VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

VÝVOJÁŘSKÉ TECHNOLOGIE NOSITELNÝCH ZAŘÍZENÍ PRO YSOFT SAFEQ

WEARABLES DEVELOPMENT TECHNOLOGIES FOR YSOFT SAFEQ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JAN STÁREK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. FILIP ORSÁG, Ph.D.**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/19425/2016/xstare03

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Stárek Jan**
Obor: Informační technologie
Téma: **Vývojářské technologie nositelných zařízení pro YSoft SafeQ**
Wearables Development Technologies for YSoft SafeQ
Kategorie: Uživatelská rozhraní

Pokyny:

1. Seznamte se s produkty v oblasti nositelných zařízení, s možnostmi jejich interakce s okolními zařízeními a systémy, a se systémem SafeQ firmy YSoft.
2. Zpracujte přehled a porovnání dostupných zařízení a vývojářských technologií pro programování nositelných zařízení (např. Xamarin, XCode, Tizen). Zaměřte se na dostupnost možností internacionalizace aplikace, přístupu k NFC, Eddystone nebo iBeacon, možnostmi zabezpečení komunikace a výkon.
3. Vyberte si zařízení a vytvořte aplikaci, která bude komunikovat se serverem správy tisku YSoft SafeQ prostřednictvím rozhraní REST.
4. Otestujte funkčnost implementované aplikace a zhodnoťte přínos řešení v porovnání se stávajícími technologiemi a postupy.

Literatura:

- JAYGARL, Hojun. *Professional Tizen application development*. Chichester, England: Wiley, 2014. ISBN 978-1-118-80925-9
- HERMES, Dan. *Xamarin Mobile Application Development: Cross-Platform C# and Xamarin.Forms Fundamentals*. California: Apress, 2015. ISBN 9781484202159.
- Dokumentace YSoft SafeQ

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2 zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Orság Filip, Ing., Ph.D.**, UITS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Nositelná zařízení se řadí mezi stále oblíbenější elektroniku. Tento fakt přispívá ke zvyšující se snaze rozšířit mobilní i jiné aplikace na nositelná zařízení. Na tyto skutečnosti navazuje tato práce, která shrnuje informace o nositelných zařízeních a jejich typických operačních systémech. V dalších kapitolách se nacházejí informace o možnostech pro vývojáře, které tyto operační systémy a jejich vývojářské nástroje nabízejí. Na základě získaných poznatků se v práci nachází i demonstrace tvorby aplikace, konkrétně pro obsluhu tiskáren skrze systém YSoft SafeQ společnosti Y Soft Corporation.

Abstract

Wearable devices grew in popularity in recent years. This fact contributes to increasing efforts to expand mobile and other applications to wearable devices. Exploring these possibilities, this thesis summarizes information on wearable devices and their typical operating systems, including available tools and restrains these systems offer. Based on the conclusion of theoretical part of this thesis an application for wearable devices is designed and implemented, specifically an application for printers that operate via SafeQ system developed in Y Soft Corporation.

Klíčová slova

nositelná zařízení, nositelnosti, vývojářské technologie, chytré hodinky, watchOS, android wear, tizen, Y Soft, YSoft SafeQ

Keywords

wearable devices, wearables, development technology, smart watch, watchOS, android wear, tizen, Y Soft, YSoft SafeQ

Citace

STÁREK, Jan. *Vývojářské technologie nositelných zařízení pro YSoft SafeQ*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Orság Filip.

Vývojářské technologie nositelných zařízení pro YSoft SafeQ

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Filipa Orsága, Ph.D.. Další informace mi poskytl pan Mgr. Juraj Michálek ze společnosti Y Soft Corporation. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jan Stárek
4. května 2017

Poděkování

Tímto bych chtěl poděkovat vedoucímu bakalářské práce, Ing. Filip Orság, Ph.D., za odbornou pomoc, rady, konzultace a připomínky při jejím psaní. Na tomto místě bych chtěl poděkovat také panu Mgr. Juraji Michálkovi za odbornou konzultaci ohledně textu bakalářské práce a výsledné aplikace.

Obsah

1 Úvod **4**

2 Nositelná elektronika **5**
2.1 Co je to nositelná elektronika . . . 5
2.2 Typy nositelností . . . 5
2.2.1 Zdravotní pomůcky . . . 6
2.2.2 Chytré oblečení . . . 7
2.2.3 Chytré hodinky . . . 7
2.2.4 Zařízení pro sledování tělesné kondice . . . 8
2.2.5 Brýle pro virtuální realitu . . . 8
2.2.6 Brýle pro rozšířenou realitu . . . 9

3 Chytré hodinky **10**
3.1 Historie . . . 10
3.2 Obvyklé komunikační kanály . . . 10
3.2.1 NFC . . . 10
3.2.2 Bluetooth . . . 12
3.2.3 Wi-Fi . . . 14
3.3 Jazyky čtené zprava doleva . . . 15
3.4 Technické parametry a výkon . . . 16
3.5 Shrnutí . . . 17

4 Operační systémy chytrých hodinek **18**
4.1 Android Wear . . . 18
4.1.1 Interakce telefonu s hodinkami a distribuce balíčků do hodinek . . . 19
4.1.2 Komunikace mezi mobilním telefonem a chytrými hodinkami . . . 19
4.1.3 Přístup k NFC a Bluetooth majákům . . . 20
4.1.4 Internacionalizace . . . 20
4.2 watchOS . . . 22
4.2.1 Rozdíly ve verzích watchOS . . . 22
4.2.2 Distribuce balíčku do hodinek . . . 23
4.2.3 Komunikace mezi mobilním telefonem a chytrými hodinkami . . . 23
4.2.4 Internacionalizace . . . 24
4.2.5 Přístup k NFC a Bluetooth majákům . . . 25
4.3 Tizen . . . 25
4.3.1 Historie platformy Tizen . . . 25
4.3.2 Distribuce aplikací do hodinek se systémem Tizen . . . 26
4.3.3 Komunikace mezi mobilním telefonem a chytrými hodinkami . . . 26

4.3.4 Komunikační rozhraní chytrých hodinek . . . 27
4.3.5 Internacionalizace . . . 27
4.4 Shrnutí a porovnání operačních systémů . . . 27

5 Vývojářské technologie chytrých hodinek 29
5.1 Nativní vývoj Android aplikací . . . 29
5.1.1 Android Studio . . . 29
5.1.2 Operační systém Android z pohledu vývojáře . . . 30
5.1.3 Android Studio a vývoj aplikací pro Android Wear . . . 31
5.2 Nativní vývoj Apple aplikací . . . 31
5.2.1 Interface Builder . . . 32
5.2.2 Swift . . . 32
5.2.3 Xcode simulátory . . . 32
5.3 Xamarin . . . 33
5.3.1 Historie . . . 33
5.3.2 Komponenty . . . 33
5.3.3 Mono . . . 34
5.3.4 Vývojová prostředí . . . 35
5.3.5 Xamarin a nositelná zařízení . . . 37
5.4 Nativní vývoj Tizen aplikací . . . 37
5.4.1 Prerekvizity a systémové požadavky . . . 38
5.4.2 Tizen Studio . . . 38
5.4.3 Typy Tizen aplikací . . . 38
5.4.4 Vývoj nativních aplikací . . . 39
5.4.5 Vývoj aplikací založených na webových technologiích . . . 40
5.5 Shrnutí a porovnání vývojářských technologií . . . 41

6 Implementace terminálové aplikace YSoft SafeQ 43
6.1 YSoft SafeQ . . . 43
6.1.1 Identifikace tiskárny a příslušného serveru . . . 43
6.1.2 YSoft SafeQ server . . . 44
6.2 Použité nástroje a vývojové prostředí . . . 44
6.3 Princip činnosti výsledné aplikace . . . 45
6.3.1 Vyhledávání tiskáren . . . 45
6.3.2 Přihlášení uživatele ke konkrétnímu terminálu . . . 45
6.3.3 Komunikace mezi chytrými hodinkami a telefonem . . . 46
6.3.4 Komunikace mobilní aplikace se serverem YSoft SafeQ . . . 48
6.4 Vzhled aplikace . . . 49
6.5 Struktura kódu aplikace . . . 49
6.6 Zhodnocení aplikace . . . 50
6.6.1 Doba překladu a testování . . . 50
6.6.2 Navrhované vylepšení terminálové aplikace . . . 50
6.6.3 Porovnání se stávajícími řešeními . . . 51

7 Závěr 52

Literatura 53

Přílohy 57

A Tabulky **58**

B Obrázky **59**

C Manuál k překladu aplikace **61**
C.1 Potřebné prostředí 61
C.2 Stažení NuGet balíčků a překlad zdrojových souborů 61

D Manuál k Android a Android Wear aplikaci **62**
D.1 Potřebná zařízení 62
D.2 Instalace vygenerovaného balíčku 62
D.3 Hledání tiskáren 62

E Obsah nosiče **63**

Kapitola 1

Úvod

V moderní technické době je stále větší poptávka po zařízeních, které dokáží předat informace uživateli rychleji a přirozeněji než jiná stávající elektronická zařízení. Pomocí takzvaných nositelných zařízení mohou uživatelé obvykle velmi snadno získat informace, o které mají zájem. Trh nositelných zařízení je velmi různorodý a dynamicky se mění a přizpůsobuje potřebám uživatelů. Za dobu své existence si nositelnosti získaly oblibu u sportovců, technických nadšenců ale i u lidí, kterým nositelná zařízení pomáhají řešit situace související s jejich zdravotními problémy. Jejich uplatnění můžeme nalézt jak v osobní tak podnikové sféře. Nositelnosti mohou být například zdravotní pomůckou, nástrojem pro sledování fyzických aktivit, zařízením zprostředkujícím virtuální či rozšířenou realitu a pro některé uživatele i módním doplňkem.

Vývojáři operačních systémů pro nositelná zařízení mohou vývojářům třetích stran zpřístupnit jejich rozhraní a umožnit jim vyvíjet aplikace pro jejich operační systémy. Vývojářské technologie nositelných zařízení jsou pak prostředky, jakými mohou vývojáři pro operační systémy nositelných zařízení psát aplikace. Pro vývojáře, kteří chtějí distribuovat svůj produkt na běžná nositelná zařízení, může být obtížné zmonitorovat nabídku trhu a vybrat konkrétní platformy, které budou chtít v rámci své aplikace podpořit.

Práce je cílena především na vývojáře, kteří začínají přemýšlet o vytváření a distribuci aplikací pro nositelná zařízení. Tato bakalářská práce má za cíl předat úvod do všeobecné problematiky nositelných zařízení a dále detailněji popsat konkrétní typ nositelností – chytré hodinky. Čtenář zde může nalézt také popis tří známých operačních systémů, kterými jsou Android Wear, watchOS a Tizen. Dále také detailní popis vývojových prostředku pro vývoj aplikací pro tyto operační systémy.

Kapitola 2

Nositelná elektronika

Nositelná zařízení jsou elektrotechnická zařízení spojená s okolním světem pomocí různých komunikačních technologií. Přináší tak jejich nositeli nové možnosti při práci s informacemi. Mohou být využitelná jak pro sběr informací například o zdravotním stavu nositele, tak i pro prezentaci informací nositeli například pomocí vibrací, LED diody[1], vestavěného displeje nebo reproduktoru. Možností konstrukce, implementace a využití nositelností je nepřeberné množství. V následujících podkapitolách budou shrnuty a prezentovány některé základní informace o nositelných zařízeních.

2.1 Co je to nositelná elektronika

Neexistuje žádná celosvětově uznávaná definice pojmu nositelná elektronika. Pojem nositelné zařízení tedy můžeme označit jako libovolné elektronické zařízení připevnitelné k lidskému tělu nebo oděvu na něm oblečeném [21] [32]. U některé elektroniky je problém určit, zda se jedná o nositelnost. Může to být například u kamery GoPro[2] uchycené na helmě, která je připevněna k lidské hlavě. Pokud chápeme termín nositelná elektronika tak jak jsem jej uvedl výše, pak by se dala tato sestava považovat za nositelnou elektroniku.

Nositelnosti velmi často komunikují s dalšími zařízeními prostřednictvím různých komunikačních kanálů [30]. Samy obvykle nedisponují plnohodnotnými zobrazovacími prvky, které by umožňovaly kvalitní a plnohodnotnou interakci s uživatelem, proto jsou data z nich často zasílána dále například do chytrých telefonů, nebo přímo do internetové sítě na centralizovaná úložiště pro zpracování a zobrazení. Pojem nositelné zařízení úzce souvisí s aktuálně diskutovanými pojmy *Big data*[3] nebo *Internet of Things*[4].

2.2 Typy nositelností

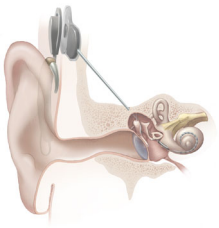
Nositelnosti se často rozlišují dle jejich užití. Některé typy nositelností umožňují snímat základní lidské funkce a tato data seskupovat pro pozdější analýzu [14]. Mohou také upozorňovat svého nositele na nezvyklé jevy, které se v jeho těle dějí. Může to být například vysoký krevní tlak nebo příliš vysoká hladina glukózy v těle. Další typy nositelností pak mohou pomocí kamery, mikrofonu či jiných standardních funkcí chytrých telefonů daleko

[1] https://en.wikipedia.org/wiki/Light-emitting_diode
[2] https://gopro.com/
[3] https://en.wikipedia.org/wiki/Big_data
[4] https://en.wikipedia.org/wiki/Internet_of_things

příměji interagovat s uživatelem [30]. Tato práce se ovšem bude nejblíže zajímat o nositelná zařízení, se kterými může uživatel interagovat a mají tak využití pro YSoft SafeQ. V následujících sekcích budou popsány různé typy nositelností.

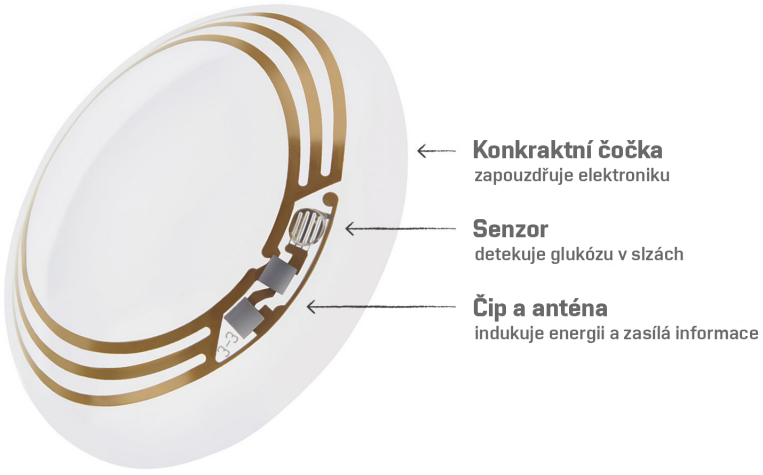
2.2.1 Zdravotní pomůcky

Mezi nositelnosti můžeme zařadit různé zdravotní pomůcky. Obvykle monitorují činnosti lidského těla, případně stimulují či nahrazují některé lidské orgány [14]. Mezi tato zařízení patří například kochleární implantáty [38]. Jedná se o elektrotechnická zařízení nahrazující funkci šneka v lidském uchu. Tato technologie stimuluje nervy ve vnitřním uchu a dovoluje tak některým tělesně postiženým jedincům částečně slyšet.



Obrázek 2.1: Kochleární implantát [49]

Mezi další stimulační zdravotní elektronická zařízení patří například kardiostimulátory. Kardiostimulátor se používá k léčbě poruch srdečního rytmu, kdy je rytmus příliš pomalý. Toto zařízení vydává pulsy, které přímo ovlivňují činnost srdce. Pod zdravotnické pomůcky můžeme pak zařadit i nositelnost od společnosti Google s názvem *Google Contact Lens* [49], které slouží pro detekování hladiny glukózy v lidských slzách. Toho se využívá u lidí trpící cukrovkou, kteří tak mají přehled o hladině glukózy v jejich těle po celou dobu nošení těchto čoček.



Obrázek 2.2: *Google Contact lens* [14]

2.2.2 Chytré oblečení

Informace o chytrém oblečení jsou čerpány z elektronických článků [27] [28], které napsal Michael Sawh. Chytré oblečení je typ oděvu nebo obuvi, která v sobě obsahuje elektronické zařízení pro konkrétní specifický účel. V roce 2014 instituce Gartner predikovala, že v roce 2016 bude chytré oblečení četněji prodáváno než chytré náramky, které budou zmíněny v textu níže. Tato predikce se ovšem nenaplnila, nicméně i tak je chytré oblečení významným zástupcem nositelností a do budoucna má velký potenciál.

Jedním z významných výrobců nositelností pro sportovní užití je i společnost Polar. Tato společnost oznámila produkci sportovního trička[1] se zabudovaným elektronickým zařízením pro sledování různých vlastností jeho nositele. Tento produkt by měl vyjít v březnu roku 2017.

Dalším zástupcem tohoto typu nositelností je produkt Neviano[2] společnosti Spinali Design. Jedná se o dámské plavky, které obsahují senzor pro detekci UV záření. Po spárování s chytrým telefonem s operačním systémem Android nebo iOS tak nositele mohou upozornit na nebezpečí popálení kůže vlivem intenzivního slunečného svitu.

2.2.3 Chytré hodinky

Informace v tomto odstavci jsou čerpány ze zdrojů [3] [24]. Chytré hodinky[3] jsou typ nositelné elektroniky, který nahrazuje běžné mechanické hodinky za hodinky s rozšířenou funkcionalitou. Chytré hodinky jsou často neodlučitelně propojeny s mobilním telefonem, případně využívají jeho operačního systému a slouží jen pro zobrazení některých prvků z mobilního telefonu. Pomocí chytrých hodinek můžeme například jednoduše číst textové zprávy, e-mailové zprávy a využívat mnoho dalších komunikačních kanálů. Chytré hodinky pohání obvykle ARM[4] procesory, které jsou uzpůsobeny nízkému příkonu pro maximální dobu provozu na jedno nabití. Mohou mít buď vlastní proprietární firmware, do kterého obvykle není možné zasahovat, nebo naopak obsahovat otevřený operační systém. Zástupci otevřených systémů jsou například Android Wear, watchOS či Tizen, pro které existují vývojářské nástroje, díky kterým je nám umožněno vyvíjet pro ně aplikace. Tyto operační systémy budou popsány v kapitole č. 4.

Chytré hodinky Moto 360 [45] spadají do kategorie nositelností s operačním systémem Android Wear, tudíž je pro ně možné psát aplikace. Tyto hodinky jsou osazeny výkonným čipem Qualcomm Snapdragon 400, grafickým čipem Adreno 305 a obsahují 512 MB RAM paměti a 4 GB paměti ROM. S chytrým telefonem komunikují skrze Bluetooth 4.0 LE. Se sítí internet pak mohou komunikovat pomocí Wi-Fi. Mezi senzory můžeme nalézt senzor zrychlení, gyroskop a detekce světla z okolí hodinek.

[1] https://www.polar.com/en/b2b_products/team_sports/team_pro
[2] http://www.spinali-design.com/collections/neviano-intelligent-swimsuit-women
[3] http://dictionary.cambridge.org/dictionary/english/smartwatch
[4] https://en.wikipedia.org/wiki/ARM_architecture

Obrázek 2.3: Moto 360 od společnosti Motorola [45]

2.2.4 Zařízení pro sledování tělesné kondice

Zařízení pro sledování tělesné kondice se dají zařadit do dvou významných kategorií. Jedná se o chytré náramky (nebo také *fitness trackers*) a sporttestery (nebo také *sport watches*) [30].

