory, které jsou při synchronizaci se serverem aktualizovány. Je nutno aktualizovat i cizí klíče. Navíc je nutné ošetřit kolizi, kdy serverem vygenerovaný klíč na straně klienta již existuje.

HiLo – Každému klientovi je přidělen rozsah identifikátorů při první komunikaci se serverem. V případě, že klient spotřebuje všechny klíče, musí znovu kontaktovat server o přidělení nového rozsahu.

Mobilní aplikace nemusí mít vždy kvalitní přístup k internetu, a tak je velmi důležité minimalizovat nutnost komunikace po síti. Z těchto přístupů je pro aplikaci tudíž nejvhodnější GUID. Další výhodou tohoto přístupu je jednoduchost implementace.

3.8 Webové služby

Nejběžněji používaná architektura webových služeb v kontextu mobilních aplikací je jednoznačně architektura REST. Je možné, že bude aplikace v budoucnu rozšířena o webové rozhraní, a tak se jako nejvhodnější formát pro komunikaci jeví JSON.

Jelikož pro některé uživatele mohou být data v aplikaci citlivá, je nutno aplikaci řádně zabezpečit proti zneužití. Jedním ze základních útoků na mobilní aplikace respektive webové služby, je odposlouchávání komunikace. Druhým běžným útokem je získání přihlašovacích údajů z klientského zařízení. V nativních aplikacích nejsou data aplikace přístupná ostatním aplikacím v systému. Nicméně například v případě ztráty telefonu může útočník data získat.

Pro zabezpečení proti zmíněným typům útoků je nutno provést následující protiopatření. Zaprvé server musí povolit komunikaci s API jen přes protokol HTTPS, který zajišťuje šifrovanou komunikaci. Zadruhé přihlašovací údaje nesmí být uloženy v persistentním úložišti klientského zařízení. Pro komunikaci se serverem musí být použit token, který klientská aplikace získá při přihlášení. Tímto způsobem nelze získat přihlašovací údaje a zároveň lze zneplatnit token pro konkrétní zařízení.

Webová služba má celkem 15 endpointů. Jedná se například o endpointy pro registraci, přihlášení, získání všech projektů, vytvoření projektu atd. Kompletní specifikace se nachází na přiloženém DVD.

Kapitola 4

Realizace a testování

V první části této kapitoly jsou popsány jednotlivé nástroje a knihovny, které byly použité v průběhu vývoje. Dále je v kapitole krátce popsán způsob realizace REST rozhraní a příprava vývojového prostředí. V další části kapitoly je popsáno rozdílné chování a vzhled na cílových platformách. V poslední části jsou popsány metody a výsledky testování.

4.1 Použité nástroje

4.1.1 IDE

Pro vývoj mobilní aplikace byly použity tři vývojová prostředí. Jednalo se o Xamarin Studio na operačním systému Windows, Xamarin Studio na operačním systému Mac OS a Microsoft Visual Studio 2013. Všechna tři prostředí jsou kompatibilní s projektem vytvořeném ve Visual Studio, a tak mezi nimi lze snadno přepínat.

Od dubna 2015 byla zpřístupněna studentská Business licence, jejíž součástí je plugin do Microsoft Visual Studia. Tento plugin umožňuje vyvíjet aplikaci pro všechny tři platformy ve vývojovém prostředí Microsoft Visual Studio. Nutností je mít na lokální síti Mac OS s nástrojem pro sestavení výsledné aplikace.

Pro vývoj serverové části aplikace lze použít jakékoli vývojové prostředí, které podporuje jazyk Java. Pro účely této práce bylo použito prostředí IntelliJ.

4.1.2 Apache Maven

Nástroje Apache Maven slouží k automatizaci sestavování aplikací. V XML souboru lze specifikovat veškeré závislosti a způsob sestavení projektu. Nástroj už je takřka standardem pro sestavování projektů v jazyce Java. Nicméně nástroj lze použít i pro projekty například v jazycích C# a Python.

4.1.3 Fiddler

Nástroj Fiddler je proxy server, pomocí kterého lze sledovat komunikaci po síti. Je vhodný pro ladění a testování webových a mobilních aplikací. Umožňuje například upravovat požadavky před samotným odesláním na server.

4.1.4 NinjaMock

NinjaMock je webový nástroj pro tvorbu wireframů. Poskytuje prvky uživatelského rozhraní specifické pro platformy iOS a Android.

4.1.5 Apiary

Služba Apiary slouží k vytváření specifikací webových služeb. Pomocí této služby lze velmi snadno vytvořit dokumentaci v jazyce Blueprint. Služba se postará o generaci výsledného dokumentu z jazyka Blueprint a zároveň automaticky vytvoří mock endpointy. Klientskou aplikaci tak lze implementovat dříve, než samotnou webovou službu.

4.1.6 Bitnami

Bitnami je služba poskytující balíčky pro snadné zprovoznění serverových aplikací na lokálním počítači nebo na vlastním serveru. V nabídce jsou instalační balíčky serverových aplikací pro rozšířené operační systémy, ale také například obrazy pro virtuální stroje. Není tak potřeba manuálně stahovat veškeré závislosti. Služba byla použita pro nasazení testovacího ticketovacího systému Redmine.