Chytré náramky jsou nositelnosti sloužící k monitorování každodenních fyzických aktivit, jakými jsou například počet ušlých kroků, množství spálených kalorií nebo monitorování spánkové aktivity [30]. Mezi známé zařízení této kategorie patří například Xiaomi Mi Band[1]. Tento náramek umožňuje využít budík implementovaný vibrační jednotkou, sledování spánku či akcelerometr. Především však sdílí upozornění s telefonním zařízením vybaveným operačním systémem Android, a tak uživatele informuje o aktuálním stavu jeho telefonu. Další zajímavou vlastností pak je například automatické odemykání telefonu v blízkosti náramku.

Sporttester je typ nositelnosti, který dokáže sbírat a analyzovat údaje během konkrétní fyzické aktivity [30]. Každá nositelnost je pak specializovaná pro určité typy činností. Existují sporttestery zaměřené na cyklistiku, běh či plavání. Pro různé typy fyzické aktivity pak sporttester nabízí specifické funkce.

2.2.5 Brýle pro virtuální realitu

Brýle pro virtuální realitu[2] se snaží plně upoutat všechny smysly uživatele a vnést jeho vnímání do virtuálního světa [1]. Tento princip můžeme využít u hraní her, nebo také u procházení historických památek a prohlídek interiérů bez nutnosti konkrétní místo skutečně navštívit. Z informací z výše uvedených zdrojů a osobních zkušeností jsem rozdělil brýle pro virtuální realitu do dvou kategorií.

První z kategorií jsou brýle obsahující samostatnou zobrazovací jednotku, obvykle složenou ze dvou LCD displejů. Tyto brýle jsou pak často propojeny s počítačem, který pro ně generuje obraz. Příkladem může být například Oculus Rift[3] či HTC Vive[4].

Druhou, aktuálně čím dál tím více se rozmáhající kategorií jsou pak brýle, které obsahují pouze dvě čočky a obraz je tvořen mobilním telefonem. Obraz chytrého telefonu je pak obvykle speciálním programem rozdělen na dvě poloviny. Takto nastavený mobilní

[1] http://www.mi.com/en/miband/
[2] http://whatis.techtarget.com/definition/VR-headset-virtual-reality-headset
[3] https://www3.oculus.com/en-us/rift/
[4] https://www.vive.com/eu/

telefon se poté vloží do brýlí a uživatel se na dvě rozdělené části displeje dívá skrze jednoduchou optiku. Příkladem může být produkt Google Cardboard [39]. Tyto brýle jsou strohou papírovou krabicí s dvěma čočkami a magnetem pro kvalitní upevnění mobilního telefonu.

Obrázek 2.4: Google Cardboard V2 [39]

2.2.6 Brýle pro rozšířenou realitu

Brýle pro rozšířenou realitu [2] [26] doplňují realitu pomocí virtuálních objektů. Brýle mohou být doplňkem běžného života a sloužit pro veškeré druhy činností, které jejich uživatel aktuálně prožívá.

Obrázek 2.5: Microsoft HoloLens [43]

Do vývoje vlastních virtuálních brýlí se aktivně zapojují velké IT společnosti, jakou je například Microsoft [43]. Na obrázku 2.5 můžeme vidět vývojovou verzi brýlí Microsoft HoloLens. Firma Google se do vývoje brýlí pro rozšířenou realitu zapojila taktéž, nicméně vývoj nových verzí brýlí byl zastaven[1].

[1] https://www.google.com/glass/

Kapitola 3

Chytré hodinky

Chytré hodinky jsou vhodnou volbou pro detailnější popis, protože obsahují důležité komunikační prvky pro komunikaci s okolím a tím i se systémem YSoft SafeQ. Dalším motivem je pak fakt, že jsou mezi uživateli velmi rozšířeny a jejich obliba stále roste[1].

3.1 Historie

V této sekci vycházím z článku [15] Paula Lamkina. V roce 1927 vznikly hodinky určené pro navigaci jejich nositele k cíli[2]. V té době neexistoval globální navigační systém ani LCD displeje. Vytištěná mapa příslušných rozměrů byla svinuta na dvou koncích do svitku a vsunuta do kovových válců. Mezi nimi se nacházel prostor zhruba dvou centimetrů, ve kterém byla rozbalena příslušná část papírové mapy. Takto šlo otáčením papíru v jednotlivých válcích mapu posouvat a udávat tak uživateli směr cesty.

V 70. letech minulého století pak postupně vznikaly různé modely hodinek firmy Pulsar, které mimo jiné nosil i James Bond ve filmu Žít a nechat zemřít. Jednalo se o jednoduché číslicové hodinky, které zobrazovaly digitální čas. Od 80. let minulého století již vznikaly různé hodinky pro sledování televize, počítání na kalkulačce s klávesnicí či dokonce hodinky s FM radiem a reproduktorem. V 90. letech přišly první hodinky od firmy Seiko, které dokázaly předat uživateli informace o zmeškaném volání včetně identifikace volaného, případně umožnily přenášet aktuality ze sportu či předpovědi počasí. Pro přenos informací se používal bezdrátový přenos pomocí frekvenční modulace.

V roce 1998 vznikly první hodinky s operačním systémem Linux. Od té doby již pokračoval vývoj chytrých hodinek do té podoby, jak je známe dnes.

3.2 Obvyklé komunikační kanály

Chytré hodinky obvykle komunikují s okolím pomocí technologií Bluetooth nebo Wi-Fi. V následujících podkapitolách si shrneme tyto dvě a některé další technologie přenosu dat.

3.2.1 NFC

Near Field Communication (viz sekce Near Field Communication na webu [49]) je technologie pro přenos dat pomocí bezdrátového radiového přenosu. Přenos dat je prováděn

[1] https://www.statista.com/statistics/302722/smart-watches-shipments-worldwide/

[2] http://gajitz.com/scrolling-down-the-highway-vintage-1927-analog-gps/

na velmi krátkou vzdálenost, obvykle do 5 až 10 centimetrů. NFC se používá například pro výměnu dat při bezkontaktních platbách. Zařízení NFC mohou být buď aktivní nebo pasivní.

Aktivní NFC zařízení poskytuje pro pasivní zařízení nosné elektromagnetické pole, které poté pasivní člen moduluje. Velká výhoda pasivních členů NFC je nezávislost zařízení na elektrické energii. Jediný zdroj energie pro jejich funkci je ten, který je jim poskytován při snaze o komunikaci s nimi. Nevýhodou je, že dvě pasivní zařízení spolu nejsou schopna komunikovat.

Zabezpečení

Zabezpečení technologie NFC je rozebráno v článku [9], ze kterého budu čerpat informace pro tuto kapitolu. Při používání NFC může dojít k následujícím možnostem narušení bezpečnosti komunikace.

- **Odposlech** je typickou hrozbou bezdrátové komunikace. Útočník může vložit anténu mezi dvě komunikující zařízení a odposlouchávat tak vše, co se v daném pásmu přenáší. Experimentováním či studováním standardu pak útočník může ze získaného radiofrekvenčního signálu získat data. Jelikož přenos probíhá obvykle do vzdálenosti 10 cm, důležitou otázkou je jak daleko může útočník být při snaze o odposlech alespoň části přenášených dat. Na tuto otázku nelze exaktně odpovědět, jelikož záleží na souhře mnoha různých parametrů, mezi které patří například charakteristika vysílací antény, výkon vysílacího zařízení, kvalita antény útočníka a mnoho dalších. NFC neposkytuje dle standardu žádnou ochranu proti odposlechu komunikace. Jedinou možnou obranou je vytvořením zabezpečeného kanálu, který bude popsán níže.

- **Poškození dat** způsobuje nemožnost komunikace. Pokud má útočník dobrý přehled o použitém kódování, které se může lišit od přenosové rychlosti, může v pravý čas začít vysílat signál stejné frekvence, který znehodnotí probíhající komunikaci. Jedná se o příklad útoku za účelem odepření služeb (neboli DoS[1]). Tento typ útoku může být aktéry komunikace lehce detekovatelný. Výkon, který je potřeba vysílat pro znehodnocení informace je značně vyšší než výkon použitý při běžné komunikaci. Aktéři tedy mohou tento typ útoku detekovat a reagovat na něj.

- **Modifikace dat** je zásah do probíhající komunikace takový, aby byla výsledná data validní a čitelná. Možnosti tohoto útoku se silně odvíjejí dle použité amplitudové modulace. Pokud zařízení komunikují rychlostí 106 000 baudů v aktivním módu, je pro útočníka nemožné modifikovat veškerá přenášená data právě kvůli použitému kódování a modulaci dat. Více informací je možné nalézt ve výše zmiňovaném článku [9]. Detekce takového útoku pak je možná neustálým kontrolováním radiové komunikace odesílajícím zařízením. Spolehlivou ochranou je pak pouze zabezpečený kanál.

- **Vložení dat** je útok, který spočívá ve vložení dat do již probíhající komunikace. Je možný pouze tehdy, pokud zařízení, které odpovídá na nějaký dotaz, potřebuje dlouhou dobu na zaslání odpovědi. Útok je úspěšný pouze tehdy, pokud je útočník rychlejší než zařízení generující odpověď. Pokud se vysílání dat překryje, dojde k poruše přenášených dat. Tomuto útoku se dá zabránit například minimalizací doby potřebné k odpovědi určitým zařízením. Další možností je naslouchání zařízením generujícím

[1] https://en.wikipedia.org/wiki/Denial-of-service_attack

odpověď a případným vyhodnocením provedeného útoku. Nejspolehlivější metodou je ovšem opět použití zabezpečeného kanálu.

- **MITM** (*Man in the middle*) je typ útoku, při kterém dochází k podsunutí jiného zařízení do probíhající komunikace. Útočníkovo vsunuté zařízení se tváří jako zařízení, s kterým chtějí účastníci komunikovat. Pokud tedy útočník vloží své zařízení mezi zařízení A a zařízení B, pak tyto dvě zařízení nekomunikují navzájem, ale obě komunikují s útočníkem. Ten komunikaci přeposílá dále, takže zařízení netuší, že mezi ně byl vložen útočník. Útočník pak má přístup k datům, která jsou v daném pásmu vysílána, případně je může i modifikovat. Tento útok je nicméně v prostředí NFC velmi nepravděpodobný. Útočník musí být aktivním zařízením, a musí tak vytvářet další nosné elektromagnetické pole pro pasivního či aktivního klienta. To ovšem může kdykoliv vyústit v překrytí vysílaných dat a dojít tak k detekci daného útoku. Dle výše citovaného článku je tento typ útoku v prostředí NFC komunikace v reálném prostředí téměř nemožný.

Zabezpečený kanál je nejlepším způsobem, jakým chránit NFC komunikaci. Jedná se o jednoduché použití asymetrické kryptografie. Je možné použít například Diffieho-Hellmanovu protokol pro výměnu privátních klíčů.

3.2.2 Bluetooth

Bluetooth (viz sekce Bluetooth na webu [49]) je otevřený standard sloužící pro bezdrátovou komunikaci na krátké vzdálenosti. Využívá elektromagnetického pole o krátké vlnové délce (2,4 až 2,485 GHz). Standard Bluetooth je spravován institucí Bluetooth SIG (Bluetooth Special Interest Group)[1]. V rámci vývoje standardů vznikla také verze zaměřená na zařízení s nízkým příkonem. Tato verze se používá ve zdravotnictví, v chytrých náramcích, v zabezpečovacích prvcích a také v Bluetooth majácích. Jedná se o Bluetooth Smart[2] (nebo také Bluetooth Low Energy, Bluetooth LE, BLE).

Bluetooth majáky

Bluetooth maják (*Bluetooth beacon*) [29] je označení pro zařízení konstruované k přilákání pozornosti ke specifické lokaci. Může se jednat o zajímavé místo, předmět či může sloužit k navádění uživatele dle konkrétní trasy. Elektronická zařízení jsou schopna tento maják detekovat a upozornit uživatele, že je kolem něho v reálném světě něco potenciálně zajímavého. Maják je možné použít i pro zjištění lokace zařízení a tím i uživatele. Obvykle se určování lokace pomocí Bluetooth majáků používá v uzavřených prostorech, kde je obtížné zjistit lokaci podle systému GPS nebo mobilní sítě. Ke svému provozu používá maják komunikační technologii Bluetooth Low Energy. Maják je pak detekovatelný pomocí komunikace Bluetooth, která zároveň umožňuje přenést z majáku datové položky. Maják může být realizován mnoha způsoby. Může jej realizovat mobilní zařízení, může se vyskytovat ve formě malého plastového kamínku, či jakéhokoliv jiného tvaru a materiálu, které v sobě dokáže zapouzdřit napájecí zařízení (obvykle baterie) a samotný Bluetooth LE čip.

iBeacon[3] je standard od společnosti Apple[4], který umožňuje mobilním zařízením s operačním systémem iOS a Android naslouchat signálům z majáků v reálném prostředí a re-

[1] https://www.bluetooth.com/
[2] https://www.bluetooth.com/what-is-bluetooth-technology/how-it-works/low-energy
[3] https://developer.apple.com/ibeacon/
[4] Apple®

agovat na ně. iBeacon standard tedy umožňuje například zjistit polohu uživatele uvnitř místnosti, či předat telefonu zprávu o tom, u jakého objektu reálného světa se právě uživatel ocitl.

Eddystone[1] je otevřený protokol od společnosti Google, který má velmi podobné vlastnosti a stejné využití jako iBeacon od společnosti Apple. Eddystone může být detekován operačním systémem Android i iOS pomocí Bluetooth LE. Protokol Eddystone podporuje více datových typů pro přenos informací. Jsou jimi Eddystone-UID, Eddystone-URL, Eddystone-TLM a Eddystone-EID. Více informací je možné nalézt v knize *Beacon Technologies* [29].

Zabezpečení

Bezpečnostní rizika při použití komunikační technologie Bluetooth jsou podrobně popsány v článku *Bluetooth security threats and solutions: a survey* [19], ze kterého budu čerpat informace pro tuto sekci. Komunikace Bluetooth je takzvanou *master/slave* komunikací. Toto rozděluje zařízení, které právě komunikují, do dvou odlišných rolí. Role *master* označuje zařízení, které moderuje celou komunikaci, kdežto zařízení *slave* je řízeno zařízením role *master*. Navazování důvěryhodných spojení se nazývá párování. Párování je prováděno pomocí výměny sdílených tajných klíčů. Tento klíč je tvořen zašifrovaným PIN kódem.

Jelikož je možné přenášet data pomocí Bluetooth na větší vzdálenosti než pomocí NFC, zabezpečení komunikace zde hraje důležitější roli. Instituce Bluetooth SIG, která spravuje standard Bluetooth, zajistila pro tuto komunikaci mnoho různých bezpečnostních opatření. Zařízení, které používá Bluetooth komunikaci, se vzhledem k možnostem připojitelnosti k ostatním může nacházet ve třech různých módech.

- **Tichý** mód je stav, při kterém zařízení nepřijímá žádné spojení. Zařízení pouze sleduje okolní Bluetooth komunikaci, ale aktivně se jí nijak neúčastní.
- **Privátní** mód značí, že zařízení nemůže být detekováno ostatními zařízeními. Se zařízením se můžeme spojit pouze pokud známe jeho unikátní adresu. Tato adresa se značí *BD_ADDR* a je tvořena 48 bity.
- **Veřejný** mód označuje chování, při kterém může být zařízení detekováno ostatními zařízeními a zároveň je možné se k němu připojit.

Bez ohledu na tyto módy, ve kterých se zařízení může nacházet, může zařízení implementovat různé bezpečnostní režimy, které jsou popsány na straně 131 v článku *Bluetooth security threats and solutions: a survey* [19].

Při práci s Bluetooth existují tři hlavní přístupy k zajištění bezpečnosti komunikace.

- **Autentizace** zahrnuje identifikaci jednoho zařízení druhému. Autentizace je ověřována kontrolou linkových klíčů. Linkový klíč je zašifrovaný PIN kód. Odesílatel zašifruje jednoznačnou adresu příjemce pomocí linkového klíče, který je pak ověřen právě u příjemce. Pokud se tyto klíče shodují, může být mezi zařízeními ustanoveno spojení.
- **Autorizace** je proces přidělení nebo zamítnutí přístupu k síťovému prostředku.
- **Šifrování** je kódování informace, která je vyměňována mezi zařízeními. Toto kódování zabrání odposlechu dat přenášených pomocí Bluetooth komunikace. Šifrování je

[1] https://developers.google.com/beacons/

základem pro Bluetooth zabezpečení. Může využívat 8 až 128 bitové klíče pro šifrování. Vývojář ani uživatel nemůže zasáhnout do délky klíče pro šifrování, jelikož je tato délka určena výrobcem dle regulací jednotlivých států.

Jelikož k šifrování informace, které je navíc dle standardu volitelné, dochází až ke konci párovacího procesu, je zde velké množství potenciálních možností k ohrožení Bluetooth komunikace. Mezi možné útoky patří například podvržení *BD_ADDR* adresy, snaha uhádnout PIN kód, MITM útok, zasílání nevyžádaných zpráv a mnoho dalších.

Z uživatelského hlediska patří k bezpečnostním opatřením především správné nastavení módu, ve kterém se naše zařízení nachází. Pokud nechceme komunikovat, můžeme Bluetooth úplně deaktivovat, nebo jej alespoň nemít ve veřejném módu. Dalším důležitým aspektem zabezpečení je pak výběr PIN kódů, který by neměl být některou ze známých a typických kombinací. Další opatření je možné nalézt v článku *Bluetooth security threats and solutions: a survey* [19] na straně 144 a dále.

3.2.3 Wi-Fi

Wi-Fi [8] je technologie založena na standardu IEEE 802.11. Standard Wi-Fi spravuje organizace Wi-Fi Alliance. Wi-Fi je velmi rozšířeným prostředkem pro připojení zařízení do sítě internet. Komunikace probíhá bezdrátově, většinou na kmitočtech 2,4 GHz nebo 5 GHz. Standard IEEE 802.11 dovoluje pracovat i s frekvencemi 2,6 GHz a 60 GHz.

Tato technologie dovoluje chytrým hodinkám přístup k internetu i bez přítomnosti spárovaného mobilního zařízení. Chytré hodinky využívají obvykle standardu 802.11 b/g, případně 802.11 b/g/n (viz tabulka č.1 v příloze A). Pokud se jedná pouze o standardy 802.11 b/g, probíhá komunikace pouze na frekvenci 2,4 GHz. Pokud čip podporuje standard 802.11 n, některé hodinky umožňují komunikaci i na frekvenci 5 GHz.

Zabezpečení komunikace

O problematice zabezpečení Wi-Fi komunikace pojednává diplomová práce [5] pana Pavla Endreleho, ze které budu čerpat informace pro tuto sekci. Z pohledu zabezpečení a přístupu můžeme Wi-Fi sítě rozdělit do dvou základních kategorií. První z nich jsou otevřené sítě, které žádným způsobem nezabezpečují probíhající komunikaci. Dochází zde pouze k výměně autentizačních rámců pro identifikaci jednotlivých zařízení.

Druhou kategorií jsou sítě, které autentizují zařízení sdíleným klíčem. Tento klíč je stanicím doručen zabezpečeným kanálem. Tento způsob komunikace vyžaduje přítomnost šifrovacího algoritmu. Nyní si uvedeme několik známých šifrovacích algoritmů.

- **WEP** je jedním z prvních hromadně používaných algoritmů pro šifrování sítí Wi-Fi. Dnes již nespolehlivý způsob ochrany bezdrátové Wi-Fi sítě, který je možné za určitých okolností lehce prolomit. Princip algoritmu spočívá ve sčítání hodnot na bitové úrovni přenášené zprávy s generovaným číslem dle aktuálního WEP klíče a inicializačního vektoru.
- **WPA** (*Wi-Fi Protected Access*) opravuje slabiny algoritmu WEP. Jedná se o vylepšení protokolu WEP, které spočívá například v delším inicializačním vektoru, delším privátním klíči nebo v použití TKIP[1] mechanismu.