4.1.7 Android Holo Colors a Action Bar Generator

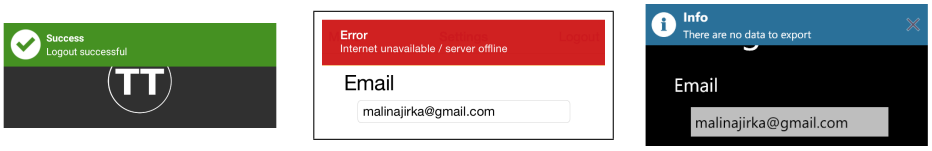
Nástroje Android Holo Colors Generator a Android Action Bar Generator slouží k automatickému vytvoření prvků uživatelského rozhraní pro systém Android v požadovaných barvách. Pro aplikace na Android v5.0 a vyšší již nejsou nástroje zapotřebí, jelikož lze snadno upravovat barvy prvků přímo v XML dokumentu se styly aplikace.

4.1.8 Platforma HERE

Platforma HERE poskytuje REST rozhraní pro převod zeměpisné šířky a délky na adresu, neboli pro tzv. reverzní geocoding.

4.2 Použité knihovny a komponenty

V této sekci jsou zmíněny nejzajímavější externí knihovny a Xamarin komponenty, které byly použity při vývoji aplikace. Pro stažení jednotlivých závis-



Obrázek 4.1: Vzhled toastů

lostí byl použit NuGet Package Manager, který je součástí Microsoft Visual Studio.

4.2.1 Flurl

Flurl je HTTP knihovna poskytující jednotné rozhraní pro všechny tři cílové platformy.

4.2.2 SQLite.NET a SQLiteNetExtensions

Obě tyto komponenty, jak již lze odvodit z názvů, slouží k práci s SQLite databází. Obdobně jako Flurl poskytují jednotný přístup ze všech cílových platforem. SQLiteExtensions je jednoduchá ORM komponenta. Podobně jako u knihovny Hibernate lze pomocí anotací u jednotlivých atributů specifikovat například vztahy mezi entitami. Pomocí této komponenty lze velmi snadno vytvářet databázové dotazy například pro načtení entit včetně potomků.

4.2.3 Xamarin.Forms Toasts plugin

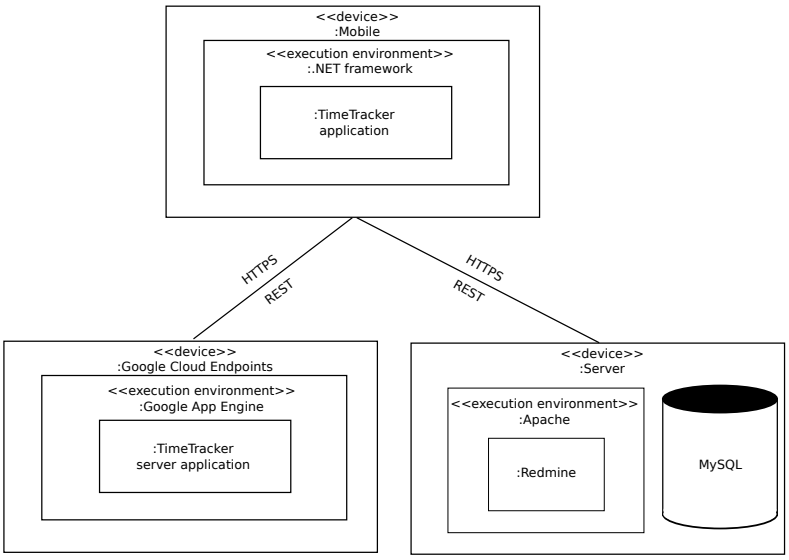
Tento plugin slouží k jednotnému zobrazování tzv. toastů na všech třech platformách. Každá z platforem poskytuje nativní způsob zobrazení krátkých zpráv uživateli. Nicméně například na platformně Android nelze jednoduchým způsobem upravovat barvu těchto zpráv. Mezi uživateli již je zažito zobrazovat chybové zprávy červenou barvou, a naopak zprávy označující úspěch operace barvou zelenou. Na obrázcích 4.1 jsou zobrazeny toasty vytvořené pomocí tohoto pluginu.

4.2.4 PCLStorage

PCLStorage je komponenta, která vývojáře odstiňuje od rozdílů jednotlivých platforem k přístupu k systému souborů. Umožňuje snadno vytvářet, upravovat a mazat soubory a složky na lokálním úložišti zařízení.

4.2.5 TMS FlexCel

TMS FlexCel poskytuje rozhraní v jazyce C# pro tvorbu a úpravu tabulek ve formátu XLS. V aplikaci byla použita zkušební verze knihovny. Plná verze stojí 240 amerických dolarů. Tato knihovna podporuje jen operační systémy



Obrázek 4.2: Diagram nasazení

Android a iOS. Ve Windows Phone verzi aplikace není export do formátu XLS podporován. [36]

4.3 Realizace REST rozhraní

Pro vývoj REST rozhraní byla použita cloudová služba Google Cloud Endpoints. Jedná se o službu poskytující sadu nástrojů a knihoven pro snadný a rychlý vývoj API. Jedním z hlavních důvodů výběru této služby je automatická podpora pro komunikaci přes HTTPS. Poskytovaný certifikát je navíc podepsán Google Internet Authority, která je automaticky uznávána na všech třech cílových operačních systémech.

Na obrázku 4.2 je znázorněno fyzické propojení jednotlivých artefaktů. Jelikož služba Google Cloud SQL není zdarma dostupná, data na TimeTracker Server Application jsou uložena pouze v provizorním nepersistentním úložišti.

4.4 Realizace mobilní aplikace

4.4.1 Příprava vývojového prostředí

Pro vývoj je potřeba mít nainstalované SDK všech cílových platforem. Nicméně Xamarin poskytuje instalační balíček, který automaticky stáhne všechny potřebné závislosti pro operační systémy iOS a Android. Pro operační systém Windows Phone je nutno potřebné závislosti a vývojové prostředí stáhnout manuálně.

Pro operační systém Android lze ihned po dokončení instalace začít vyvíjet. Pouze pro nasazení na reálné mobilní zařízení je potřeba povolit ladění aplikací v nastavení telefonu. Pro publikování aplikace na Google Play je třeba vytvořit certifikát a vlastnit vývojářský účet. Tento účet si lze snadno aktivovat za jednorázový poplatek 25 amerických dolarů. [37]

Pro operační systém Windows Phone je nutné manuálně nainstalovat SDK a Microsoft Visual Studio. Pro vývoj na reálném zařízení je nutné dané zařízení registrovat. Pro registraci slouží nástroj Windows Phone Developer Registration Tool, který je součástí SDK. Registrace vyžaduje Microsoft účet. Pro registraci více než jednoho zařízení nebo vydání aplikace na Store, je třeba aktivovat vývojářský účet. Tento účet je zpoplatněn jednorázovým poplatkem ve výši 19-ti amerických dolarů. [38]

Pro operační systém iOS je příprava vývojového prostředí nejkomplikovanější. Pro vývoj je třeba aplikace sestavovat na Mac OS X verze 10.9.4 nebo vyšší. Navíc je třeba nainstalovat mimo vývojové prostředí Xamarin Studio ještě XCode. Pro vývoj na reálném zařízení nebo publikování aplikací na App Store je třeba vlastnit vývojářský účet a vytvořit vlastní certifikát podepsaný společností Apple. Roční předplatné vývojářského účtu stojí 99 amerických dolarů. [39]

4.4.2 Implementace dlouhotrvajících operací

U mobilních aplikací je nutné, aby dlouhotrvající operace byly prováděny asynchronně a nebylo blokováno UI vlákno. Mezi dlouhotrvající operace patří práce s databází, práce se sítí atp. Framework .NET poskytuje několik přístupů, jak provádět asynchronní operace.

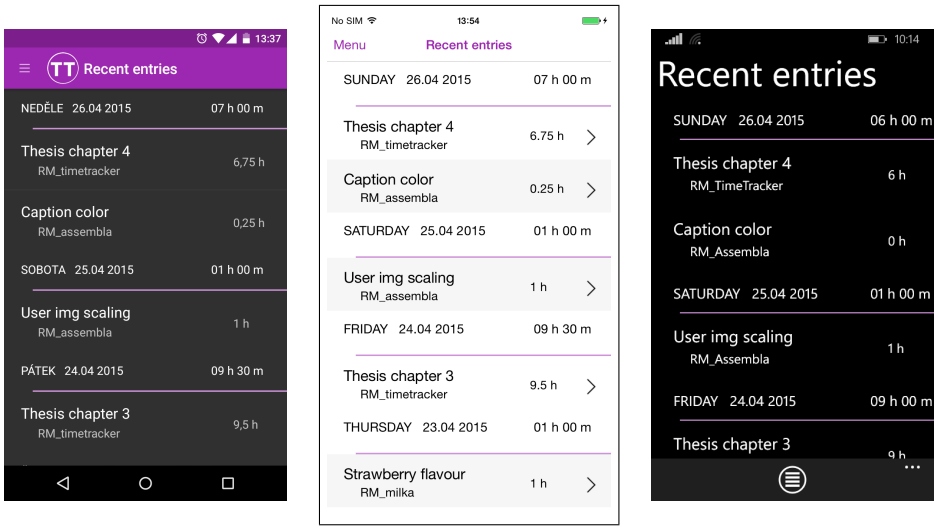
Naivní způsob je manuálně vytvářet vlákna pomocí operátoru *new*. Tento způsob se u novodobých aplikací již moc nepoužívá. Dalším způsobem je použít třídu BackgroundWorker, která provede operaci ve vlákně z thread poolu a po dokončení informuje volající vlákno pomocí callbacku. Posledním způsobem je použít operátory *asnyc* a *await*.

Při implementaci aplikace byla použita jak třída BackgroundWorker, tak operátory *asnyc* a *await*. Třída BackgroundWorker byla použita v případech, kde byly volány metody knihovny, které nevracely odvozenou třídu od třídy Task.

V aplikaci jsou veškeré dlouhotrvající operace prováděny asynchronně. Ve všech případech, kdy uživatel čeká na výsledek operace, jsou zobrazeny nativní indikátory aktivity.

4.4.3 Rozdíly v chování a vzhledu na cílových platformách

Vzhled i chování aplikace je přizpůsobeno každé cílové platformě. Jedinou výjimkou je zobrazení tzv. splash obrazovky na platformně Android. Tato obrazovka zpravidla slouží k indikaci načítání aplikace. U běžných aplikací na