[1] https://en.wikipedia.org/wiki/Temporal_Key_Integrity_Protocol

- **WPA2** vylepšuje algoritmus WPA a přidává do něj tzv. CCMP[1] protokol. Protokol CCMP implementuje blokovou šifru AES a tím zajišťuje velmi silné zabezpečení pro Wi-Fi síť.

Dalšími možnostmi zabezpečení, které jsou běžně využívány, jsou například skrytí vysílaného SSID[2] nebo filtrace na základě fyzických (MAC) adres jednotlivých klientů. Tyto dvě techniky samy o sobě však nepřináší plnohodnotnou ochranu, vzhledem k jejich relativně jednoduchému prolomení. Proto se doporučuje tyto techniky kombinovat právě s výše vypsanými protokoly pro šifrování komunikace.

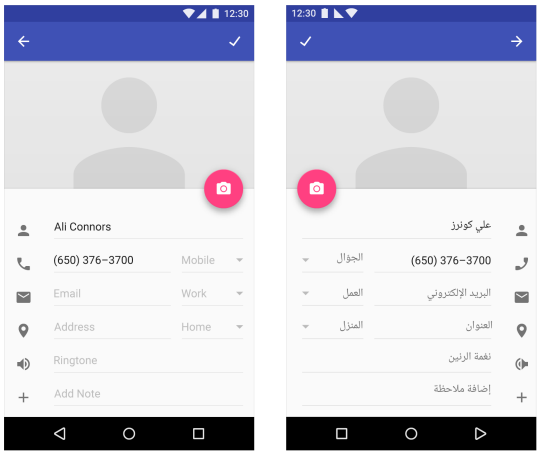
Poslední zmíněnou možnou ochrany Wi-Fi sítě je pak přidělování přístupu pomocí WPA a WPA2. Tyto dva protokoly implementují dva způsoby autentizace.

- **Podnikový mód** (*Enterprise*) je způsob autentizace, který je prováděn na takzvaném Radius serveru. Autentizace je možná na základě různých položek. Jsou jimi například jméno a heslo, certifikát nebo smart karta.
- **Osobní mód** (*Personal*) je způsob autentizace, který se provádí na centrálním bodě komunikace (AP, bezdrátový router). Autentizace probíhá na základě hesla, které je zadáno jak v centrálním bodě komunikace, tak v klientovi, který se snaží autentizovat.

Stejně jako platí u ostatních typů komunikace i zde je důležité při konfiguraci šifrovacích protokolů myslet především na kvalitní volbu hesla.

3.3 Jazyky čtené zprava doleva

Většina světových jazyků se čte zleva doprava. Existují však i jazyky, které se čtou naopak. Jedná se především o arabštinu, hebrejštinu a perštinu (viz sekce *Right-to-left* v internetové encyklopedii [49]).



Jazyk psaný zleva doprava (LTR) Jazyk psaný zprava doleva (RTL)

Obrázek 3.1: Rozdíl mezi angličtinou a hebrejštinou v systému Android (viz sekce *Bidirectionality* v dokumentaci [42])

[1] https://en.wikipedia.org/wiki/CCMP
[2] https://en.wikipedia.org/wiki/Service_set_(802.11_network)

Pokud například čteme obsah elektronické knihy a jsme zvyklí otáčet strany přejetím prstu po displeji z pravé strany na levou, u jazyka čteného zprava doleva je tento pohyb otočen tak, aby byla zachována stejná funkcionalita jako u knihy v daném jazyce. To platí pro všechny akce s pohybem po horizontální ose zařízení. Dále bývají vertikálně převráceny veškeré funkční tlačítka a ikony. Na obrázku č. 3.1 můžeme vidět rozdíl uživatelského grafického rozhraní v jazyce angličtina a hebrejština v systému pro telefonní zařízení Android. Další ukázky můžeme nalézt v sekci *Bidirectionality* v dokumentaci [42], odkud jsem čerpal informace pro tento odstavec.

Po nastavení některého z jazyků čtených zprava doleva tedy od chytrých hodinek očekáváme především převrácení veškerých uživatelských akcí, tlačítek, ikon a textů. Další podrobnosti budou rozepsány u konkrétních operačních systémů.

3.4 Technické parametry a výkon

Technickými parametry chytrých hodinek se zabývá článek redakce BBC s názvem *Smartwatches: Specs and reviews for the leading models WatchBBC* od Lea Keliona, ze kterého budu čerpat některé informace. Ostatní technické parametry komponent společnosti Qualcomm je možné nalézt na jejich oficiálních webových stránkách[1].

Ve výše zmíněném článku je psáno o snaze výrobců chytrých hodinek zvýšit dobu provozu jednotlivých modelů před dobitím jejich baterie. Jelikož kapacita baterií je vzhledem k velikosti a konstrukci těchto zařízení obtížně rozšiřitelná, zaměřují se především na komponenty s nízkým příkonem. Společnost Qualcomm je jeden z nejznámějších výrobců technického vybavení pro mobilní zařízení. Tato firma vyvinula čipovou sadu, která je uzpůsobena pro chytré hodinky a jejich obvykle nízké kapacity baterií. Tato čipová sada s názvem Snapdragon Wear 2100 tak reaguje na aktuální potřeby výrobců nositelných zařízení. Konkrétní technické parametry je možné nalézt na oficiální stránce společnosti Qualcomm[2]. Menším kapacitám baterií se snaží přizpůsobit i operační systémy v chytrých hodinkách. Například operační systém watchOS ve verzi 1 prováděl veškerý program aplikací v přidruženém mobilním telefonu a hodinky pak pouze sloužily k zobrazení grafického uživatelského rozhraní. Další informace ohledně vývoje systému watchOS budou detailněji rozepsány v kapitole 4.2.1.

Jelikož v době psaní této práce je Snapdragon Wear 2100 novinkou, není v běžně dostupných chytrých hodinkách osazen. Aktuálně bývají hodinky osazeny různými čipovými sadami s nízkým příkonem a potřebným technickým vybavením pro zajištění chodu všech senzorů, které chce výrobce do hodinek vložit. Jednou z populárnějších čipových sad je pak Snapdragon 400[3], která je používána pro některé chytré telefony. Čipová sada Snapdragon Wear je osazena čtyřjádrovým procesorem ARM Cortex A7 s taktovacím kmitočtem do 1,2 GHz. Operační paměť typu LPDDR3[4] je taktována na 400 MHz. Oproti tomu sada Snapdragon 400 může být osazena buď dvoujádrovým procesorem Qualcomm Krait 300 s taktovacím kmitočtem do 1,7 GHz, nebo čtyřjádrovým procesorem ARM Cortex A7 s taktovacím kmitočtem do 1,6 GHz. Paměťové moduly typu LPDDR2 nebo LPDDR3 jsou pak taktovány na 533 MHz.

Ostatní parametry chytrých hodinek bývají mezi jednotlivými modely velmi podobné. V příloze A nalezneme tabulku A.1, která obsahuje výčet známých chytrých hodinek i s je-

[1] https://www.qualcomm.com/

[2] https://www.qualcomm.com/products/snapdragon/processors/wear-2100

[3] https://www.qualcomm.com/products/snapdragon/processors/400

[4] https://en.wikipedia.org/wiki/Mobile_DDR

jich technickými parametry. Můžeme zde vidět, že všechny hodinky mají kapacitu operační paměti 512 MB. Výjimkou je model Pebble Times Round. Jelikož společnost Pebble na svém blogu[1] oznámila ukončení podpory a zánik společnosti, nebudou tyto hodinky již dále v textu práce zmiňovány. V tabulce si dále můžeme všimnou AMOLED[2] displejů, které taktéž přispívají k nižší spotřebě elektrické energie.

3.5 Shrnutí

Pro nositelnou elektroniku a konkrétně chytré hodinky jsou typické komunikační technologie, pomocí kterých komunikují se svým okolím. V této kapitole je čtenář uveden do obvyklých komunikačních kanálů chytrých hodinek včetně popisu jejich bezpečnostních rizik. Konkrétně se jedná o komunikační technologie Bluetooth, NFC a Wi-Fi. Chytré hodinky pak disponují i displejem, pomocí kterého předávají uživatelům různé informace. Vzniká zde tedy požadavek na různé formy zobrazení včetně korektního formátu jazyků čtených zprava doleva. V této kapitole nalezneme základní informace o zobrazení těchto jazyků na displeji elektronického zařízení. V předposlední podkapitole je možné nalézt informace o aktuálních technických parametrech chytrých hodinek a jejich výpočetním výkonu.

[1] https://blog.getpebble.com/
[2] https://en.wikipedia.org/wiki/AMOLED

Kapitola 4

Operační systémy chytrých hodinek

Základním rozdílem mezi jednotlivými systémy je absence či dispozice vývojářských nástrojů. Některé operační systémy pro chytré hodinky jsou proprietární, tedy vývojář třetí strany nemá možnost pro dané zařízení tvořit aplikace. Naopak některé systémy jsou otevřené a poskytují vývojářům vývojové nástroje [3] [6]. Ve svém textu se budu zaobírat především systémy, které poskytují vývojářské nástroje a jsou tak zajímavé z pohledu vývoje aplikací a vývojových technologií.

4.1 Android Wear

Operační systém Android Wear (viz článek [25] Jamese Rogersona a sekce Android Wear v internetové encyklopedii [49]) se stává mezi chytrými hodinkami s neproprietárním systémem relativně populární[1]. Jedná se o operační systém od firmy Google, který ve značné míře vychází z operačního systému Android pro mobilní telefony. Android Wear je systém, který spolupracuje s operačními systémy pro mobilní platformy Android 4.3 a vyšší, případně iOS 8.2 a vyšší. Skrze Bluetooth komunikaci se dorozumívá s chytrými hodinkami a umožňuje tak vzájemnou interakci. Android Wear implementuje službu *Google Now* a sdílení notifikací s telefonním zařízením.

Služba *Google Now*[2] zprostředkovává komunikaci uživatele s prostředím, ve kterém je spuštěna. Tato služba je implementována v mobilní aplikaci Vyhledávač Google pro Android i iOS, v internetovém prohlížeči Google Chrome a je základním prvkem interakce v systému Android Wear. Služba Google Now používá uživatelské rozhraní přirozeného jazyka (viz sekce *Natural language user interface* na webu [49]) a na základě hlasového vstupu vytváří doporučení a provádí akce. Pomocí této služby můžeme v chytrých hodinkách psát textové zprávy, spouštět aplikace s určitou činností a především používat vyhledávač Google.

V průběhu psaní textu bakalářské práce se pracovalo na vývoji systému Android Wear verze 2, která na jaře roku 2017 oficiálně vyšla, nicméně tato práce se bude zabývat operačním systém Android Wear verze 1.

[1] https://www.letemsvetemapplem.eu/2016/04/29/apple-watch-ztraci-podil-na-trhu/
[2] https://www.google.com/intl/en-GB/landing/now/

4.1.1 Interakce telefonu s hodinkami a distribuce balíčků do hodinek

V této části textu vycházím z informací ze sekcí *Creating a Notification for Wearables* a *App Distribution*, které je možné nalézt v dokumentaci [40] k systému Android. Instalace aplikace na chytré hodinky probíhá skrze spárovaný mobilní telefon. Vývojáři mají na výběr ze dvou možností, jak zajistit podporu pro chytré hodinky.

- V rámci mobilní aplikace je možné vytvářet notifikace pro chytré hodinky. K těmto notifikacím lze přidat i různé akce, typicky reakce na notifikaci nebo otevření dané aplikace v mobilním telefonu z chytrých hodinek. Notifikace se pak tedy většinou skládá z více stránek, mezi kterými lze listovat. Na první stránce jsou informace o notifikaci (e-mail, SMS zpráva, budík apod.), a na dalších stránkách vždy jedno tlačítko pro určitou akci, například odpovědět na e-mail, SMS zprávu, vypnout budík apod. Listováním na opačnou stranu můžeme notifikaci odstranit.
- Druhou možností je pak k samotné mobilní aplikaci přibalit samostatnou aplikaci pro chytré hodinky. Vývojář tedy píše dvě různé aplikace, jedna je pro mobilní telefon, druhá pro chytré hodinky. Tyto dvě aplikace lze zabalit do jednoho aplikačního balíčku. Po vytvoření balíčku s aplikací pro chytrý telefon pak stačí balíček buď manuálně nebo skrze Obchod Play nainstalovat na mobilní zařízení. Mobilní zařízení již samo redistribuuje balíček do chytrých hodinek, kde je daná aplikace instalována.

4.1.2 Komunikace mezi mobilním telefonem a chytrými hodinkami

Komunikace mezi chytrými hodinkami a mobilním telefonem probíhá skrze `The Wearable Data Layer API` (viz sekce *Sending and Syncing Data* v dokumentaci [40]), které je součástí balíčku `Google Play Services`[1]. Toto aplikační rozhraní spojuje jednotlivé uzly a poskytuje jim prostředky ke komunikaci. Uzlem se rozumí každé Android zařízení, jehož součástí je balíček `Google Play Services`. Všechna Android zařízení, která jsou připojena k internetu pomocí Wi-Fi nebo mobilní sítě, jsou automaticky spojena s centrálním uzlem Google. Chytré hodinky jsou pak pomocí Bluetooth přímo připojeny k telefonu. Pokud tedy vlastníme dvoje hodinky, jedny připojeny k mobilnímu telefonu pomocí Bluetooth, druhé připojeny k internetu pomocí Wi-Fi a zároveň je náš mobilní telefon připojen k Wi-Fi, všechna tato tři zařízení spolu mohou komunikovat pomocí `The Wearable Data Layer API`.

Toto aplikační rozhraní implementuje dva způsoby [4] [34] výměny informací mezi zařízeními s operačním systémem Android nebo Android Wear. První z nich je `MessageAPI`. Jedná se o jednorázové jednosměrné zaslání zprávy pro konkrétní uzel v naší síti uzlů. Tento způsob se často využívá k vzdálenému volání procedur. Druhý způsob, jakým můžeme komunikovat mezi hodinkami a telefonem, je `DataAPI`. Jedná se o sdílený prostor s daty, který se po modifikaci znovu distribuuje sítí a všechna zařízení jsou o změně dat upozorněna. Obě tato aplikační rozhraní jsou implementována v třídě `GoogleApiClient`. Tato třída implementuje ještě jedno rozhraní, které ovšem neslouží k zasílání dat. Jedná se o `NodeAPI`. Díky tomuto aplikačnímu rozhraní je nám umožněno zjistit, zda jsme připojeni k nějakým uzlům. Dále nám poskytuje základní informace o připojených uzlech pro jejich rozeznání.

Android Wear 1 neumožňuje pro vývojáře přístup k HTTP klientovi, ani jiným prostředkům zajišťující komunikaci s internetem. Pokud chtějí chytré hodinky komunikovat

[1] https://developers.google.com/android/guides/overview

s jiným uzlem v síti internet, například stáhnout informace z nějakého serveru, musí nejprve požadavek zaslat mobilnímu zařízení, které pak zprostředkuje komunikaci se serverem a zašle do chytrých hodinek požadovaný výsledek. Tento princip je na rozdíl od jiných operačních systémů v režii programátora. Tedy programátor musí naprogramovat komunikaci mezi zařízeními a stáhnutí požadovaných dat v rámci mobilní aplikace.

4.1.3 Přístup k NFC a Bluetooth majákům

Android Wear hodinky mohou být sice osazeny čipem pro NFC, nicméně vývojářům není umožněno k NFC přistupovat. Jiná situace už je v případě Bluetooth majáků. Zde může vývojář využít `Android Bluetooth API` (viz sekce Bluetooth v dokumentaci [40]) pro vyhledávání ostatních zařízení disponující technologií Bluetooth. Může se k nim libovolně připojovat a přenášet mezi nimi data.

4.1.4 Internacionalizace

Internacionalizace operačního systému Android Wear je komplikovanější než internacionalizace operačního systému Android pro mobilní telefony z důvodu časté absence velkého množství světových jazyků. Konkrétní podporovaná množina jazyků závisí od konkrétního modelu chytrých hodinek. Například chytré hodinky Moto 360 od společnosti Motorola podporují pouze devět jazyků[1]. Patří mezi ně angličtina, němčina, japonština, ruština a další. Jelikož uživatelským vstupem chytrých hodinek je nejčastěji lidský hlas, má v operačním systému Android Wear širší podporu. Jazyk se v systému Android Wear odvodí z nastavení jazyku v telefonním zařízení, s kterým jsou hodinky spojeny. To platí jak pro nastavení jazyku systému, tak pro jazyk hlasového vstupu. Pokud jazyk systému není hodinkami podporován, systém v hodinkách vybere automaticky angličtinu a uživatele o této skutečnosti v chytrých hodinkách upozorní. Jazyk hlasového vstupu do značné míry ovlivní ovládání operačního systému [2]. Pro spouštění a ovládání některých aplikací hlasem nejsou vždy k dispozici ekvivalenty příkazů pro všechny světové jazyky. Anglicky mluvící uživatelé tímto získávají výhodu (viz sekce *Adding Voice Capabilities* v dokumentaci [40]). Pokud bude však uživatel diktovat například znění SMS zprávy, nijak jej absence angličtiny nebo jiného světového jazyku neovlivní. Škála jazyků pro rozpoznání řeči je stejná jako pro mobilní telefon.

Správa zdrojů

Informace v následujících odstavcích jsou čerpány ze sekce *Providing Resources* v dokumentaci [40] k systému Android.

Internacionalizace aplikací a notifikací pro Android Wear je velmi podobná jako pro aplikace a notifikace systému Android pro mobilní zařízení. Android Wear i Android pro mobilní telefony obsahuje mechanismus, který za běhu programu dokáže rozeznat konfiguraci daného zařízení a dle toho vybrat správné zdroje k zobrazení. Tento mechanismus je využíván především pro zjištění rozlišení obrazovky a jazykové konfigurace daného zařízení. Operační systém po detekci dané konfigurace vybere odpovídající zdroje. Pokud tyto zdroje nejsou k dispozici, přechází na výchozí konfiguraci, která musí být vývojářem specifikována.

[1] https://motorola-global-portal.custhelp.com/app/answers/prod_answer_detail/a_id/100837/
[2] https://youtu.be/_rtzrSvRCbA

Při vytváření projektu pomocí `Android SDK Tools`[1] je automaticky vytvořen adresář `res`, obsahující veškeré zdroje. Jedná se především o obrázky a jazykové mutace různých řetězců. Tento adresář pak obsahuje adresáře s názvy v pevně daném formátu. Formát adresáře je následující: `<typ-zdroje>-<kvalifikátor>`, kde `<typ-zdroje>` je typ hodnot, které budou v adresáři obsaženy a `<kvalifikátor>` je jednoznačné určení specifické konfigurace systému. Pokud nezadáme kvalifikátor, jedná se o výchozí hodnotu konfigurace. Tato konfigurace se použije v případě, kdy systémová konfigurace nevyhovuje žádnému kvalifikátoru.

```
res
├── values
│   └── strings.xml
├── values-fr
│   └── strings.xml
├── drawable
│   └── graphic.png
└── drawable-hdpi
    └── graphic.png
```

Obrázek 4.1: Ukázka struktury projektu Android Wear aplikace

Na obrázku č. 4.1 vidíme dva příklady použití správy zdrojů. Projekt obsahuje jazykové řetězce pro francouzštinu v souboru `strings.xml` v adresáři `values-fr`. Pro všechny ostatní jazyky se zvolí hodnoty v souboru `strings.xml` v adresáři `values`. Obdobně je tomu u adresáře `drawable`, který obsahuje jeden obrázek k vykreslení. Tento obrázek se použije, pokud rozlišení displeje nespadá do kategorie `hdpi`. Pokud systémové rozlišení spadá do kategorie `hdpi`, je vykreslen obrázek z adresáře `drawable-hdpi`. Další varianty typů zdrojů a kvalifikátorů můžeme nalézt v dokumentaci [40].

Jelikož škála jazyků podporovaných konkrétním zařízením není obvykle vysoká, vývojáři aplikací třetích stran se mohou dostat do situace, kdy nemohou podpořit některý ze známých jazyků. Při selekci zdroje pro konkrétní aplikaci se systém řídí striktně dle jazykového nastavení operačního systému, na kterém je aplikace provozována. Pokud je například v telefonním zařízení nastaven český jazyk a chytré hodinky jej nepodporují, nebudou v aplikaci použity řetězce z adresáře `values-cs`, nýbrž výchozí řetězce z adresáře `values`. Vývojáři pak mohou ve vlastní režii zjistit z mobilního zařízení podrobnější informace o nastaveném jazyku a tuto skutečnost uplatnit pro aplikaci v chytrých hodinkách. Za běhu aplikace je umožněno vývojářům měnit správu zdrojů pro konkrétní aplikaci.

Jazyky čtené zprava doleva

Při vývoji aplikace pro systém Android Wear můžeme použít mechanismy, které byly přidány do operačního systému Android již ve verzi 4.2 [17]. Při vytváření uživatelského rozhraní pak stačí všem elementům místo atributů `left`/`right` zapsat jejich ekvivalentní hodnoty `start`/`end`. Tím můžeme docílit například toho, že text bude automaticky zarovnán ke správné straně, dle aktuální jazykové konfigurace systému. Dalším nezbytným krokem pro podporu RTL jazyků je přidání atributu `android:supportsRtl="true"` do souboru `AndroidManifest.xml`, který se nachází v kořenovém adresáři aplikace.

[1] https://developer.android.com/studio/releases/sdk-tools.html

Pro jazyky čtené zprava doleva můžeme vytvářet design naší aplikace zvlášť. K tomu účelu nám slouží kvantifikátor `ldrtl` [17]. Pokud je v systémové konfiguraci nastaven jazyk, který je čtený zprava doleva, uplatní se tento design před výchozí podobou aplikace. Tímto můžeme snadno docílit uživatelsky přívětivé aplikace i pro osoby preferující jazyky čtené zprava doleva.

Důležité je si však uvědomit, že můžeme být limitováni jazykovou podporou konkrétních hodinek se systémem Android Wear. Pokud tedy chytré hodinky nepodporují žádný jazyk psaný zprava doleva, výše uvedené metody se nebudou moci uplatnit.

4.2 watchOS

watchOS[1] je operační systém od firmy Apple, který je používán výhradně pro zařízení Apple Watch. Vychází z operačního systému iOS, který slouží pro mobilní telefony firmy Apple. Verze operačního systému se od sebe do značné míry liší, aktuálně nejnovější verzí je verze 3.2 [13] [31]. Každá aplikace pro operační systém watchOS je složena ze dvou částí (viz sekce *The Watch App Architecture* v dokumentaci [36]). Jedná se o Watch aplikaci a `WatchKit` rozšíření. Watch aplikace obsahuje definici veškerých prvků grafického uživatelského rozhraní a všechny neměnné zdroje. Naproti tomu `WatchKit` rozšíření obsahuje kód pro práci s uživatelským prostředím a kód pro další výpočty potřebné pro běh aplikace.

4.2.1 Rozdíly ve verzích watchOS

V následujícím textu vycházím ze sekce watchOS na webu [49] a [13]. První verze watchOS (watchOS 1.x) obsahovala 20 předinstalovaných aplikací. Běh aplikací nebyl nativní, jelikož `WatchKit` rozšíření běželo na mobilním telefonu, který byl s hodinkami spárován. Hodinky tak sloužily pouze pro zobrazení výstupů dané aplikace. Na hodinky to kladlo daleko menší technické nároky, nicméně provádění aplikace bylo limitováno rychlostí komunikace spárovaných zařízení.

Druhá verze (watchOS 2.x) přináší zásadní změnu v podobě běhu nativních aplikací pro chytré hodinky přímo na zařízení Apple Watch. `WatchKit` rozšíření tedy běží přímo na hodinkách bez návaznosti s mobilním zařízením. Je zde podpora obousměrné komunikace mezi telefonem a hodinkami. Dále jsou vývojářům zpřístupněny informace z akcelerometru, měřiče tepu, mikrofonu a mnoho dalších. Bohužel, vývojáři nemají přístup k NFC ani Bluetooth.

Třetí verze, momentálně aktuální, přináší novinky v podobě funkce SOS, která dokáže přivolat pomoc uživateli chytrých hodinek v případě nouze a nebezpečí [23]. Při stisku bočního tlačítka po dobu pěti sekund je automaticky spuštěno tísňové volání. Další důležitou vlastností je podpora aplikačního rozhraní `HomeKit`, která slouží pro ovládání IoT (*Internet of Things*) zařízení v domácnosti uživatele. Pokud je uživatel vlastníkem notebooku se systémem macOS Sierra nebo vyšším, je mu umožněno pomocí chytrých hodinek odemykat jeho zařízení (viz sekce watchOS na webu [49]). Více informací o změnách v aplikačním rozhraní lze nalézt v dokumentaci[2].

[1] https://developer.apple.com/watchOS/

[2] https://developer.apple.com/library/prerelease/content/releasenotes/General/watchOS30APIDiffs/index.html

4.2.2 Distribuce balíčku do hodinek

Experimentálně jsem zjistil, že po spárování hodinek a telefonu pomocí Bluetooth můžeme v mobilním zařízení spravovat aplikace pro chytré hodinky pomocí aplikace Watch. Každá aplikace pro mobilní telefon, která v sobě obsahuje balíček pro chytré hodinky, je zde zobrazena a lze do hodinek manuálně instalovat, případně z hodinek odinstalovat.

4.2.3 Komunikace mezi mobilním telefonem a chytrými hodinkami

Párovat je možno pouze zařízení s operačním systémem iOS verze 8 nebo vyšší. Komunikaci mezi telefonem a hodinkami zajišťuje rozhraní `Watch Connectivity` (viz sekce `Watch Connectivity` v dokumentaci [36]). Toto rozhraní implementuje třídu `WCSession` (viz `WCSession` v dokumentaci [36] k systému watchOS), která zajišťuje komunikaci mezi `WatchKit` rozšířením a iOS aplikací. Jak iOS aplikace, tak `WatchKit` rozšíření musí vytvořit instanci této třídy a nakonfigurovat ji. Tyto instance (dále jim budeme říkat sezení) mohou být ve stavu aktivním, neaktivním a deaktivována. Po vytvoření sezení jej musí vývojář aktivovat. Jakmile jsou obě sezení v aktivním stavu, může kdykoliv jakákoliv strana započít komunikaci.

Dále budu v textu čerpat informace ze sekce `WCSession` v dokumentaci [36]. Při průběhu odpojení a připojení chytrých hodinek k mobilnímu telefonu dochází ke změnám stavu sezení. Vezměme v potaz výchozí stav takový, že telefon je spárován s prvními hodinkami. Obě sezení jsou tedy v aktivním stavu. Po odpojení chytrých hodinek v telefonní aplikaci se sezení v telefonním zařízení změní na neaktivní. Neaktivní stav v mobilním zařízení dává zařízením malý prostor času pro výměnu dat, která se ještě nestihla vlivem asynchronního přenosu přenést. Jakmile jsou data doručena, sezení se přesune do deaktivovaného stavu. V tomto bodě již může iOS aplikace kdykoliv zavolat příslušnou metodu pro připojení nových hodinek.

Pokud máme sezení aktivní, můžeme započít komunikaci. Máme na výběr jednu z následujících možností:

- **`updateApplicationContext()`** zašle datovou strukturu slovník pro synchronizaci stavu do spárovaného zařízení. Systém zašle data tak, aby je měla protistrana v čase jejího probuzení. Tato metoda nevyžaduje dosažitelnost protistrany.
- **`sendMessage()`** a **`sendMessageData()`** používá asynchronní přenos dat do dosažitelného zařízení. Přenos je vyvolán okamžitě. Přenos dat je serializován ve stejném pořadí, v jakém došlo k zasílání dat. Před voláním těchto metod musíme zajistit, že je zařízení dosažitelné. Na to slouží metoda `isReachable()`. Metoda `sendMessage()` zasílá slovník, oproti tomu metoda `sendMessageData()` zasílá objekt typu `Data`.
- **`transferUserInfo()`** podobně jako metoda `sendMessage()` zašle slovník, nicméně nemusí být odeslán okamžitě. Pořadí odeslání dat je nicméně stále zachováno.
- **`transferFile()`** je metoda, která zašle požadovaný soubor své protistraně. Soubor musí být pro odesílatele čitelný. Přenos je prováděn asynchronně.

Všechny zmíněné metody komunikace se provádí na pozadí ve vlastním vlákně.

4.2.4 Internacionalizace

Operační systém watchOS 1.0 podporoval 13 základních jazyků[1] včetně angličtiny, čínštiny, japonštiny, italštiny a dalších. Aktualizace watchOS 1.0.1 pak přinesla podporu pro další 4 jazyky. Žádný z těchto jazyků nepatřil do kategorie jazyků čtených zprava doleva. watchOS 2.1 přinesl další řadu systémem podporovaných jazyků, včetně jazyků čtených zprava doleva. Dále rozšířil takzvané diktované jazyky, tedy jazyky pomocí nichž můžeme hlasově zadávat příkazy o češtinu, řečtinu, hebrejštinu a další.[2]

Techniky internacionalizace

Při internacionalizaci systému watchOS můžeme využít stejné principy, které můžeme znát z internacionalizace aplikací pro mobilní platformu iOS. Internacionalizaci pro systém watchOS můžeme rozdělit do dvou základních skupin. Jedná se o internacionalizaci `storyboard` a internacionalizaci dynamických řetězců.

Dále budu v této sekci vycházet z článku o internacionalizaci iOS 8 v prostředí Xcode [33]. Statické řetězce, které se nacházejí například v tlačítkách a nadpisech, lze jednoduše internacionalizovat přímo ve `storyboard`. `Storyboard` je vizuální reprezentace přechodů mezi jednotlivými obrazovkami na zařízení se systémy iOS, watchOS a dalšími. V rámci vytváření `storyboard` pomocí standardních vývojových prostředí máme možnost vygenerovat více jazykových verzí. To vyústí k vygenerování souboru pro konkrétní jazyk ke každé `storyboard`. V rámci těchto souborů můžeme pomocí vygenerovaného klíče pro konkrétní prvek uživatelského rozhraní definovat řetězce pro konkrétní jazyk.

Dynamické řetězce je možné lokalizovat pomocí maker z rodiny `NSLocalizedString`. Tato makra vyhledávají za běhu aplikace pro jednotlivé klíče konkrétní překlad v souborech řetězců dle aktuální jazykové konfigurace zařízení. Nejprve je tedy nutné vytvořit soubory řetězců pro jednotlivé jazyky. Pomocí standardních vývojářských technologií můžeme do projektu přidat soubor řetězců s názvem `Localizable`. Tento soubor obsahuje položky ve tvaru: `<klíč> = "<řetězec>";`. Pro podporu více jazyků pak například ve vývojovém prostředí Xcode vybereme vytvořený soubor `Localizable` a zvolíme tlačítko lokalizovat. Vybereme příslušné jazyky a vývojové prostředí nám vygeneruje soubor pro každý zvolený jazyk. V těchto souborech už nalezneme klíče z původního (tzv. *base*) souboru. Původní soubor je vždy použit, pokud nebylo možné pro konkrétní jazyk najít odpovídající řetězec. Stačí tedy ke klíči dopsat odpovídající překlad.

Jazyky psané zprava doleva

Jak již bylo zmíněno v úvodu sekce 4.2.4, watchOS 2.1 přinesl podporu jazyků psaných zprava doleva. Při vývoji aplikací pro watchOS jsem si ověřil, že pokud je nastavený systémový jazyk na jazyk z podmnožiny jazyků psaných/čtených zprava doleva, otočí se bez zásahu programátora celé uživatelské rozhraní. Není tedy třeba nijak explicitně nastavovat pozici grafických elementů, vše za nás obstará operační systém chytrých hodinek. Převrátí se i všechny animace přechodů na další stránky aplikace. Pokud tedy chce vývojář podpořit ve své aplikaci jazyk psaný zprava doleva, jeho jediným úkolem je ke všem řetězcům obsažených v `storyboard` a ke všem dynamickým řetězcům vytvořit významové ekvivalentní řetězce v daném jazyce. Systém watchOS převrací pouze komponenty uživatelského rozhraní, nikoliv textové řetězce.

[1] http://www.idownloadblog.com/2015/05/04/change-language-apple-watch/
[2] https://support.apple.com/kb/DL1840?locale=cs_CZ

4.2.5 Přístup k NFC a Bluetooth majákům

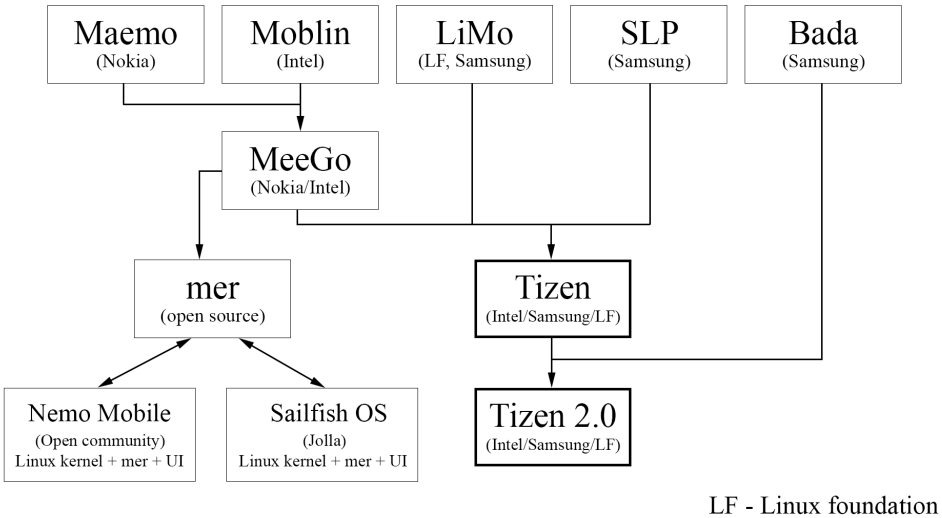
Apple Watch obsahují technické vybavení pro komunikační technologii NFC. Tuto technologii používá služba Apple Pay [37], která umožňuje platit platební nebo debetní kartou pomocí hodinek Apple Watch, telefonu iPhone a pomocí dalších Apple zařízeních. Společnost Apple avšak nezpřístupnila tuto komunikační technologii vývojářům. Vývojáři tedy nemají žádný přístup k NFC a nemohou jej využít pro své užití.

Situace ohledně Bluetooth majáků je velmi podobná. Jak popisuje článek [18] autora Bogdana Melnychuka, samotným hodinkám s operačním systémem watchOS 2 nebo 3 není umožněno pomocí aplikačního rozhraní detekovat majáky, které se v jeho okolí nachází. Výjimkou je systém watchOS 1, kde je rozšíření `WatchKit` spuštěno na mobilním telefonu, nikoliv v chytrých hodinkách. Tato situace se na novějších verzích systému watchOS dá řešit díky komunikačnímu rozhraní `WCSession`. Hodinky mohou chytrému telefonu zaslat žádost o získání informací o majácích, které se v jeho okolí nacházejí. Záleží pak na účelu dané aplikace, zda je nezbytně nutné, aby tuto funkcionalitu dělaly přímo chytré hodinky, nebo zda stačí, aby ji prováděl mobilní telefon uživatele aplikace.

4.3 Tizen

Tizen je standardizovaná platforma s otevřeným softwarem vytvořená pro vývoj a běh aplikací na chytrých telefonech, tabletech, automobilech (*in-vehicle infotainment*[1]) a chytrých televizích [12]. Tizen je zcela standardizován a umožňuje běh aplikací na různých architekturách. Vývoj platformy Tizen je veden organizacemi Tizen Association a Technical Steering Group (TSG).

4.3.1 Historie platformy Tizen



Obrázek 4.2: Vznik platformy Tizen (viz sekce Tizen na webu [49])

[1] https://en.wikipedia.org/wiki/In-car_entertainment

Ke konci roku 2011 oznámila The Linux Foundation[1] příchod projektu Tizen [12]. V roce 2012 vznikla přejmenováním LiMo Foundation organizace Tizen Association[2] za účelem podpory tohoto projektu. Tizen Association je vedena představenstvem firem Samsung, Intel, Huawei, Fujitsu a dalších (viz sekce Tizen na webu [49]).

Jak je vidět na obrázku číslo 4.2, Tizen je pokračováním předchozích platforem MeeGo, LiMo a Samsung Linux Platform (SLP).

4.3.2 Distribuce aplikací do hodinek se systémem Tizen

Aplikace pro chytré hodinky s operačním systémem Tizen lze nově instalovat přímo v hodinkách. Toto platí i pro hodinky Samsung Gear S3 [46]. Další možností je pak nainstalovat aplikace pro chytré hodinky skrze mobilní aplikaci Samsung Gear[3], kterou je možné nainstalovat na mobilní telefon se systémem Android.

Pokud se uživatel rozhodne instalovat aplikaci přímo v chytrých hodinkách, po zmáčknutí příslušného tlačítka na hodinkách se dostane na seznam aplikací, kde může nalézt tlačítko pro instalaci nové aplikace. V seznamu aplikací pak vybere aplikaci o kterou má zájem a nainstaluje ji.

Instalace z telefonu s mobilním systémem Android začíná úspěšným spárováním telefonu s hodinkami pomocí aplikace Samsung Gear. Po spárování je možné v aplikaci Samsung Gear nalézt možnost instalace dalších aplikací pro chytré hodinky s operačním systémem Tizen.

Konfigurace nainstalovaných aplikací[4] pak probíhá vždy skrze mobilní telefon, kde v aplikaci Samsung Gear nalezneme požadovanou aplikaci a můžeme měnit její konfiguraci.

4.3.3 Komunikace mezi mobilním telefonem a chytrými hodinkami

Princip platformy Tizen je co se týče komunikačních schopností hodinek dost odlišný od principu komunikace u dvou předešlých operačních systémů pro chytré hodinky (viz sekce *Connectivity and Wireless* v Tizen dokumentaci [48]). Hodinky s operačním systémem Tizen jsou schopny fungovat nezávisle na mobilním zařízení a mají téměř stejné možnosti komunikace jako mobilní telefony. Je důležité si uvědomit, že na rozdíl od systémů watchOS a Android Wear zde nemusíme mít žádnou aplikaci pro mobilní telefon. Následující scénáře tedy budou pojednávat o komunikaci mezi zařízeními jako takovými.

Pokud chceme komunikovat s Android mobilním telefonem, můžeme postupovat obdobně jako Adam Nadoba v článku [20]. Budeme potřebovat Android aplikaci, kde implementujeme službu nebo aktivitu (dle toho, zda chceme komunikovat i pokud aplikace v telefonu nebude aktivní). V rámci této služby nebo aktivity potom implementujeme komunikační rozhraní, které bude schopno komunikovat s chytrými hodinkami se systémem Tizen.

Pokud máme mobilní zařízení i chytré hodinky s operačním systémem Tizen, můžeme využít některého z komunikačních aplikačních rozhraní, které Tizen nabízí (viz sekce *Connectivity and Wireless*). Tento scénář je však méně častý.

[1] https://www.linuxfoundation.org/
[2] http://www.tizenassociation.org/
[3] https://play.google.com/store/apps/details?id=com.samsung.android.app.watchmanager
[4] http://www.samsung.com/us/support/answer/ANS00061436/

4.3.4 Komunikační rozhraní chytrých hodinek

Ohledně komunikačních schopností operačního systému Tizen pro chytré hodinky se lze dočíst v dokumentaci [48] v sekci *Connectivity and Wireless*. Odtud budou čerpány informace pro tuto sekci. Dostupná sada aplikačních rozhraní se může lehce měnit dle zvoleného typu vývoje aplikací, které budou popsány v kapitole 5.4. Všechny níže vypsané způsoby komunikace však jsou v určité míře pro vývojáře dostupné v obou typech vývoje Tizen aplikace.