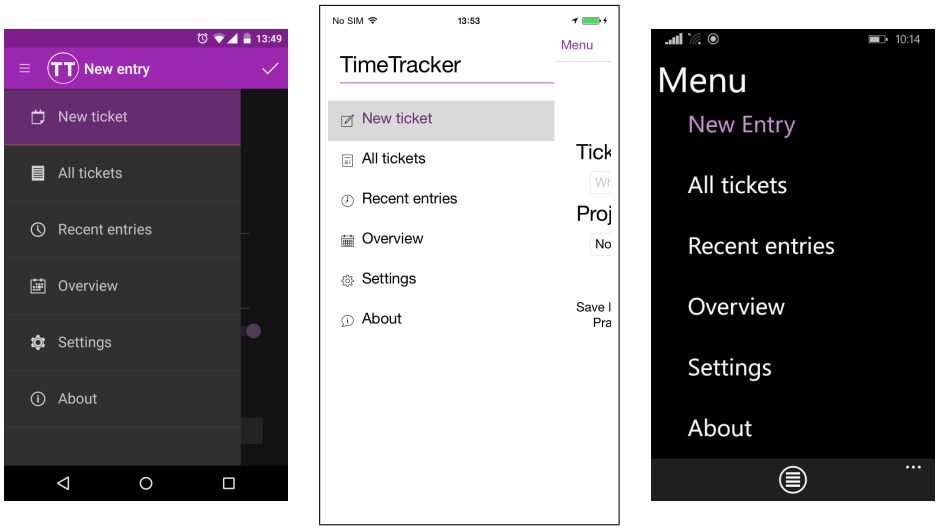
Obrázek 4.3: Snímky obrazovky „Nedávné záznamy“

platformně Android není zobrazení této obrazovky běžné, jelikož načtení trvá jen okamžik. Nicméně doba inicializace prostředí Mono trvá několik sekund, a z toho důvodu je vhodné splash zobrazit.

Jak bylo zmíněno ve druhé kapitole, nástroj Xamarin.Forms umožňuje vytvořit dva typy projektů. Jedná se o PCL projekt a Shared projekt. PCL projekt je vhodný především pro vývoj knihoven. Jelikož kód této aplikace není navrhnut pro tento účel, je využíván druhý způsob uspořádání projektu. Pro oddělení kódu specifického pro konkretní platformu jsou využívány direktivy kompilátoru a metoda *OnPlatform* třídy Device.

Naprostá většina kódu, vyjma kódu některých knihoven, je sdílena všemi třemi platformami. Přesto lze na první pohled pozorovat odlišný vzhled a odlišné chování na každé platformně.

Na obrázku 4.3 jsou snímky obrazovek všech třech platforem se seznamem nedávno přidaných záznamů . V levé části obrázku je zobrazen snímek z platformy Android. V prostřední části obrázku je zobrazen snímek z platformy iOS. V pravé části je zobrazen snímek z platformy Windows Phone. Toto uspořádání je zachováno i u všech zbylých snímku v této kapitole. Je patrné, že na každé platformně má aplikace rozdílný barevný motiv. Zachována je pouze tzv. primární barva aplikace, která je v tomto případě fialová. Jak je pro platformu Android zvykem, v horní části je zobrazen tzv. ActionBar s ikonou aplikace a názvem dané obrazovky, na které se uživatel nachází. Na platformně iOS se lišta v horní části obrazovky nazývá Navigation Bar. Rozdíl je v podstatě pouze v chybějící ikoně aplikace a textovým popisem menu. Na platformně Windows Phone není zvykem používat podobný prvek v horní části obrazovky. Z pravidla se v horní části nachází pouze název dané obrazovky. Nicméně pro zobrazení akcí slouží tzv. Toolbar, který se nachází naopak ve



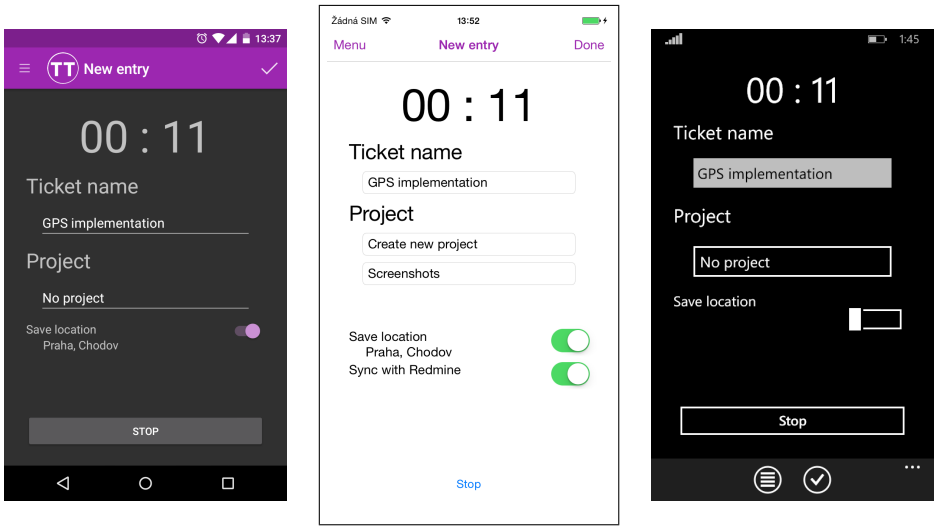
Obrázek 4.4: Snímky menu

spodní části obrazovky.

Na obrázku 4.4 je zobrazeno hlavní menu aplikace. Na platformách Android a iOS lze menu zobrazit dvěma způsoby. Prvním způsobem je kliknutí na ikonku v případě platformy Android a na položku Menu v případě platformy iOS v levém horním rohu obrazovky. Druhým způsobem je vytažení menu pomocí gesta. Na platformně iOS lze menu vytáhnout pohybem prstu zleva doprava ve kterékoli části obrazovky. Nicméně na platformně Android je nutno použít tzv. OnEdge gesto. Pohyb je nutné zahájit na levé hraně displeje. Na platformně Windows Phone není zvykem zobrazovat menu pomocí gesta. Menu lze zobrazit kliknutím na ikonu ve spodní části obrazovky. Za povšímnutí také stojí, že na platformně iOS menu odsune celou horní lištu, zatímco na platfromně Android je menu zobrazeno pod ní.