Chytré hodinky s operačním systémem Tizen mohou komunikovat pomocí různých komunikačních rozhraní. Operační systém Tizen 2.3.4 a vyšší pro chytré hodinky umožňuje vývojářům přímý přístup k NFC aplikačnímu rozhraní. Vývojáři tedy mohou tvořit aplikace, které dokáží číst okolní aktivní či pasivní NFC zařízení. Vývojáři také mají plný přístup k Bluetooth. Mohou chytré hodinky spárovat s jiným Bluetooth zařízením pro výměnu dat nebo číst Bluetooth majáky pomocí příslušných aplikačních rozhraní. Vývojáři mají možnost využít aplikační rozhraní Smart Card. Více informací k tomuto rozhraní je možné nalézt v sekci Smart Card v dokumentaci [48]. Tizen také zpřístupnil možnost z aplikace ovládat Wi-Fi modul daného zařízení. Je tedy možné v rámci aplikace připojit hodinky ke konkrétní síti. Jedinou podmínkou pro danou síť je viditelné SSID[1]. Pokud jsou hodinky připojeny k síti, je možné využít HTTP klienta pro komunikaci se síťovým uzlem pomocí protokolu HTTP/HTTPS.

Dalším způsobem komunikace s okolím je pak rozhraní *IoT Connectivity*. Toto rozhraní slouží pro komunikaci se zařízeními implementujícími IoTivity[2]. Jedná se o univerzální komunikaci mezi dvěma zařízeními prostřednictvím sítě internet. IoTivity staví svoji funkčnost nad nejznámějšími komunikačními rozhraními jakými jsou Wi-Fi Direct, Bluetooth Low Energy, ANT+, Zigbee a dalšími. Komunikační rozhraní IoTivity již podporuje například známý systém openWRT[3] pro síťové směrovače.

4.3.5 Internacionalizace

Operační systém Tizen podporuje mnoho světových jazyků (viz sekce *Internationalization* a *Localization* v dokumentaci [48]). Nastavení jazyka operačního systému se provádí přímo v chytrých hodinkách a konkrétně v systému Tizen verze 3 je možné si vybrat ze 49 možností. V rámci těchto jazyků můžeme zvolit i jazyky psané zprava doleva. Po zvolení jednoho z těchto jazyků, například arabštiny, se ovšem grafické uživatelské rozhraní nezmění. Nedochází k žádné automatické změně grafických elementů, pouze se pro systémové oblasti zobrazují řetězce v příslušném jazyce.

Konkrétní techniky internacionalizace se do určité míry liší dle zvoleného typu vývojové sady. Proto budou tyto techniky popsány až v sekci 5.4.

4.4 Shrnutí a porovnání operačních systémů

V této kapitole se čtenář dozvěděl o operačních systémech pro chytré hodinky pro které je umožněno vývojářům třetích stran vyvíjet aplikace. Nalezne zde jejich podrobný popis, možnosti distribuce aplikací do těchto systémů, jejich možnosti internacionalizace a způsob,

[1] https://en.wikipedia.org/wiki/Service_set_(802.11_network)
[2] https://www.iotivity.org/
[3] https://openwrt.org/

jakým tyto zařízení komunikují s okolím. Následuje porovnání dle dostupnosti aplikačních rozhraní pro ovládání důležitých komunikačních technologií.

Komunikační technologie	**Android Wear**	**watchOS**	**Tizen**
NFC	✗	✗	✓
Bluetooth majáky	✓	✗	✓

Tabulka 4.1: Dostupnost aplikačních rozhraní pro NFC a Bluetooth pro jednotlivé operační systémy

Pokud tedy například budeme po aplikaci vyžadovat komunikaci skrze NFC i detekci okolních Bluetooth majáků, pak jedinou možnou volbou bude operační systém Tizen. Tímto ovšem omezíme běh aplikace pouze na pět konkrétních hodinek[1], na kterých aktuálně operační systém Tizen běží. Dalším možným porovnáním je možnost internacionalizace a podpora jazyků psaných zprava doleva.

Vlastnost	**Android Wear**	**watchOS**	**Tizen**
Internacionalizace	✓	✓	✓
Podpora jazyků RTL	✓	✓	✓
Automatické převrácení GUI po detekci RTL jazyka	✗	✓	✗

Tabulka 4.2: Internacionalizace a podpora jazyků psaných zprava doleva v operačních systémech chytrých hodinek

Všechny zmiňované operační systémy umožňují určitým způsobem internacionalizaci aplikací a tvorbu různých grafických uživatelských rozhraní dle typu nastaveného jazyka. Automatické převracení celého grafického uživatelského rozhraní po detekci jazyka psaného zprava doleva však provádí pouze systém watchOS.

[1] https://developer.tizen.org/tizen/wearable

Kapitola 5

Vývojářské technologie chytrých hodinek

Vývojářské technologie pro chytré hodinky se liší podle operačního systému, který je na daném zařízení provozován. Někteří výrobci operačních systémů chytrých hodinek nepodporují vývoj aplikací pro vývojáře třetích stran. Uživatelé těchto hodinek jsou pak odkázáni na předinstalovanou sadu aplikací, která je neměnná nebo jen velmi omezeně modifikovatelná a doplňovatelná o aplikace, které sám výrobce zpřístupní. Výrobci chytrých hodinek však mohou použít otevřený operační systém, pro který je umožněno vyvíjet aplikace. Společnost, která daný operační systém spravuje, vydá a průběžně aktualizuje takzvané SDK (sekce *Software development kit* v internetové encyklopedii [49]). SDK je sada vývojářských nástrojů, která umožňuje vyvinout aplikaci pro konkrétní cílovou platformu. Tyto nástroje jsou většinou dostupné v rámci specifického vývojového prostředí (viz sekce *Integrated development environment* na webu [49]). Vývojové prostředí (nebo také IDE) je sada aplikací, která poskytuje kompletní zázemí pro vývoj aplikace. Skládá se obvykle z editoru zdrojového kódu, automatizačních nástrojů pro překlad programu, nástrojů pro ladění napsaného programu a také již zmíněné sady vývojářských nástrojů. V následujících kapitolách si ukážeme některá vývojová prostředí pro vývoj aplikací na nejznámější operační systémy chytrých hodinek.

5.1 Nativní vývoj Android aplikací

Nativní vývoj aplikací je realizován pomocí Android Studia (sekce Android Studio v dokumentaci [40]). Android Studio je vývojové prostředí založené na IntelliJ IDEA[1] [22]. Jedná se o vývojové prostředí pro vývoj aplikací pro operační systémy Android a Android Wear.

5.1.1 Android Studio

Od roku 2014 je Android Studio hlavní vývojové prostředí pro vývoj aplikací pro platformu Android a nahradilo tak vývojové prostředí Eclipse. Jak již bylo zmíněno, Android Studio je založeno na IntelliJ IDEA. IntelliJ IDEA je vývojové prostředí od společnosti JetBrains. Slouží především pro vývoj v programovacím jazyce Java.

[1] https://www.jetbrains.com/idea/

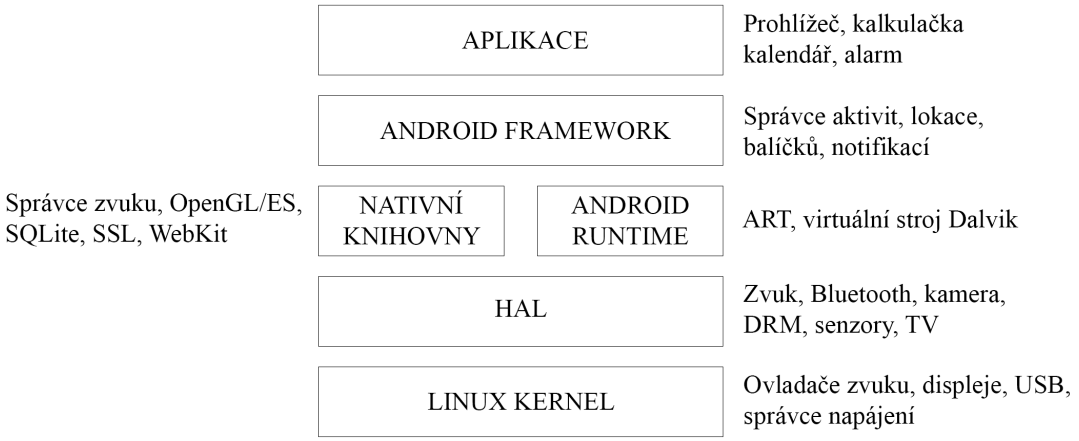
Následující informace budu čerpat převážné z knihy *Android Programming* [22]. Jako prerekvizitu pro vývoj aplikací budeme potřebovat *Java Development Kit*[1] (zkrácené JDK). JDK je produkt firmy Oracle obsahující základní nástroje pro vývoj aplikací v jazyce Java.

Při instalaci Android Studia jsou pak staženy a instalovány následující komponenty:

- Vývojové prostředí IntelliJ IDE s rozšířením pro tvorbu Android projektů.
- `Android SDK Tools` – sada nástrojů pro vývoj aplikací pro platformu Android. Jedná se například o sadu *Android Virtual Devices* (zkráceně AVD) pro emulování jednotlivých Android zařízení. Dále v ní můžeme nalézt nástroje pro ladění kódu (například DDMS[2]) a mnoho dalších.
- `Android Platform-tools` – sada nástrojů, která je aktualizována s každým novým vydáním `Android SDK Tools`. Obsahuje například ADB[3], což je programový můstek pro komunikaci s emulátorem nebo mezi zařízeními navzájem.
- Obraz systému Android pro spuštění v emulátoru.

5.1.2 Operační systém Android z pohledu vývojáře

Informace ohledně implementace operačního systému Android je možné nalézt v dokumentaci [35] k projektu Android. V této sekci bude uveden teoretický základ pro pochopení fungování operačního systému Android.



Obrázek 5.1: Android model (viz sekce *The Android Source Code* v dokumentaci [35])

Na obrázku č. 5.1 můžeme vidět, že operační systém Android je tvořen pětivrstvým modelem, kde každá vrstva poskytuje zázemí pro vrstvu vyšší. Nejnižší vrstvou je jádro operačního systému Linux, na kterém je celý model postaven. V této vrstvě jsou implementovány například ovladače všech hardwarových komponent. Nad touto vrstvou se nachází

[1] https://en.wikipedia.org/wiki/Java_Development_Kit
[2] https://developer.android.com/studio/profile/ddms.html
[3] https://developer.android.com/studio/command-line/adb.html

vrstva *Hardware abstraction layer* (zkráceně HAL), která definuje standardizované programové rozhraní pro výrobce hardwarových komponent. Implementace HAL je po přeložení zabalena do dynamické `.so` knihovny a může být systémem Android kdykoliv zavolána.

Nad vrstvou HAL se nachází vrstva se dvěma odlišnými komponenty. První z nich je sada nativních knihoven běžících na operačním systému Linux. Můžeme mezi nimi nalézt například knihovnu pro práci s SQLite databází, openGL/ES nebo třeba SSL. Druhá komponenta v rámci třetí vrstvy se nazývá *Android runtime* (zkráceně ART). Tato vrstva se stará o běh některých systémových a všech uživatelských aplikací. ART je následník Dalviku. ART provádí kód ve formátu *Dalvik Executable format* (zkráceně Dex), který je zapsán dle Dex specifikace. Jelikož Dalvik prováděl kód stejného formátu jako jeho následník ART, mohlo by se zdát, že je zde zaručena úplná zpětná kompatibilita. Nicméně ne všechen kód psaný pro Dalvik funguje bezchybně na ART. Nejvýznamnější problémy jsou zachyceny v dokumentaci [40] v sekci *Verifying App Behavior on the Android Runtime (ART)*.

Čtvrtou vrstvou je pak *Android framework*, který poskytuje rozhraní pro doručení obsahu. Jedná se o správce notifikací, lokace, balíčků nebo třeba samotných oken. Pátou, nejvyšší vrstvou, jsou pak samotné uživatelské aplikace.

5.1.3 Android Studio a vývoj aplikací pro Android Wear

Jelikož je Android Wear z velké části pouhá podmnožina operačního systému Android, je vývoj aplikací pro systém Android Wear velmi podobný vývoji aplikací pro Android. Při vytváření nového projektu stačí zvolit možnost vývoje aplikací pro Wear projekt a Android Studio nám vygeneruje dva projekty pro Android a Android Wear aplikaci. Jedná se o dvě nezávislé Android aplikace, které vyústí ve dva balíčky formátu `.apk`. Instalační balíček pro Android Wear je automaticky zabalen do instalačního balíčku pro Android aplikaci z důvodu automatické instalace aplikace pro chytré hodinky pro instalaci aplikace v přidruženém mobilním zařízení.

5.2 Nativní vývoj Apple aplikací

Společnost Apple nabízí vývojářům vývojové prostředí Xcode (viz sekce Xcode v dokumentaci [36]), které zprostředkovává vývoj aplikací pro všechna Apple zařízení. Jmenovitě se jedná o vývoj pro platformy iOS, watchOS, tvOS a macOS. Xcode je vydáván pouze pro operační systém macOS. Toto prostředí se instaluje skrze aplikaci App Store.

Instalace vývojového prostředí Xcode zahrnuje[1]:

- Xcode IDE – samotná aplikace vývojového prostředí
- Překladač jazyka Objective-C
- Překladač jazyka Swift
- *Instruments analysis tool* – nástroje pro analýzu výkonu a výpočetních nároků testované aplikace (sekce *Instruments Analysis Tool* v dokumentaci [36] pro vývojáře)
- Sadu simulátorů jednotlivých Apple zařízení
- Aktuální verzi všech vývojářských nástrojů (SDK) pro všechny platformy

[1] https://itunes.apple.com/us/app/Xcode/id497799835?ls=1&mt=12

Xcode podporuje vývoj především v programovacích jazycích Swift a Objective-C. Jsou ale podporovány i jazyky C, C++, Java, AppleScript, Python, Ruby a další. Existují i rozšíření pro vývojové prostředí Xcode, které umožní vývoj i pro Mono C#, které bude blíže popsáno v sekci o platformě Xamarin.

5.2.1 Interface Builder

Interface Builder[1] je nástroj pro tvorbu grafických uživatelských rozhraní, který se nachází v rámci aplikace Xcode. Díky tomuto nástroji je vývojáři umožněno tvořit grafické uživatelské rozhraní bez potřeby popisu elementů programovacím jazykem. Každý projekt vytvořený pomocí prostředí Xcode obsahuje jednu nebo několik `storyboard`. Jak je zmíněno v textu výše, jedná se o soubor definující grafické uživatelské rozhraní obrazovek v rámci vyvíjené aplikace. `Storyboard` určuje také vztahy mezi jednotlivými obrazovkami, tedy definici přechodů mezi nimi. Komponenta *Interface Builder* tedy vizualizuje a zobrazuje jednotlivé `storyboard` soubory. Pomocí této komponenty můžeme vidět přechody mezi obrazovkami naší aplikace i s definicí jednotlivých grafických elementů, které se na daných obrazovkách nacházejí.

5.2.2 Swift

Swift [47] [49] je moderní programovací jazyk spravovaný převážně společností Apple. Využívá moderní přístup k otázkám, jakými jsou bezpečnost, výkon a návrhové vzory.

Vývoj tohoto jazyka začal v roce 2010 [47]. Jazyk byl vyvíjen programátory společnosti Apple. V čele projektu byl Chris Lattner[2], známý například vývojem překladače Clang [49]. V roce 2014 byla veřejně publikována první aplikace napsaná v programovacím jazyce Swift. Ke konci roku 2014 přišla první verze Swift 1.0 a ještě téhož roku se dočkala aktualizace na verzi 1.1. V roce 2015 pak přišla verze 2.0, která rozšířila jazyk Swift 2 na operační systém Linux. Swift 2 se následně stal otevřeným jazykem. Programovacímu jazyku Swift se tak otevřela možnost běhu i na jiných zařízeních než Apple. Na podzim roku 2016 pak vyšla verze 3.0.

Swift 3 je aktuálně výchozím programovacím jazykem po založení projektu pro jakékoliv zařízení v programovacím prostředí Xcode, nahradil tím pozici programovacího jazyka Objective-C.

5.2.3 Xcode simulátory

Xcode simulátory jsou popsány v dokumentaci [36] pro vývojáře v sekcích *About Simulator*, *Getting Started in Simulator*, *Interacting with iOS and watchOS* a *Testing and Debugging in Simulator*, ze kterých jsem čerpal informace pro tuto sekci. Oproti ostatním vývojovým technologiím se při ladění Apple aplikací pomocí prostředí Xcode nepoužívají emulátory ale simulátory. Fundamentální rozdíl mezi těmito dvěma pojmy je, že emulátor napodobuje fyzické i programové prostředky emulovaného zařízení. Simulátor napodobuje pouze programové prostředí, nikoliv fyzické.

Sada simulátorů je součástí nástrojů Xcode, jak bylo uvedeno v sekci 5.2. Simulovat můžeme zařízení s operačními systémy watchOS, iOS a tvOS. Jelikož macOS aplikace můžeme pomocí Xcode nástrojů vyvíjet pouze na systému macOS, není třeba simulovat macOS operační systém.

[1] https://developer.apple.com/Xcode/interface-builder/
[2] http://nondot.org/sabre/

Operační systém watchOS nelze simulovat samostatně. Při výběru zařízení které budeme simulovat máme na výběr z mnoha iOS zařízení, ke kterým se dají přidružit konkrétní hodinky. Při spuštění simulace aplikace pro chytré hodinky se tak spustí simulátor iOS zařízení a simulátor watchOS zařízení přidružený k iOS simulátoru.

5.3 Xamarin

Xamarin je platforma pro vývoj mobilních aplikací. Využívá ke své činnosti Mono [44], které bude popsáno níže. Poprvé jej představil Miguel de Icaza v květnu roku 2011 [10]. Xamarin je možné využít pro vývoj nativních Android, iOS a Windows aplikací pod platformou .NET.

5.3.1 Historie

Historie platformy Xamarin je podrobně rozepsána v článku autora Lex Li [16]. V roce 2011 byl v rámci týmu Mono spuštěn projekt Xamarin. Miguel de Icaza, zakladatel platformy Xamarin, jmenoval nového generálního ředitele Xamarinu téhož roku. Stal se jím Nat Friedman. Roku 2012 Xamarin uvolnil výzkumný projekt XobotOS[1], který měl za cíl použít Mono pro interpretaci kódu místo Dalviku[2]. V roce 2013 byl vydán Xamarin 2.0, který již přejmenoval jeho hlavní komponenty na Xamarin.Android a Xamarin.iOS tak, jak je známe dnes. O rok později byla uvolněna další komponenta s názvem Xamarin.Forms, která umožňuje vytvářet multiplatformní mobilní aplikace. Až do roku 2016 pokračoval vývoj Xamarinu dále jako uzavřený a zpoplatněný projekt. V roce 2016 přišla změna v podobě odkoupení platformy firmou Microsoft. Xamarin je od té doby otevřeným softwarem a je vydáván pod MIT[3] licencí.

5.3.2 Komponenty

Xamarin nabízí čtyři základní komponenty pro vývoj nativních aplikací [50].

Xamarin.iOS je komponenta, pomocí které můžeme vyvíjet nativní aplikace pro mobilní telefony s operačním systémem iOS. Xamarin přináší vývojářům možnost použít jakékoliv aplikační rozhraní z iOS SDK. Dále také v omezené míře dovoluje vytvářet `WatchKit` aplikace. V rámci této komponenty můžeme volat kód psaný v jazyce Objective-C[4]. Pokud vyvíjíme na operačním systému Microsoft Windows, musíme mít alespoň vzdálený přístup k aktivnímu zařízení s macOS pro sestavení a ladění aplikace.

Xamarin.Android je komponenta, díky které můžeme vytvářet nativní aplikace pro zařízení s operačním systémem Android nebo Android Wear. Vývojáři mohou přistupovat k plnohodnotnému aplikačnímu rozhraní pro Android i Android Wear operační systém. Vytvořené aplikace jsou automaticky kompilovány do APK[5] balíčku, tedy připravené k instalaci nebo nahrání na Google Play (viz sekce Google Play v dokumentaci [40]).

[1] https://blog.xamarin.com/android-in-c-sharp/
[2] https://cs.wikipedia.org/wiki/Dalvik_(software)
[3] https://opensource.org/licenses/MIT
[4] https://en.wikipedia.org/wiki/Objective-C
[5] https://en.wikipedia.org/wiki/Android_application_package

Xamarin.Forms je komponenta sloužící k vytváření nativních aplikací pro iOS, Android i Windows (Windows Phone a UWP) včetně společného kódu pro grafické uživatelské rozhraní. Vzhled i funkci aplikace tedy definujeme pouze jednou. Xamarin.Forms nám zajistí vytvoření nativního vzhledu pro každou platformu zvlášť.

Xamarin.Mac je komponenta, pomocí které můžeme vyvíjet aplikace pro macOS. Pokud vyvíjíme na operačním systému Microsoft Windows, můžeme projekt pouze vytvořit, případně přeložit. Sestavení a ladění zde již není podporováno.

5.3.3 Mono

Mono [44] je implementace platformy .NET s otevřeným softwarem, která vznikla na základě ECMA norem pro jazyk C#[1]. Mono vzniklo v roce 2001 a až do roku 2016 bylo vyvíjeno nejdříve společností Novel, později společností Xamarin (viz sekce *History* v dokumentaci [44]). V roce 2016 se připojilo v rámci Xamarinu k .NET Foundation[2]. Rozrůstající se rodina aplikací využívajících Mono a aktivní přístup komunity pomáhá dostat Mono do pozice nejčastěji zvolené platformy pro budování multiplatformních aplikací [44].

Jak je popsáno v dokumentaci (sekce *About Mono* v dokumentaci [44]), Mono se skládá ze čtyřech hlavních komponent:

Překladač jazyka C# umožňuje překlad jazyka C# verze 1.0, 2.0, 3.0, 4.0, 5.0 a 6.0. Podrobnější popis jednotlivých verzí je možné najít na Wikipedii [49] v sekci *C Sharp (programming language)*.

Mono Runtime je implementace virtuálního stroje dle standardu *Common Language Infrastructure* [11]. Tato implementace poskytuje mnoho nástrojů, mezi které patří například nástroj pro připojování knihoven, *garbage collector*[3] a systém pro zpracování vláken. Dále je podporováno *Just-in-Time* (JIT) a *Ahead-of-Time* (AOT) překládání pro jednodušší vývoj aplikací. Více informací je možné nalézt v sekci *The Mono Runtime* v dokumentaci [44].

.NET Framework Class Library je komplexní sada tříd pro vývoj aplikací. Tyto třídy jsou kompatibilní s .NET knihovnami společnosti Microsoft.

Mono Class Library je sada tříd, která implementuje další funkcionalitu mimo rámec .NET knihoven společnosti Microsoft. Jedná se především o užitečnou funkcionalitu při vývoji pro operační systémy Linux. Jsou jimi například třídy pro práci s LDAP, Gtk+, openGL, Cairo, POSIX a mnoho dalších.

Jak již bylo zmíněno, Mono slouží pro vývoj multiplatformních aplikací platformy .NET. Mono je kompatibilní se systémy Linux, macOS, watchOS, BSD, Microsoft Windows a dalšími. Jsou podporovány architektury x86, x86-64, ARM, s390, PowerPC a další. Celý výčet kompatibility systémů a architektur je možné nalézt v sekci *Supported Platforms* v dokumentaci [44]. Hlavní Mono C# překladač má název `mcs`. Tento překladač podporuje různá .NET rozšíření jakým je například LINQ[4] nebo dynamický datový typ `dynamic`. V rámci Mono můžeme použít i jiné podporované programovací jazyky. Jsou jimi například Java,

[1] https://www.ecma-international.org/publications/files/ECMA-ST/Ecma-334.pdf
[2] https://dotnetfoundation.org/
[3] https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)
[4] https://en.wikipedia.org/wiki/Language_Integrated_Query

Scala, Python, JavaScript, #Smalltalk nebo také funkcionální jazyk F#. Výčet všech jazyků je možné nalézt v sekci *Languages* v dokumentaci [44].

5.3.4 Vývojová prostředí

Vývoj v Xamarinu je oficiálně podporován na operačních systémech Microsoft Windows a macOS. Veškeré informace obsažené v této sekci je možné dohledat v sekci *System Requirements* v dokumentaci pro vývojáře [50].

Komponenta	**macOS - Xamarin Studio**	**MS Windows - Visual Studio**
Xamarin.iOS	Ano	Ano*
Xamarin.Android	Ano	Ano
Xamarin.Forms	Pouze iOS a Android	Android, Windows, Windows Phone, iOS*
Xamarin.Mac	Ano	Pouze otevřít projekt a kompilovat

* Pouze s lokálně nebo vzdáleně připojeným macOS skrze síť.

Tabulka 5.1: Možnosti vývoje v konkrétních vývojových prostředích (viz sekce *System Requirements* v dokumentaci [50])

V tabulce č. 5.1 můžeme vidět, že oficiálně jsou podporovány pouze dvě vývojová prostředí. Před odkoupení Xamarinu společností Microsoft však bylo pouze právě jedno vývojové prostředí. Tím bylo Xamarin Studio, které bylo distribuováno i pro operační systém Windows. Toto studio je ve verzi pro Windows stále funkční, nicméně nelze jej už oficiální cestou z webových stránek Xamarinu získat.

Požadavky pro vývoj na operačním systému macOS

Pro vývoj skrze platformu Xamarin na operačním systému macOS je doporučeno mít verzi El Capitan nebo Sierra (viz sekce *System Requirements* v dokumentaci [50]).

Komponenta	**Doporučené verze SDK**
Xamarin.iOS	iOS 10 SDK
Xamarin.Android	Aktuální Android API SDK
Xamarin.Mac	macOS El Capitan (10.11) SDK

Tabulka 5.2: Doporučené verze SDK pro vývoj na operačním systémy macOS (viz sekce *System Requirements* v dokumentaci [50])

Xamarin.Forms slučuje doporučení která jsou v tabulce uvedena pro každou komponentu. Vývoj aplikací pro Windows není na macOS pomocí Xamarin.Forms podporován.

Testování a ladění na operačním systému macOS

Xamarin.Mac aplikace mohou být laděny a testovány přímo na hostitelském zařízení (viz sekce *System Requirements* v dokumentaci [50]). Mobilní aplikace mohou být pro ladění a testování do fyzických zařízení šířena skrze USB kabel. watchOS aplikace jsou pak nejprve distribuována do telefonu, který je distribuuje do připojených chytrých hodinek.

Můžeme také využít simulátory a emulátory pro jednotlivé platformy:

- Pro testování a ladění aplikací z rodiny Xamarin.iOS můžeme použít simulátory, které jsou distribuovány v rámci prostředí Xcode. Jedná se o simulátor zařízení iPhone,

iPad, Apple Watch a Apple TV. Kompletní informace a postupy je možné nalézt v dokumentaci [50].

- Pro testování a ladění aplikací pro Android a Android Wear můžeme použít AVD (*Android Virtual Device*). Instalátor Xamarinu nám automaticky předkonfiguruje virtuální zařízení dle verze nainstalovaných Android SDK. Pomocí aplikace *AVD manager* můžeme editovat stávající a vytvářet nová virtuální zařízení. Pro rychlý běh emulátoru se doporučuje použít Intel HAXM[1] (*Intel Hardware Execution Manager*).

Požadavky pro vývoj na operačním systému Microsoft Windows

Pokud chceme vyvíjet pomocí platformy Xamarin na operačním systému Microsoft Windows, potřebujeme Windows 7 nebo vyšší (viz sekce *System Requirements* v dokumentaci [50]). Pro Xamarin.Forms je vyžadována verze Windows 8.1 a vyšší. Pokud chceme v rámci Xamarin.Forms použít UWP, budeme potřebovat Windows 10. Dále bude zapotřebí Visual Studio 2013 Community nebo vyšší edice. Pro Visual Studio 2015 již toto neplatí, tedy může být použita i Professional verze.

- Xamarin.iOS – Pro vývoj aplikací rodiny Xamarin.iOS je potřeba ze sítě dostupné zařízení s nainstalovaným macOS. K tomuto zařízení se pak Visual Studio připojí a skrze něj bude mít přístup k příslušným SDK. Dále je zapotřebí mít nainstalované Visual Studio 2013 nebo novější.
- Xamarin.Android – Vývoj Android a Android Wear aplikací v rámci rodiny Xamarin.Android má stejné požadavky jako v případě vývoje na operačním systému macOS. Doporučeno je použít aktuální Android API, které je zpětně kompatibilní s předchozími verzemi.
- Xamarin.Mac projekt může být v programu Visual Studio pouze otevřen a přeložen. Sestavení macOS aplikace není na Microsoft Windows možné.
- Xamarin.Forms slučuje požadavky a doporučení výše vypsaných komponent. Přidává však možnost vývoje pro Windows Phone a UWP. Pro UWP aplikace potřebujeme Windows 10 SDK, které můžeme zahrnout v instalaci Visual Studia, případně dodatečně stáhnout ze stránek Microsoftu[2].

Testování a ladění na operačním systému Microsoft Windows

Mobilní aplikace mohou být pro ladění a testování do fyzických zařízení šířeny stejně jako při vývoji na operačním systému macOS, tedy skrze USB kabel (viz sekce *System Requirements* v dokumentaci [50]). To ovšem neplatí v případě, pokud chceme testovat a ladit aplikace rodiny Xamarin.iOS. V tom případě musíme zařízení připojit do zařízení se systémem macOS, nikoliv do zařízení, kde je spuštěna instance programu Visual Studio.

Simulátory a emulátory pro jednotlivé platformy:

- Nejjednodušší způsob ladění a testování aplikací z rodiny Xamarin.iOS je spuštění simulátoru na počítači, ke kterému jsme v rámci Visual Studia po síti připojeni. Simulátory jsou přístupné na připojeném macOS zařízení i během ladění ve Visual Studiu.

[1] https://software.intel.com/en-us/android/articles/intel-hardware-accelerated-execution-manager

[2] https://developer.microsoft.com/windows/downloads/windows-10-sdk

- Pro testování a ladění Android a Android Wear zařízení můžeme použít AVD stejným způsobem jako na macOS. Nicméně alternativních emulátorů pro Windows můžeme nalézt více. Patří mezi ně například emulátor Genymotion[1] nebo emulátor od firmy Microsoft, který je šířen v rámci instalátoru aplikace Visual Studio 2015.
- Xamarin.Forms aplikace pro platformy iOS a Android se mohou ladit a testovat jak je psáno ve výše uvedených případech. Windows Phone aplikace pak můžeme testovat pomocí emulátorů od firmy Microsoft. UWP nebo Windows 8.1 aplikace můžeme testovat přímo na hostitelském zařízení.

5.3.5 Xamarin a nositelná zařízení

V následujících podkapitolách si shrneme informace psané výše a doplníme je o informace ohledně vývoje aplikací pro chytré hodinky.

watchOS

Jak již bylo zmíněno v sekci 5.1.2, komponenta Xamarin.iOS umožňuje vývoj `WatchKit` aplikací. Aktuálně je možné vyvíjet aplikace pouze pro watchOS verze 1 a 2 (sekce *Introduction to watchOS* v dokumentaci [50]). Vývoj aplikací pro watchOS 3 je zatím stále v *preview* stádiu. Je tedy umožněno vyvíjet aplikace i pro watchOS 3, ovšem *preview* verze aplikačního rozhraní není vhodná pro stabilní aplikace. Dle zveřejněných informací (sekce *Introduction to watchOS 3* v dokumentaci [50]) nemusí být aplikace psané pro *preview* watchOS3 kompatibilní s budoucí stabilní verzí tohoto rozhraní. Slouží tedy pouze pro otestování nových vlastností a možností, které watchOS 3 nabízí. Jedná se například o zpracování úloh na pozadí například pomocí třídy `WKApplicationRefreshBackgroundTask` a dalších. Simulátory hodinek s operačním systémem watchOS poskytuje prostředí Xcode. V rámci těchto simulátorů můžeme testovat a ladit aplikace pro různé verze watchOS. Jelikož jsou aplikace pro chytré hodinky často nezbytně svázány s aplikací v mobilním telefonu, nelze simulovat samostatné chytré hodinky, nicméně vždy je simulován i mobilní telefon, který je s hodinkami virtuálně spárován a nabízí tak veškerou funkcionalitu třídy `WCSession`.

Android Wear

Jelikož je systém Android Wear z největší míry pouhá podmnožina systému Android, Xamarin podporuje systém Android Wear v celé jeho míře. Pokud máme komponentu Xamarin.Android ve verzi 5 nebo vyšší, automaticky můžeme vyvíjet i Android Wear aplikace. Do vytvořeného projektu pak postačí pouze vložit NuGet[2] balíček pro doplnění uživatelských prvků specifických pro chytré hodinky. Jedná se o NuGet balíček *Xamarin Android Wear Support Libraries*[3]. Použití Xamarinu tedy nevede k žádnému oslabení použitelných aplikačních rozhraní oproti vývoji v Android Studiu.

5.4 Nativní vývoj Tizen aplikací

Vývojářské technologie platformy Tizen (Tizen SDK) jsou komplexní sadou nástrojů pro vývoj, ladění a testování Tizen aplikací. V této sekci se zaměříme na prerekvizity potřebné

[1] https://www.genymotion.com/
[2] https://www.nuget.org/
[3] XamarinAndroidWearSupportLibraries

pro fungování Tizen SDK nástrojů, minimální nároky pro vývoj aplikací a především na rozdělení vývoje do dvou hlavních kategorií, které nám Tizen SDK umožňuje. Jedná se o vývoj aplikací založených na webových technologiích a vývoj nativních aplikací. Informace budou čerpány především z knihy *Professional Tizen Application Developer* [12] a z dokumentace [48] pro vývojáře.

5.4.1 Prerekvizity a systémové požadavky

Operační systém	Verze systému	Bitová verze
Ubuntu	14.04/12.04	32/64
MS Windows	7/8/10	32/64
macOS	10.8/10.9/10.10/10.11	64

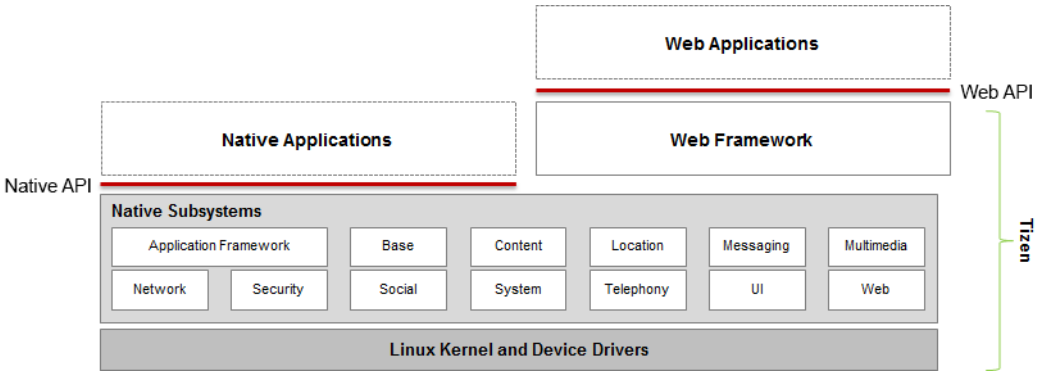
Tabulka 5.3: Podporované operační systémy pro Tizen SDK (viz sekce *Prerequisites for the Tizen SDK* v dokumentaci [48])

V tabulce č. 5.3 můžeme zjistit, na jakých operačních systémech můžeme Tizen aplikace vyvíjet. Mezi nároky na výpočetní výkon a dispozice počítače patří minimálně dvoujádrový procesor o frekvencích alespoň 2 GHz. Dále pak alespoň 3 GB hlavní paměti a alespoň 6 GB volného místa v úložišti. Posledním nezbytným požadavkem je pak mít nainstalovaný *Java Development Kit* (JDK). Další požadavky, například pro zvýšení rychlosti emulátoru, je možné nalézt v dokumentaci [48].

5.4.2 Tizen Studio

Tizen Studio je tvořeno ze samotného vývojového prostředí (IDE), emulátorů, správce balíčků, ukázkových zdrojových kódů a dokumentace. Ve správci balíčků *Tizen Studio Package Manager* pak můžeme stáhnout a instalovat nástroje pro konkrétní platformy včetně jejich různých verzí. Samotné vývojové prostředí Tizen Studio je založeno na známém vývojovém prostředí Eclipse[1]. Eclipse je vývojové prostředí s otevřeným softwarem, které je možné použít pro spoustu programovacích jazyků.

5.4.3 Typy Tizen aplikací



Obrázek 5.2: Hierarchie platformy Tizen (viz sekce *Tizen Application Types* v dokumentaci [48])

[1] https://eclipse.org/

Na obrázku č. 5.2 můžeme vidět hierarchii platformy Tizen. Na spodní vrstvě leží jádro operačního systémy Linux. Nad ním běží nativní subsystém, který se stará o práci se sítí, systémem, zprávy, multimédii a mnoho dalších věcí. V operačním systému Tizen je nad nativním subsystémem už pouze webové prostředí (*Web Framework*), které zajišťuje chod a vykreslení aplikací běžících nad ním.

Vývojář může využít jedno ze dvou aplikačních rozhraní, dle toho jakou aplikaci chce vytvořit. Nativní aplikační rozhraní (`Native API`) slouží pro vývoj nativních aplikací, které běží přímo nad subsystémem Tizen. Webové aplikační rozhraní (`Web API`) pak slouží pro vývoj webových aplikací, které běží ve webovém prostředí operačního systému Tizen. Oba principy budou detailněji popsány níže.

5.4.4 Vývoj nativních aplikací

Nativní aplikace jsou vyvíjeny pomocí jazyků C a C++. Tyto aplikace obvykle mohou pomocí aplikačního rozhraní přistupovat k více pokročilým funkcím systému (viz sekce *Tizen Application Types* v dokumentaci [48]). Jedná se například o některé specifické funkce při práci s kamerou, GPS, akcelerometrem a dalšími zařízeními.

Jak již bylo zmíněno, nativní aplikace využívají aplikačního rozhraní `Native API`, které přináší veškeré výhody při práci s pamětí a výkonností při vytváření aplikací v jazyku C pro Linux. Možnosti nativního aplikačního rozhraní jsou velmi široké a dokáží tak pokrýt široké spektrum možností, jaké Tizen zařízení přinášejí. Rozhraní přináší nespočet dalších rozhraní pro práci s fyzickými komponenty, které můžeme v chytrých hodinkách, telefonech a televizích nalézt.

Při vytváření nativní aplikace máme na výběr ze dvou možností [12] [48].

Grafické aplikace jsou typické aplikace se kterými se uživatelé mohou setkat. Jsou to aplikace, které disponují grafickým uživatelským rozhraním. Jsou spustitelné z hlavního menu, mají svou ikonu a mohou přistupovat ke všem funkcionalitám aplikačního rozhraní. Tizen používá jako hlavní platformu pro práci s grafickým uživatelským rozhraním *Enlightenment Foundation Libraries* (EFL[1]). Jedná se o správce oken, který byl původně zamýšlen pouze pro prostředí X11[2]. Aktuálně se EFL vyvíjí i pro Wayland[3]. EFL je vyvíjeno především pro linuxové prostředí, jsou zde však snahy uživatelů podpořit vývoj i pro systém Windows a macOS.