Na obrázku 4.5 je zobrazena obrazovka sloužící k měření času a vytváření záznamů. Jak je ze snímků patrné, tlačítka, editovatelná textová pole i další komponenty uživatelského rozhraní mají nativní vzhled.

Výsledná aplikace se na jednotlivých platformách liší ještě v mnoha dalších komponentách uživatelského rozhraní. Za zmínku stojí například nativní zobrazení komponenty pro výběr data. Nebo také nativní chování a vzhled při vnořování se v rámci navigace aplikací. U vnořených obrazovek je na platformně Android v levém horním rohu místo ikony menu zobrazena šipka pro návrat o úroveň výš. U platformy iOS je na stejném místě název předchozí obrazovky a pro návrat zpět lze použít stejné gesto, které slouží k zobrazení menu. Rozdílné je také chování a vzhled kontextové akce v seznamech položek. Na platformně iOS lze kontextovou akci zobrazit gestem zprava doleva. Na zbylých dvou platformách lze akci zobrazit delším podržením položky seznamu.



Obrázek 4.5: Snímky obrazovky „Nový záznam“

Celkově chování i vzhled aplikací působí nativním dojmem a běžný uživatel v podstatě nemá šanci poznat rozdíl oproti nativní aplikaci. Je nutné ještě poznamenat, že u některých komponent nelze jednoduše přizpůsobit komponentu barevnému schématu aplikace. Je nutné barevné schéma upravit s využitím SDK konkrétní platformy. Úprava těchto komponent je pak časově relativně náročná na vývoj.

4.5 Testování

V průběhu celého vývoje byla aplikace testována na emulátorech a zařízeních na všech třech platformách. Výhodou testování na emulátorech je výrazně rychlejší nahrávání nových verzí aplikace. Proto je vhodné použít emulátory pro prvotní ladění chování a vzhledu. Nicméně následně je nutno aplikaci otestovat i na reálných zařízeních, jelikož některé funkce nejsou na emulátoru dostupné. Aplikace byla testována na zařízeních různých velikostí displejů a různých verzí operačních systémů.

4.5.1 Emulátory

Pro platformu Windows Phone byl použit emulátor simulující zařízení s 4 palcovým displejem v rozlišení WVGA, 512 MB RAM a verzí operačního systému 8.1. Pro platformu Android byl použit emulátor simulující zařízení se stejnými parametry displeje a verzemi operačního systému 4.1,4.3 a 4.4. Nicméně zařízení mělo pouze 384 MB RAM. Pro platformu iOS byl použit emulátor simulující zařízení iPhone 5S s verzí operačního systému 8.1. Toto

zařízení je vybaveno 4 palcovým displejem s rozlišením 1136x640 pixelů a 1 GB RAM.

4.5.2 Zařízení

Na platformě Windows Phone byla aplikace testována na zařízení Nokia Lumia 520 s verzí operačního systému 8.1. Na platformně Android byla pro testování použita zařízení Nexus 4 s verzí operačního systému 5.1 a Nexus 7 s verzí operačního systému 4.4. Na platformně iOS bylo pro testování použito zařízení iPhone 6 s verzí operačního systému 8.3.

4.5.3 Nástroj Monkey

Na platformně Android byla aplikace testována nástrojem Monkey. Tento nástroj umožňuje automaticky generovat pseudonáhodné interakce uživatele. Hlavním cílem nástroje je odhalit kritické chyby aplikace. Nástroj vývojáře informuje o neočekávaných pádech aplikace a o situacích, kdy aplikace delší dobu nereaguje. V záznamu z testování lze snadno dohledat sekvenci akcí uživatele, které vedly k chybě. Aby se testování aplikace nezaseklo na přihlašovací obrazovce, bylo pro účely testování deaktivováno tlačítko pro odhlášení z aplikace. Aplikace bez chyby opakovaně prošla testy o sekvenci 20 000 pseudonáhodných akcí uživatele.

4.5.4 Uživatelské testování

Pro ověření použitelnosti a intuitivnosti je nutné aplikaci nechat otestovat reálnými uživateli. Jelikož je aplikace cílena na vývojáře, pro testování byli vybráni uživatelé právě z této skupiny. Aplikace na každé z cílových platforem byla testována minimálně dvěma uživateli. V průběhu testování byl kladen důraz na nulový kontakt testera a pozorovatele. Scénář a průběh testování lze nalézt v přílohách C a D.

4.6 Shrnutí

Příprava na vývoj pro tři cílové platformy obnáší instalaci mnoha nástrojů. Nástroj Xamarin poskytuje instalační balíček a návody, které tento proces znatelně zjednoduší. Velkou výhodou nástroje je možnost vyvíjet na všechny tři platformy v rámci jednoho vývojového prostředí. Nicméně v případě nutnosti lze díky vzájemné kompatibilitě projektů bez problému pro vývoj použít více vývojových prostředí.

V průběhu vývoje nastalo několik problémů, nicméně po diskuzích na komunitním fóru se většinu z nich podařilo překonat. Několik chyb bylo přímo v nástroji Xamarin. Některé z těchto chyb se podařilo odstranit po poradě

s oficiální podporou. Zbylé chyby byly nahlášeny do oficiálního ticketovacího systému. Nové verze s opravou chyb vycházejí každých několik dnů.

V průběhu uživatelského testování byly odhaleny pouze drobné nedostatky. Všechny tyto nedostatky byly vyřešeny. Výsledná aplikace je plynulá a všichni testeři se shodli, že aplikace působí nativním dojmem.

Závěr

V této diplomové práci byly srovnány nejpoužívanější nástroje pro multiplatformní vývoj mobilních aplikací. Výsledky srovnání slouží převážně pro usnadnění výběru vhodného nástroje. Do srovnání nebyly zahrnuty nástroje zaměřené na vývoj her.

Ve srovnání nejlépe dopadly nástroje Apache Cordova, Appcelerator a Xamarin. Aplikace v nástroji Apache Cordova lze vyvíjet pomocí běžně používaných webových technologií. Nástroj je vhodný pro finančně nenáročný a rychlý vývoj hybridních aplikací. Nástroje Appcelerator a Xamarin jsou zaměřeny na vývoj nativních aplikací. Po podrobném srovnání převážně z hlediska výkonu, dostupných funkcí a komfortnosti vývoje byl jako nejvhodnější nástroj vybrán Xamarin.

Pro ověření vhodnosti výběru byla v nástroji Xamarin implementována aplikace pro měření a reportování odpracovaného času. Aplikace splňuje veškeré požadavky, které na ní byly v zadání práce kladené. Naprostá většina kódu je sdílena mezi jednotlivými platformami. Z hlediska výkonu a vzhledu je aplikace téměř k nerozpoznání od aplikací vyvinutých v nativních nástrojích pro vývoj na dané operační systémy. Vývoj aplikace byl relativně časově náročný, nicméně to bylo způsobeno především seznamováním se s jazykem C# a nástrojem Xamarin. V průběhu vývoje bylo odhaleno několik interních chyb nástroje. Nicméně společnost Xamarin pravidelně vydává nové verze s opravami nahlášených chyb. Rozsáhlá komunita, aktivní podpora a kvalitní dokumentace spolu s vlastnostmi jazyka C# činí z nástroje Xamarin vhodnou platformu pro komerčním vývoj mobilních aplikací.

Relativně novým trendem jsou tzv. chytré hodinky. V nástroji Xamarin lze vyvíjet aplikace jak pro hodinky na platformně Android, tak na platformně iOS. Možným rozšířením této práce by bylo otestovat tento nástroj při rozšíření výsledné aplikace o podporu běhu na chytrých hodinkách.

Literatura

[1] An overview of IBM MobileFirst Platform. Technická Zpráva WSW14181-USEN-09, IBM Corporation 2014, Route 100, Somers, NY 10589, 11 2014.

[2] What Should I Choose? Native or Hybrid Mobile App? 12 2012, cit. 2015.21.01. Dostupné z: http://bizitpi.blogspot.cz/2012/12/what-should-i-choose-native-or-hybrid.html

[3] Compatibility tables for support of HTML5, CSS3, SVG and other technologies in various browsers. 2015, cit. 2015.20.01. Dostupné z: http://caniuse.com/

[4] PhoneGap and the Apple store. Cit. 2015.20.01. Dostupné z: http://www.awesome-robot.com/article/PhoneGap_and_the_Apple_Store/

[5] Android and iOS Squeeze the Competition. 1 2015, cit. 2015.25.01. Dostupné z: www.idc.com/getdoc.jsp?containerId=prUS25450615

[6] Official MoSync website. Cit. 2015.27.01. Dostupné z: http://www.mosync.com/

[7] Official RhoMobile Suite website. Cit. 2015.29.01. Dostupné z: http://rhomobile.com/

[8] Official Codename One website. Cit. 2015.30.01. Dostupné z: http://www.codenameone.com/

[9] Official Appcelerator website. Cit. 2015.02.02. Dostupné z: http://www.appcelerator.com/

[10] Official Apache Cordova website. Cit. 2015.03.02. Dostupné z: https://cordova.apache.org/

[11] Official Xamarin website. Cit. 2015.04.02. Dostupné z: http://xamarin.com

[12] Official Alpha Anywhere website. Cit. 2015.08.02. Dostupné z: http://www.alphasoftware.com/products.asp

[13] Official Trigger.IO website. Cit. 2015.09.02. Dostupné z: http://http://trigger.io/

[14] Official Neomade website. Cit. 2015.12.02. Dostupné z: http://neomades.com/

[15] IBM MobileFirst SDK. Cit. 2015.12.02. Dostupné z: https://components.xamarin.com/view/ibm-worklight?version=6.3.0.1

[16] IBM MobileFirst Platform compared to do-it-yourself. Technická Zpráva WSW14226-USEN-01, IBM Corporation 2014, Route 100, Somers, NY 10589, 10 2014.

[17] LEEUWEN, D. V.: Getting Started with Kapsel. 2015, cit. 2015.12.02. Dostupné z: http://scn.sap.com/docs/DOC-49592

[18] Official Telerik website. Cit. 2015.13.02. Dostupné z: http://www.telerik.com/

[19] Official Sencha website. Cit. 2015.20.02. Dostupné z: http://www.sencha.com/

[20] PhoneGap Cordova and whats in a name. 3 2012, cit. 2015.21.02. Dostupné z: http://phonegap.com/2012/03/19/phonegap-cordova-and-whats-in-a-name/

[21] Official PhoneGap website. Cit. 2015.21.02. Dostupné z: http://phonegap.com/

[22] Titanium platform documentation. Cit. 2015.28.02. Dostupné z: http://docs.appcelerator.com/titanium/3.0/#\protect\relax\kern-.16667em/guide/Titanium_Platform_Overview