Služby jsou naopak aplikace, které nemají grafické uživatelské rozhraní. Nenachází se v hlavním menu, lze je nalézt pouze v aplikaci *task switcher*. Tyto aplikace běží na pozadí a mohou se spouštět například periodicky, nebo je může spouštět jiná grafická aplikace. Další možností spuštění je například po dokončení nějakého přenosu dat na pozadí.

Grafické aplikace a služby mohou být zabaleny do jednoho balíčku. Platí zde ovšem pravidlo, že jeden balíček může obsahovat pouze jednu grafickou aplikaci, ale může obsahovat žádnou nebo více služeb. Samotné služby není možné nahrát na Tizen Store[4] ke stáhnutí. Služby se dají kombinovat i s webovými aplikacemi (sekce *Packaging a Hybrid Application* v dokumentaci [48]).

[1] https://www.enlightenment.org/
[2] http://www.opengroup.org/desktop/x/
[3] https://wayland.freedesktop.org/
[4] http://www.tizenstore.com

Internacionalizace nativních aplikací

Základní informace o jazykových možnostech platformy Tizen jsou popsány v sekci 4.3.5. Uvedeme si tedy již konkrétní techniky jak internacionalizovat nativní grafické aplikace. Při internacionalizaci Tizen aplikací můžeme využít následující možnosti (sekce *Internationalization* v dokumentaci [48]).

i18n Aplikační rozhraní `i18n` je implementováno pomocí knihovny ICU[1] (viz sekce `i18n` v dokumentaci [48]). Pomocí tohoto aplikačního rozhraní můžeme formátovat data, čísla a čas dle vhodného formátu pro konkrétní jazyky. Hlavní komponenty aplikačního rozhraní jsou správa řetězců, správa kalendáře a dat, správa čísel a správa časových zón. Více informací o tomto komplexním aplikačním rozhraní lze nalézt v dokumentaci [48] v sekci `i18n`.

Lokalizace zdrojů Pomocí lokalizace zdrojů můžeme vytvářet různé řetězce pro různé jazyky. Při této činnosti je využíváno `Internationalization API`, které je možné použít pouze na chytrých telefonech a hodinkách (viz sekce *Resource Localization* v dokumentaci [48]). Toto aplikační rozhraní ukládá řetězce pro různé jazyky do `.po` (*portable object*) souborů. Tyto soubory jsou vytvářeny a upravovány pomocí editoru PO souborů (*PO file editor*), což je komponenta dostupná v Tizen Studio aplikaci. Dále jsou `.po` soubory kompilovány do `.mo` souborů. Aplikace poté dle jazyka a regionu nastaveném v konfiguraci operačního systému načte vhodný soubor zdrojů. Pokud není tento zdroj nalezen, použije se výchozí soubor zdrojů. Při vytváření `.po` souborů je pro každý řetězec zadán klíč `msgid`. Pomocí tohoto klíče můžeme daný řetězec vypsat funkcí `_()`, jejímž prvním a jediným argumentem je právě `msgid`.

Správa telefonních čísel Tizen aplikace umožňují formátovat telefonní čísla dle jazyka a regionu nastaveném v operačním systému zařízení (viz sekce *Phone Number Management* v dokumentaci [48]). Pro tyto účely je využívána knihovna `libphonenumber`[2]. Pomocí této knihovny jsme schopni dle zadaného telefonního čísla získat informace o regionu, odkud číslo pochází. Můžeme také formátovat zadané telefonní číslo podle konkrétního regionu zadané argumentem do příslušné funkce. V poslední řadě můžeme normalizovat telefonní číslo do standardního mezinárodního tvaru.

Jak již bylo avizováno v kapitole 4.3.5, operační systém Tizen podporuje i jazyky psané zprava doleva. Převracení uživatelského rozhraní je však v rukou vývojáře. Jelikož jsme schopni v aplikaci zjistit jaký jazyk je v systému nakonfigurován, můžeme na základě toho vykreslit specifické uživatelské rozhraní. Je možné i detekovat změnu jazyka a regionu za běhu aplikace a zareagovat na ni ve vyvíjené aplikaci.

5.4.5 Vývoj aplikací založených na webových technologiích

Aplikace založené na webových technologiích (web aplikace) jsou v podstatě webovou stránkou uloženou v zařízení. Jedná se tedy o kód v jazycích HTML, CSS a JavaScript. Web aplikace využívá webového prostředí (*Web Framework*), které zprostředkovává interakci s nativním subsystémem. Web aplikace využívají webového aplikačního rozhraní *Web API* pro sestavení aplikací pomocí základních webových technologií.

[1] http://site.icu-project.org/
[2] https://github.com/googlei18n/libphonenumber

Program sestavený pomocí webového aplikačního rozhraní obsahuje soubor `index.html`, který je kořenovým souborem pro tvořenou aplikaci. Dále obsahuje oddělené adresáře pro jednotlivé zdroje, jakými mohou být například soubory se zdrojovým kódem v jazyce JavaScript, CSS, nebo také adresáře s obrázky a zvukovými nahrávkami.

Při vytváření web aplikací máme na výběr ze čtyř možností [12] [48] (viz sekce Web *Application Models*):

Grafické aplikace – popsány v kapitole 5.4.4.

Služby – popsány v kapitole 5.4.4.

Widgety (pouze pro nositelná zařízení) jsou specializované aplikace pro rychlé předání informací uživateli z rodičovské aplikace. Widgety mohou umožnit uživateli omezený repertoár operací s aplikací bez spuštění rodičovské aplikace.

Watch aplikace (pouze pro nositelná zařízení) je aplikace, která je spuštěna na hlavní obrazovce chytrých hodinek. Obvykle je tedy jednou z jejích funkcí grafické zobrazení aktuálního času. Dále může uživateli předat aktuální informace z jiných aplikací či služeb.

Internacionalizace web aplikací

Internacionalizace web aplikací probíhá pomocí správce zdrojů (sekce *Localization* v dokumentaci [48]). V adresáři s projektem vytvoříme adresář `locales`, do které přidáme zkratku jazyka, který chceme v naší aplikaci podpořit. Jedná se o zkratku `subtag` dle norem W3C IANA[1]. Do tohoto adresáře pak vložíme `.js` soubor s libovolným názvem. Tento soubor se musí v každém adresáři s konkrétním jazykem jmenovat stejně. Obvykle je pojmenován `language.js`. Obvykle je ještě přidán jeden soubor téhož názvu do výchozího adresáře projektu pro výchozí hodnoty v případě, že jazyk nastavený v daném zařízení není aplikací podporován. Tyto soubory pak obsahují asociativní pole jednotlivých řetězců. Při implementaci aplikace pak stačí vložit tento `.js` soubor do stránek, kde chceme řetězce použít a použít přístup ke konkrétní položce daného pole. Správa zdrojů pak dle nastavení jazyka operačního systému Tizen rozhodne, který soubor řetězců vloží do daného bloku. Obdobně můžeme lokalizovat i obrázky a další zdroje, které v aplikaci používáme. Stačí opět vytvořit soubory stejného názvu a vložit je pod adresář zapouzdřující patřičný jazyk.

Podpora jazyků RTL je popsána v kapitole *Globalization* v dokumentaci [48] systému Tizen. Pro RTL jazyky můžeme psát specifické kaskádové styly a HTML elementy a následně detekovat, zda je aktuální jazyk psaný zprava doleva nebo nikoliv. Zjištění lokalizace můžeme provést například pomocí jQuery knihovny Globalize[2] nebo pomocí systémového volání `tizen.systeminfo.getPropertyValue()` s prvním argumentem `"LOCALE"`. Po zjištění, zda aktuální jazyk je psaný zprava doleva nebo nikoliv můžeme načíst specifické kaskádové styly, javascriptové soubory a jakkoliv aplikaci upravit dle potřeb daného jazyka.

5.5 Shrnutí a porovnání vývojářských technologií

V této kapitole jsou představeny vývojářské technologie pro vývoj aplikací chytrých hodinek. Můžeme zde nalézt informace o vývoji aplikací pro operační systémy Android Wear,

[1] http://www.iana.org/assignments/language-subtag-registry/language-subtag-registry
[2] https://github.com/globalizejs/globalize

watchOS i Tizen. Výše popsaná technologie Xamarin je určitou alternativou k čistě nativnímu vývoji aplikací pro Android Wear a watchOS.

Srovnání vývojářských technologií dle operačních systémů, na kterých mohou být nainstalovány je zobrazeno v následující tabulce.

Vývojářské technologie	**Windows**	**Linux**	**macOS**
Nativní Android technologie	Android Wear	Android Wear	Android Wear
Nativní Apple technologie	–	–	watchOS
Xamarin	Android Wear, watchOS	–	Android Wear, watchOS
Nativní Tizen technologie	Tizen	Tizen	Tizen

Tabulka 5.4: Vývojářské technologie pro aplikace na různých operačních systémech

Pokud chceme vyvíjet aplikaci například pro operační systém watchOS, můžeme využít buď vývojářských technologií Xamarin nebo nativních nástrojů Apple. Pokud je naší další podmínkou vývoj na operačním systému MS Windows, výběr se zužuje pouze na vývojářskou technologii Xamarin.

Kapitola 6

Implementace terminálové aplikace YSoft SafeQ

Obsahem této kapitoly je popis systému YSoft SafeQ a popis implementace vytvářených terminálových aplikací pro tento systém. Jedná se o aplikace pro systém Android a Android Wear šířené v rámci jednoho instalačního balíčku. V rámci tohoto instalačního balíčku je šířeno i prostředí pro běh aplikace Mono runtime, které je popsáno výše.

6.1 YSoft SafeQ

YSoft SafeQ je komplexní balíček služeb pro centrální správu tisku v rámci celých institucí. Umožňuje spravovat tiskové úlohy pro 2D i 3D tisk včetně možnosti účtování tisku konkrétnímu uživateli, zabezpečení tiskové úlohy pomocí identifikace uživatele přímo u dané tiskárny a může nabídnout mnoho dalšího.

Balík služeb YSoft SafeQ pro určitou instituci vždy obsahuje server, který spravuje takzvané terminály. Jeden terminál odpovídá jedné tiskárně. Uživatel se pomocí některé z terminálových aplikací přihlašuje na příslušný server ke konkrétnímu terminálu. Uživatel se může identifikovat přihlašovacími údaji, čipovou kartou nebo dalšími způsoby, které konkrétní terminál podporuje.

6.1.1 Identifikace tiskárny a příslušného serveru

Každá tiskárna (terminál) má v sobě zakomponován jeden nebo několik způsobů jak zprostředkovat terminálovým aplikacím informaci o tom, kde leží server, který tuto tiskárnu spravuje. Dále také obsahuje jednoznačný identifikátor tiskárny, který ji na daném serveru identifikuje. Terminálová aplikace musí tyto informace znát aby mohla komunikovat s příslušným serverem, který obsluhuje daný terminál. Různé terminálové aplikace získávají tato data různě. Terminálová aplikace YSoft SafeQ pro mobilní zařízení umožňuje zjištění těchto údajů například z QR[1] kódu, který je na dané tiskárně umístěn. Dalším možným způsobem získání těchto dat je pomocí Bluetooth majáků. Pokud tiskárna disponuje nakonfigurovaným Bluetooth majákem, můžeme z něj pomocí komunikační technologie Bluetooth zjistit, na jaké adrese se nachází server obsluhující tuto tiskárnu a jaké identifikační číslo tato tiskárna zastupuje.

[1] https://en.wikipedia.org/wiki/QR_code

Chytré hodinky s operačním systémem Android Wear a watchOS mají omezené možnosti, jakými mohou tyto informace získat. Android Wear aplikace mohou využít zmíněných Bluetooth majáků a skenovat okolí, zda-li se v něm nenachází nějaká tiskárna, která by o sobě prozradila potřebné údaje. Operační systém watchOS už je na tom s možnostmi hůře. Jelikož vývojářům neumožňuje přistoupit ke komunikační technologii NFC ani Bluetooth, není možné pomocí samotných hodinek tiskárny vyhledávat. Zde nám pomůže komunikace s chytrým telefonem, který se může pokusit vyhledávat tiskárny pomocí Bluetooth místo chytrých hodinek. Uživatel terminálové aplikace pak musí při použití chytrých hodinek mít k dispozici i mobilní telefon, který bude v dostatečné vzdálenosti od tiskárny, kterou chce uživatel pomocí aplikace v chytrých hodinkách spravovat.

6.1.2 YSoft SafeQ server

Jedná se o webovou službu implementující architekturu REST[1]. Architektura REST je založena na běžné HTTP komunikaci. Pomocí této komunikace je možné komunikovat se serverem, který spravuje konkrétní tiskárnu. Můžeme tedy zjistit, zda v konkrétní tiskárně má uživatel zadané nějaké tiskové úlohy, jaké možnosti tisku můžeme upravovat a můžeme vytvořit požadavek na vytisknutí tiskové úlohy.

6.2 Použité nástroje a vývojové prostředí

Pro vývoj aplikace nad systémy Android a Android Wear jsem si vybral vývojové prostředí Visual Studio, do kterého jsem nainstaloval zásuvný modul Xamarin. V tomto vývojovém prostředí jsem vytvořil dva projekty, jeden pro Android aplikaci a druhý pro Android Wear aplikaci. Jedinou podmínkou pro zajištění komunikace mezi hodinkami a chytrým telefon pomocí služby `Google Play Services` je stejný název aplikace v projektu pro Android a Android Wear aplikaci (viz sekce *Working with Packaging* v dokumentaci [50]).

Pro ladění a testování aplikace jsem použil mobilní telefon LG Nexus 5 (Android API 23) a AVD emulátor pro chytré hodinky se systémem Android Wear. Pro rychlý běh emulátoru jsem používal obraz systému architektury x86[2] ve spojení s virtualizační technologií HAXM[3]. V pozdějších fázích vývoje jsem využíval zařízení Asus ZenWatch 2, které mi bylo zakoupeno společností Y Soft Corporation.

Při vývoji jsem využil službu Apiary[4]. Apiary je služba pro testování návrhu webových aplikací implementujících rozhraní REST. Pomocí jednoduchého značkovacího jazyka můžeme definovat očekávané dotazy a k nim příslušné odpovědi. Jsme schopni nastavit URL adresu pro příchozí požadavek, jeho typ (GET, POST apod.) a podobu jeho hlavičky. Na takto definovaný požadavek můžeme specifikovat odpověď ve formě hlavičky, těla a stavového kódu[5]. Apiary pouze porovnává textové řetězce a nepodporuje žádné větvení. Pokud je tedy rozeznán konkrétní požadavek, bude na něj zaslána odpověď bez ohledu na datové položky v hlavičce. Výsledná aplikace obsahuje pro testovací účely staticky definované zařízení, které představuje komunikaci se serverem Apiary.

[1] http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
[2] https://en.wikipedia.org/wiki/X86
[3] https://software.intel.com/en-us/android/articles/intel-hardware-accelerated-execution-manager
[4] https://apiary.io/
[5] http://www.restpatterns.org/HTTP_Status_Codes

6.3 Princip činnosti výsledné aplikace

V následujících kapitolách bude uvedeno jakým způsobem probíhá interakce aplikace s uživatelem, okolím i systémem YSoft SafeQ.

6.3.1 Vyhledávání tiskáren

Vyhledávání tiskáren jsem realizoval pomocí vyhledání Bluetooth majáků podporující protokol Eddystone. Využil jsem NuGet balíčku `Estimote SDK for Xamarin.Android`. V Android i Android Wear aplikaci jsem vytvořil aktivitu (viz sekce *Activity Lifecycle* v Xamarin dokumentaci [50]), která skenuje okolí a hledá okolní Bluetooth majáky. Tato aktivita implementuje rozhraní `IServiceReadyCallback`. V rámci tohoto rozhraní je potřeba implementovat metodu `OnServiceReady()`, která je zavolána, jakmile je knihovna `Estimote` připravena ke skenování. Po nalezení alespoň jedné tiskárny je tento seznam zobrazen uživateli.

Zařízení jsou skenována v pravidelných intervalech, které vyvolávají událost. Tato událost je v aktivitě zachycena a je skrze ní předán argument typu `EddystoneEventArgs`, ze kterého můžeme zjistit veškeré informace o nalezených majácích. Každý nalezený maják je realizován objektem typu `Eddystone`. Z tohoto objektu poté můžeme přečíst atribut `url`, ve kterém se nachází identifikace tiskárny (tzv. `TerminalID`), číslo portu, použitý protokol (HTTP/HTTPS) a adresa serveru, který tuto tiskárnu spravuje.

```
Manager.Eddystone += (sender, e) =>
{
    if (e.Eddystones.Count > 0)
    {
        foreach (Eddystone eddystone in e.Eddystones)
        {
            var url = eddystone.Url;
            ...
        }
    }
};
```

Kód 6.1: Ukázka obsluhy nalezeného majáku pomocí `Estimote SDK for Xamarin.Android`

6.3.2 Přihlášení uživatele ke konkrétnímu terminálu

Uživatele je možné identifikovat pomocí zadání uživatelského jména a hesla. Jelikož by bylo velmi nepohodlné tyto údaje zadávat v aplikaci pro chytré hodinky, jsou tyto údaje ukládány a čerpány z mobilní aplikace. Pokud se tedy uživatel přihlásí k danému terminálu v mobilní aplikaci, jsou jeho údaje uloženy pro pozdější automatické přihlášení. Tyto uložené údaje využívají i chytré hodinky. Pokud se uživatel skrze chytré hodinky k danému terminálu ještě nepřihlašoval, je skrze text na obrazovce hodinek požádán o zadání přihlašovacího jména a hesla v mobilní aplikaci. Na telefonním zařízení je mu automaticky spuštěna aktivita pro přihlášení. Po úspěšném přihlášení je hodinkám zaslána zpráva skrze `MessageAPI` a hodinky tak uživateli již zobrazí úlohy, které čekají na vytisknutí. Uživatelské jméno a heslo je v zařízení bezpečně uloženo pomocí modulu `Xamarin.Auth`[1].

[1] https://components.xamarin.com/view/xamarin.auth/

6.3.3 Komunikace mezi chytrými hodinkami a telefonem

Jelikož operační systém Android Wear 1 neumožňuje přímou komunikaci se zařízeními v síti internet, musí za něj tuto službu zprostředkovávat aplikace v mobilním telefonu. V aplikaci pro mobilní telefon tedy byla implementována služba (viz sekce *Services* v dokumentaci [40]), která se stará o veškerou komunikaci se serverem YSoft SafeQ. Tato služba se spouští při prvním startu aplikace pro mobilní telefon a od té doby běží nepřetržitě. Pokud by se operační systém Android rozhodl z důvodů přetížení technických prostředků tuto službu zastavit, je později znovu spuštěna a je tak znova připravena přijímat od aplikace Android Wear požadavky a obsluhovat je.