[23] Appcelerator: Titanium Architecture. 2013, cit. 2015.28.02. Dostupné z: http://docs.appcelerator.com/titanium/3.0/images/download/attachments/29004883/sdk.png

[24] POWELL, J.: What Titanium Appcelerator REALLY is and How it Works! 7 2012, cit. 2015.01.03. Dostupné z: http://forumone.com/insights/what-titanium-appcelerator-really-and-how-it-works/

[25] WHINNERY, K.: Comparing Titanium and PhoneGap. 5 2012, cit. 2015.01.03. Dostupné z: http://forumone.com/insights/what-titanium-appcelerator-really-and-how-it-works/

[26] Official Aptana website. Cit. 2015.04.03. Dostupné z: http://www.aptana.com/

[27] Apache Cordova documentation v4.0.0. Cit. 2015.08.03. Dostupné z: http://cordova.apache.org/docs/en/4.0.0/guide_overview_index.md.html#Overview

[28] Ian HICKSON, G. I.: Web SQL Database. 9 2010, cit. 2015.08.03. Dostupné z: http://dev.w3.org/html5/webdatabase/

[29] Web SQL Database browser support. Cit. 2015.08.03. Dostupné z: http://caniuse.com/#search=WebSQL

[30] Indexeddb browser support. Cit. 2015.08.03. Dostupné z: http://caniuse.com/#search=indexeddb

[31] de ICAZA, M.: Announcing Xamarin. 5 2011, cit. 2015.09.03. Dostupné z: http://tirania.org/blog/archive/2011/May-16.html

[32] Xamarin Crunchbase profile. Cit. 2015.08.03. Dostupné z: https://www.crunchbase.com/organization/xamarin

[33] Official Xamarin documentation. Cit. 2015.09.03. Dostupné z: http://developer.xamarin.com/

[34] About Mono. Cit. 2015.10.03. Dostupné z: http://www.mono-project.com/docs/about-mono/

[35] GooglePlay com.nordicusability.jiffy profile. Cit. 2015.15.04. Dostupné z: https://play.google.com/store/apps/details?id=com.nordicusability.jiffy&hl=cs

[36] TMS FlexCel Studio for .NET. Cit. 2015.16.04. Dostupné z: http://components.xamarin.com/view/flexcel

[37] Google Developer Console Help. Cit. 2015.16.04. Dostupné z: https://support.google.com/googleplay/android-developer/answer/6112435?hl=en&rd=1

[38] How to register your phone for development for Windows Phone 8. Cit. 2015.16.04. Dostupné z: https://msdn.microsoft.com/en-us/library/windows/apps/ff769508(v=vs.105).aspx

[39] iOS Developer Program official website. Cit. 2015.16.04. Dostupné z: https://developer.apple.com/programs/ios/

Příloha A

Seznam použitých zkratek

AES Advanced Encryption Standard

AOT Ahead Of Time

API Application Programming Interface

CLI Command Line Interface

CSS Cascading Style Sheets

CSV Comma-Separated Values

DLL Dynamic-Link Library

DVD Digital Versatile Disc

GPL General Public License

GPS Global Positioning System

GUID Global Unique Identifier

HTML HyperText Markup Language

HTTPS HyperText Transfer Protocol Secure

IDC International Data Corporation

IDE Integrated Development Environment

IL Intermediate Language

JS JavaScript

JSON JavaScript Object Notation

MVC Model-View-Controller

NFC Near Field Communication

ORM Object-Relational Mapping

OS Operating System

PC Personal Computer

PCL Portable Class Library

Q&A Questions and Answers

RAM Random Access Memory

REST Representational State Transfer

SDK Software Development Kit

SMS Short Message Service

SQL Structured Query Language

SSO Single Sign-On

UI User Interface

URL Uniform Resource Locator

WVGA Wide Video Graphics Array

WYSIWYG What You See Is What You Get

XAML Extensible Application Markup Language

XML Extensible Markup Language

Příloha **B**

Obsah přiloženého DVD

- `readme.txt` ..... stručný popis obsahu DVD
- `APIdocs.txt` ..... dokumentace REST API
- `src`
  - `implMobile` ..... zdrojové kódy mobilní aplikace
  - `implServer` ..... zdrojové kódy serverové aplikace
  - `latex` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF

Příloha C

Instrukce k uživatelskému testování

C.1 Popis aplikace

Aplikace slouží k jednoduchému a rychlému měření a reportování odpracovaných hodin.

C.2 Uživatelé

Uživatelé cílové aplikace budou převážně vývojáři pracující z domova.

C.3 Doplňující informace

Cílem tohoto testování je odhalit neintuitivní chování a vzhled aplikace. Z toho důvodu jsou testovací scénáře popsány velmi stručně. Pokuste se bez cizí pomoci postupně splnit všechny úkoly. Před provedením každého úkolu si scénář nejprve celý přečtěte.

C.4 Testovací scénáře

C.4.1 Scénář 1

1. Vytvořte nový uživatelský účet a přihlaste se.

2. Odreportujte alespoň 10 sekund práce. Název ticketu a projektu si zvolte libovolně. Součástí reportu by měla být vaše aktuální lokace.

C.4.2 Scénář 2

1. Importujte data z nástroje Redmine. URL nástroje Redmine je „http://192.168.194.1/redmine“. Přihlašovací jméno je „malinjir“ a heslo je „12345“.

2. Pokuste se zjistit, kolik hodin bylo odreportováno v pátek 24.4.2015.

C.4.3 Scénář 3

1. Vytvořte nový záznam pro export do nástroje Redmine. Jako název ticketu nastavte „Export“ a název projektu „RM export“.

2. Pokuste se vytvořená data exportovat do nástroje Redmine.

3. Ověřte na PC ve webovém prohlížeči na adrese „http://127.0.0.1/redmine/projects“, že se projekt podařilo vytvořit.

C.4.4 Scénář 4

Odešlete přehled odreportovaných hodin mezi daty 24.4.2015 – 27.4.2015 na Vámi zvolenou e-mailovou adresu ve formátu csv.

C.4.5 Scénář 5

1. Odreportujte 5s k ticketu s názvem „User img scaling“.

2. Smažte právě vytvořený záznam.

C.4.6 Scénář 6

Odhašte se z aplikace.

C.5 Otázky

Měl jste pocit, že je aplikace přizpůsobena Vašemu operačnímu systému?

Příloha D

Průběh uživatelských testů

Všichni testeři jsou technicky zdatní a mají zkušenosti s testovaným operačním systémem.

D.1 Uživatel 1 – iOS

Uživatel při registraci zadával přihlašovací jméno místo e-mailové adresy. Nicméně po zobrazení chybové hlášky z informací, že je e-mailová adresa neplatná, již s prvním scénářem neměl problémy. U druhého scénáře uživatel delší dobu váhal, pod jakou položkou menu se nachází export do nástroje Redmine. Nicméně vylučovací metodou zvolil správnou položku. Zbylé scénáře uživateli nečinily žádné problémy.

Měl jste pocit, že je aplikace přizpůsobena Vašemu operačnímu systému?

- „Ano“

Změny v aplikaci provedené na základě výsledků testu.

- Popisek „Enter login name“ přejmenován na „Enter e-mail“.

D.2 Uživatel 2 – iOS

Uživatel měl problém hned s prvním scénářem. Při stisku tlačítka s nápisem registrovat očekával přechod na jinou obrazovku. Nicméně tlačítko slouží k odeslání přihlašovacích údajů na server. Uživatel také očekával, že při registraci bude vyžadováno potvrzení hesla. Po chvíli se uživatel zorientoval a úspěšně dokončil první scénář. Druhým a třetím scénářem uživatel prošel bez zaváhání. U dalšího scénáře měl uživatel drobné problémy s nativní komponentou pro výběr data. Nicméně nakonec se mu podařilo zvolit zadané rozmezí a data odeslat e-mailem. Zbylé scénáře uživateli nečinily problémy.

Měl jste pocit, že je aplikace přizpůsobena Vašemu operačnímu systému?

- „Jsem relativně nový uživatel iOS, takže nemám zkušenosti z mnoha aplikacemi. Nicméně celkově na mě aplikace působila velmi dobře.“

Změny v aplikaci provedené na základě výsledků testu.

- Do registračního formuláře bylo přidáno ověřování hesla. Uživatel tak lépe pozná, že zobrazený formulář slouží k registraci.

D.3 Uživatel 3 – Android

Prvním scénářem uživatel prošel bez problému. Při importování dat z nástroje Redmine měl uživatel problémy s internetovým připojením. Nicméně nejednalo se o problém v aplikaci. Při zjišťování odreportovaných hodin ke konkrétnímu datu zvolil uživatel netradiční přístup. Místo toho, aby našel datum v nedávných záznamech, uživatel informaci hledal v sekci „Overview“. Nicméně došel ke správnému výsledku a později si uvědomil, že informaci šlo získat kratší cestou. Scénáře 3 a 4 nečinily uživateli žádné problémy. U scénáře 5 uživatel váhal, jak smazat záznam. Nicméně po chvíli se mu podařilo záznam smazat. Poslední scénář uživateli nečinil žádné problémy.

Měl jste pocit, že je aplikace přizpůsobena Vašemu operačnímu systému?

- „Ano“

Změny v aplikaci provedené na základě výsledků testu.

- Žádné

D.4 Uživatel 4 – Android

První scénář uživateli nečinil žádné problémy. U druhého scénáře uživatel zadal špatné heslo, ale nebyl spokojen s délkou chybové hlášky. Třetí a čtvrtý scénář uživatel dokončil bez problému. U pátého scénáře uživatel nemohl najít ticket se zadaným názvem. Nicméně po chvilce zjistil, že ho celou dobu pouze přehlížel. Zbylé scénáře uživatel prošel bez problému.

Měl jste pocit, že je aplikace přizpůsobena Vašemu operačnímu systému?

- „Ano“

Změny v aplikaci provedené na základě výsledků testu.

- Všechny texty chybových hlášek byly zkráceny.

D.5 Uživatel 5 – Windows Phone

Uživatel prošel všemi scénáři bez problému.

Měl jste pocit, že je aplikace přizpůsobena Vašemu operačnímu systému?

- „Ještě jsem se nesetkal s podobným menu, ale působilo na mě dobře. “

Změny v aplikaci provedené na základě výsledků testu.

- Žádné

D.6 Uživatel 6 – Windows Phone

Registrace proběhla bez problému. Nicméně pak se uživatel rozhodl, že se nejprve seznámí s celou aplikací. Na obrazovce s přehledem uživatel očekával, že pod nadpisem „Projects“ se později načte seznam projektů. Nicméně ten se ani po čekání nenačetl. Se scénáři neměl uživatel žádný problém.

Měl jste pocit, že je aplikace přizpůsobena Vašemu operačnímu systému?

- „Ano“

Změny v aplikaci provedené na základě výsledků testu.

- V případě, že je seznam projektů na obrazovce „Overview“ prázdný, místo seznamu se zobrazí příslušná informace.