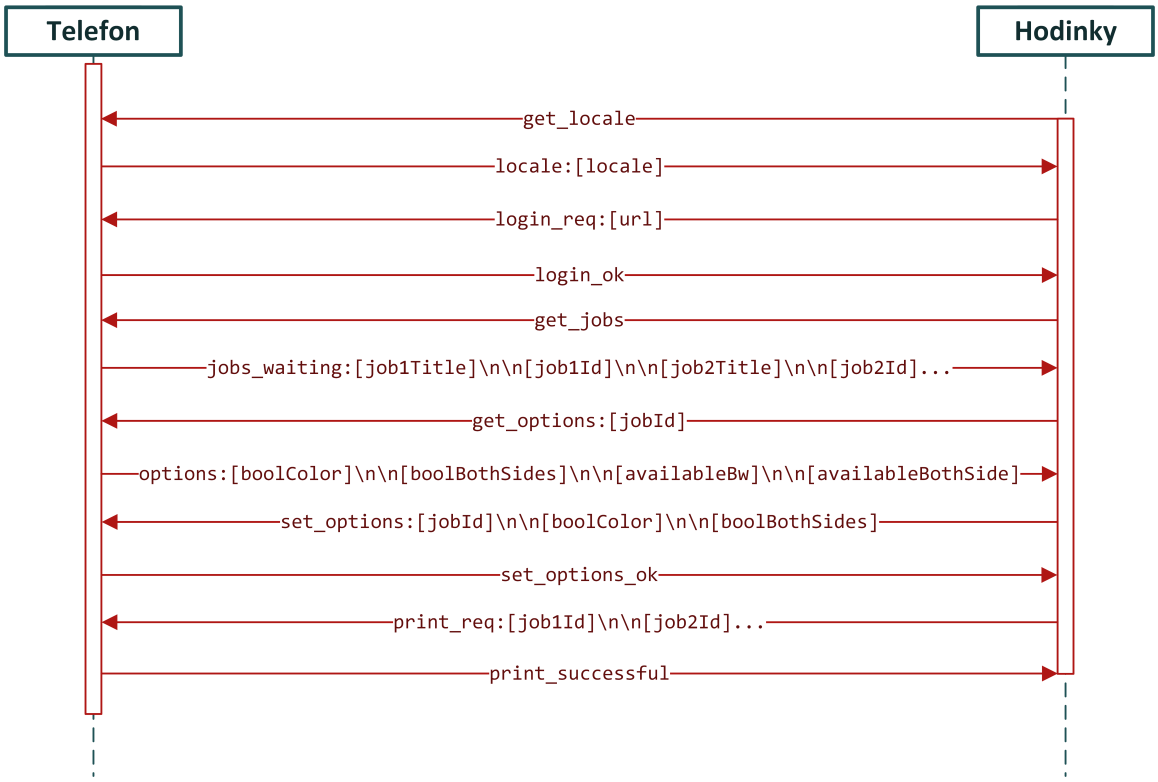
Komunikaci mezi telefonem a hodinkami provádí třída `Communicator`, která implementuje rozhraní `IMessageApiMessageListener` a `IDataApiDataListener`. V rámci těchto dvou rozhraní je nutné implementovat metody `OnMessageReceived(IMessageEvent)` a `OnDataChanged(DataEventBuffer)`. Tyto metody pak implementují chování po příchodu zprávy nebo změny dat skrze `The Wearable Data Layer API`. Prvně tedy vytvoříme instanci třídy `GoogleApiClient`, která je hlavním bodem pro začátek interakce se službou `Google Play Services`.

```
private readonly GoogleApiClient _client new GoogleApiClient.Builder(context).AddApi
    (WearableClass.API).Build();
WearableClass.MessageApi.AddListener(_client, this);
...
public void SendMessage(string message)
{
    foreach (var node in Nodes())
        var result = WearableClass.MessageApi.SendMessage(_client, node.Id, Path,
            Encoding.Default.GetBytes(message)).Await();
}
private IList<INode> Nodes()
{
    return WearableClass.NodeApi.GetConnectedNodes(_client).Await().JavaCast<
        INodeApiGetConnectedNodesResult>().Nodes;
}
public void OnMessageReceived(IMessageEvent messageEvent)
{
    MessageReceived(message);
}
public event Action<string> MessageReceived = delegate { };
...
```

Kód 6.2: Ukázka implementace komunikace pomocí `MessageAPI` a `NodeAPI`

Pomocí metody `AddListener()` v rozhraní `MessageAPI` označíme třídu, která bude zprávy přijímat. Obdobně pak pro `DataAPI`. Následně můžeme implementovat metody pro odeslání zpráv skrze `MessageAPI` nebo datových objektů pomocí `DataAPI`. Pokud odesíláme zprávu pomocí `MessageAPI`, musíme definovat příjemce. Seznam klientů, připojených k `The Wearable Data Layer API`, získáme z rozhraní `NodeAPI`. Pro potřeby terminálové aplikace YSoft SafeQ jsou cílem všechna zařízení, která jsou k dané síti uživatele připojena. Veškeré tyto úkony jsou popsány v dokumentaci projektu Xamarin [50], případně podrobněji v dokumentaci systému Android [40]. Takto univerzálně vytvořenou třídu můžeme použít jak v Android tak Android Wear aplikaci.

Jelikož implementovaná aplikace nebude posílat žádné objemné položky, u kterých by mohlo hrát roli zpomalení chodu aplikace a zbytečné mrhání kapacity baterie hodinek na přenos těchto položek, vystačíme si pouze s rozhraním `MessageAPI` a `NodeAPI`. V komunikační třídě ve výsledné aplikaci je rozhraní `DataAPI` implementováno pro případné pozdější využití.



Obrázek 6.1: Sekvenční diagram komunikace mezi hodinkami a telefonem v aplikaci YSoft SafeQ

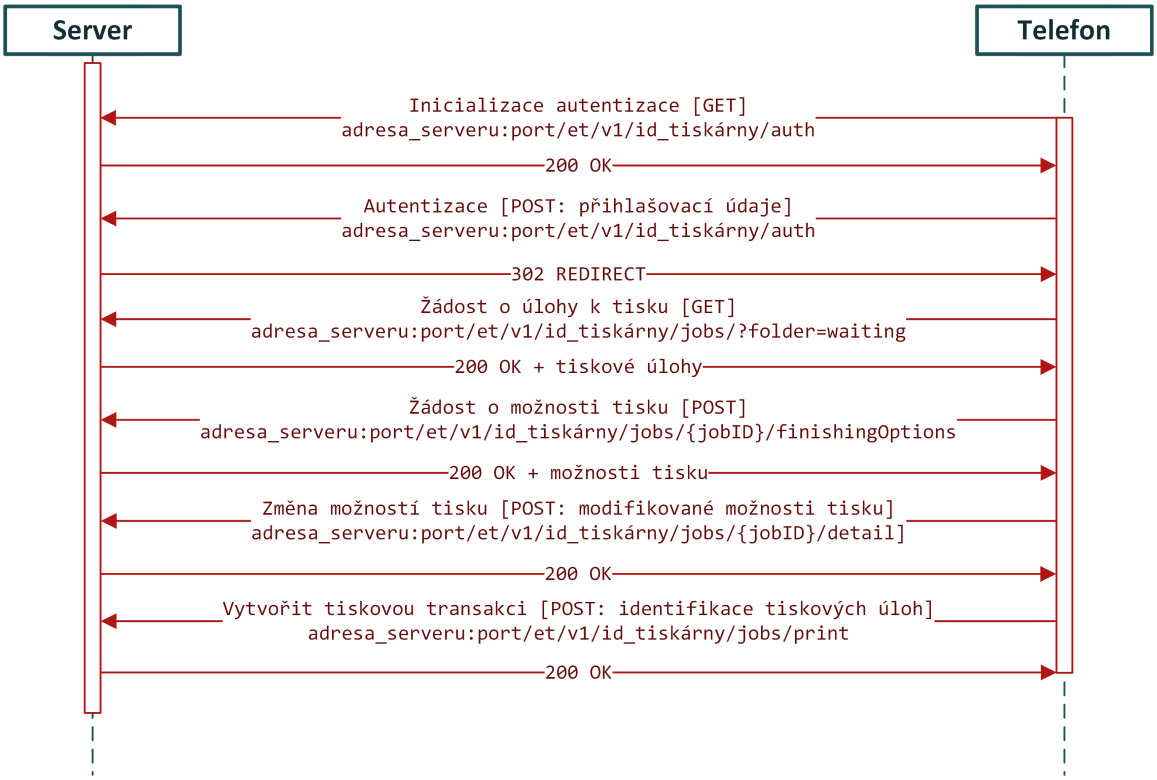
Na obrázku č. 6.1 můžeme vidět, jak je ve výsledné aplikaci implementována komunikace mezi telefonem a hodinkami. Vzhledem k velmi malé množině jazyků podporovaných jednotlivými zařízeními se není možné spolehnout, že správce zdrojů systému Android Wear nastaví požadovaný jazyk aplikace. Chytré hodinky si tedy neprodleně po spuštění aplikace zjistí jaký jazyk je nastaven v chytrém telefonu. Jakmile dostanou odpověď, překonfigurují spuštěnou aplikaci tak, aby odpovídala jazyku nastaveném v telefonu.

Následně po vybrání konkrétní tiskárny zašle aplikace v chytrých hodinkách do telefonu žádost o přihlášení. Ten se pomocí uložených údajů pokusí o přihlášení. Pokud se mu to podaří, zašle hodinkám zprávu o úspěšném přihlášení. Pokud se mu to nepodaří, vyvolá okno aplikace s aktivitou pro přihlášení. Jakmile hodinky obdrží správu o úspěšném přihlášení, vyvolají aktivitu pro zobrazení tiskových úloh. Tato aktivita zašle zprávu telefonu, který stáhne údaje o tiskových úlohách a zašle je zpět hodinkám. Hodinky tyto údaje zobrazí. Po vybrání konkrétní tiskové úlohy se zašle telefonu požadavek o zjištění možnostech tisku dané úlohy. Telefon tedy od serveru zjistí, zda tiskárna umožňuje černobílý tisk a zda umožňuje tisknout oboustranně. Tyto informace jsou zaslány zpět hodinkám pro zobrazení. Pokud je některá z těchto možností v tiskárně dostupná, uživatel ji může měnit.

Poté klikne na tlačítko tisku, které vyšle požadavek na aktualizaci možností tisku konkrétní tiskové úlohy. Následně je zaslána zpráva mobilnímu telefonu ohledně vytvoření tiskové transakce. Jakmile telefon dostane ze serveru odpověď, zašle ji hodinkám a ty uživatele informují, zda se povedlo vytvořit transakci pro tisk daného dokumentu nebo nikoliv.

6.3.4 Komunikace mobilní aplikace se serverem YSoft SafeQ

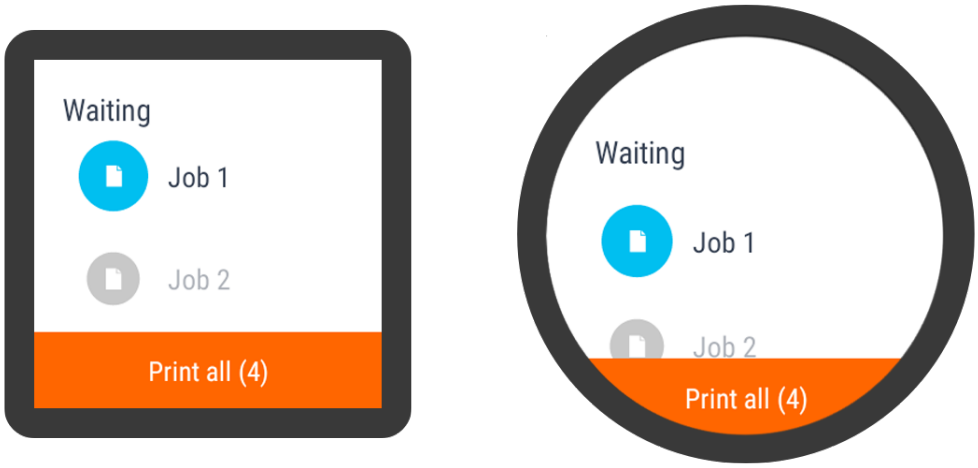
V průběhu běhu aplikace je potřeba zasílat dotazy na server YSoft SafeQ, který spravuje konkrétní tiskárnu. Pro tyto účely jsem využil aplikační rozhraní implementované společností Y Soft Corporation. Aplikační rozhraní můžeme nalézt v přiložených zdrojových souborech v adresáři `SqtaApi`. Toto rozhraní využívá návrhový vzor IoC (*Inversion of control* [7]). Dle tohoto návrhového vzoru nevytváříme instance třídy, kterou chceme využít, ale jsou do třídy dodány tzv. zvenku. Uvolníme tak pevné vazby mezi třídami a zjednodušíme si objektový návrh vytvářené aplikace. Komunikace se vzdáleným YSoft SafeQ serverem se odehrává pomocí třídy `System.Net.Http.HttpClient`, jejíž instance je vytvořena v třídě `RestClient` ve zdrojových souborech aplikačního rozhraní. Pomocí této instance vytváříme HTTP požadavky na vzdálený YSoft SafeQ server.



Obrázek 6.2: Sekvenční diagram komunikace mezi mobilním telefonem a serverem YSoft SafeQ

Na obrázku 6.2 vidíme ideální případ komunikace se serverem YSoft SafeQ. Komunikace je založena na jednoduchých HTML požadavcích. Vyskytují se zde pouze požadavky typu GET a požadavky typu POST.

6.4 Vzhled aplikace



Obrázek 6.3: Ukázka vzhledu výsledné aplikace

Vzhled celé aplikace je přiložen v příloze na obrázcích B.1 a B.2. Je tvořen nativními elementy, které jsou poskytovány systémy Android a Android Wear. Při tvoření vzhledu aplikace byl brán zřetel na *Material Design* [42], odkud byly čerpány veškeré ikony.

Jelikož se reálná zařízení se systémem Android Wear vyrábějí jak s kulatým tak s obdélníkovým displejem, byl navržen vzhled aplikace pro každý typ zvlášť. V aplikaci je implementována funkcionalita, která na základě tvaru displeje hodinek použije patřičné rozložení grafických elementů. Na obrázku č. 6.3 můžeme vidět mírné odlišnosti v rozmístění elementů na obrazovce u obou typů displejů.

6.5 Struktura kódu aplikace

Aplikace, tvořená ve vývojovém prostředí Visual Studio, obsahuje čtyři projekty. Jeden projekt obsahuje zdrojové soubory aplikace pro systém Android, druhý obsahuje zdrojové soubory pro systém Android Wear, třetí obsahuje komunikační rozhraní se systémem YSoft SafeQ dodané společností Y Soft Corporation a čtvrtý obsahuje implementaci společných tříd pro Android i Android Wear operační systém.

Projekt pro operační systém Android se nachází v adresáři `YSoftTerminal`. V jednotlivých podadresářích pak můžeme nalézt definici tříd, které reprezentují aktivity, adaptéry a další komponenty potřebné pro běh a funkčnost aplikace. V podadresáři `Resources` pak nalezneme definici vzhledů aktivit prostřednictvím XAML[1] souborů. Dále zde můžeme nalézt veškeré ikony a řetězce pro jazykové mutace vytvářené aplikace. Můžeme zde nalézt i třídu implementující službu pro komunikaci mezi oběma zařízeními i v případě, že je aplikace v mobilním telefonu vypnuta. Tato třída je implementována v souboru `DataMessageService.cs` v podadresáři `Services`.

[1] https://msdn.microsoft.com/en-us/library/cc295302.aspx

V adresáři `YSoftTerminalWear` můžeme nalézt veškeré soubory týkající se aplikace pro chytré hodinky. Struktura adresáře je velmi obdobná adresáři aplikace pro systém Android.

Implementaci aplikačního rozhraní pro komunikaci se serverem YSoft SafeQ je možné nalézt v adresáři `SqtaApi`. Tento adresář obsahuje velké množství tříd, které slouží především pro serializaci a deserializaci dat, které jsou zasílány a přijímány mezi mobilní aplikací a serverem YSoft SafeQ. Pro komunikaci je využito zápisu ve formátu JSON, který je pomocí těchto modelů převáděn buď ze C# objektů do formátu JSON, nebo naopak.

V posledním adresáři s názvem `SharedClasses` nalezneme podadresáře s definicí tříd, které využívám v Android i Android Wear projektech. Tyto třídy bohužel nemohou být uloženy v jednom z výše vypsaných projektů, jelikož při snaze o vytvoření aplikačního balíčku nejsou jejich jmenné prostory navzájem viditelné.

6.6 Zhodnocení aplikace

V této kapitole se nachází zhodnocení vytvořené aplikace, porovnání se stávajícími řešeními a návrh vylepšení vytvořené terminálové aplikace.

6.6.1 Doba překladu a testování

Doba překladu a generování instalačních balíčků ze všech zmíněných projektů trvá zhruba 2 minuty. Takto vygenerované balíčky poté pro testování a ladění dopravíme do mobilního zařízení a chytrých hodinek, což zabere přibližně 4:30 minuty. V součtu tedy zhruba 6 minut a 30 sekund. Tyto údaje byly naměřeny na výpočetním zařízení s procesorem Intel Core i7 4700HQ, SSD diskem a 16 GB RAM. Aplikace byly automatizovaně přenášeny pomocí ADB do zařízení Nexus 5 a Asus ZenWatch 2.

Výsledná aplikace byla otestována na produkčním serveru YSoft SafeQ v rámci instituce Y Soft Corporation. Server YSoft SafeQ bez problémů reagoval na akce zadávané jak z mobilních zařízení, tak z chytrých hodinek. Aplikace byla testována z následujících zařízení: Google Nexus 5, Samsung Galaxy S3 mini, Huawei Watch W1 a Asus ZenWatch 2. Na žádném zařízení nenastaly komplikace.

6.6.2 Navrhované vylepšení terminálové aplikace

Po rozšíření systému Android Wear 2.0 bude možné celou aplikaci velmi zjednodušit. Aplikace již nebude muset všechny své požadavky pro komunikaci se servery YSoft SafeQ delegovat na mobilní zařízení a bude se moci o komunikaci postarat sama. Bohužel, v době psaní toho textu je sice systém Android Wear 2.0 již oficiálně vydán, avšak žádný výrobce chytrých hodinek ještě tuto aktualizaci nerozdistribuoval pro svá zařízení. Dalším vylepšením, které Android Wear 2.0 přináší, je samostatnost instalačního balíčku. Již bude možné instalovat aplikaci pouze pro chytré hodinky z aplikace `Google Play`. Android Wear 2.0 také rozšířil možnosti uživatelských vstupů o plnohodnotnou klávesnici. Uživatel se tedy bude moci přihlašovat přímo z hodinek a aplikace pro chytré hodinky již nebude mít žádný důvod komunikovat s aplikací v mobilním telefonu.

Jednou z dalších otevřených otázek je zpřístupnění komunikačního rozhraní NFC pro vývojáře třetích stran. Pokud by k tomu v rámci aktualizace systému a vývojářských nástrojů došlo, mohli bychom pomocí NFC z konkrétní tiskárny získat informace o tom, který server ji obsluhuje. Tato komunikační technologie by pak tedy doplnila, případně nahradila komunikační technologii Bluetooth a hledání Bluetooth majáků.

6.6.3 Porovnání se stávajícími řešeními

Obsluha tiskáren z nositelných zařízení je myšlenkou, kterou neimplementuje žádný běžně dostupný produkt. Proto zde budu porovnávat stávající řešení pro správu tiskových úloh z různých zařízení pro systém YSoft SafeQ.

Obsluhu systému YSoft SafeQ jsme schopni provést z množiny takzvaných terminálových aplikací. Jedná se například o aplikaci zabudovanou v samotné tiskárně, kde se může uživatel identifikovat například čipovou kartou nebo zadáním unikátní systémem vygenerované posloupnosti čísel, která se nazývá PIN kód. Další možností je mobilní terminálová aplikace dostupná pro operační systémy Android a iOS. Tyto dvě terminálové aplikace se vyznačují především podporou úplné množiny operací, které je možné s tiskárnou a s úlohy v ní provést. Můžeme zde měnit veškeré možnosti tisku, prohlédnout si frontu již vytisknutých tiskových úloh, provést tisk čekajících úloh a další možnosti.

Aplikace pro chytré hodinky, jak již bylo psáno výše, jsou obvykle pouze rozšířením aplikací pro mobilní telefon. Vzhledem k velikosti jejich obrazovek není vhodné, aby obsahovaly veškeré funkce a možnosti, jaké má mobilní telefon či vestavěná aplikace v tiskárně. Proto obsahují pouze ty funkce, které uživatelé provádějí nejčastěji a jejichž obsluha bude trvat krátkou dobu. Konkrétně se tedy jedná pouze o tisk úloh, které jsou ve frontě čekajících úloh, včetně možnosti měnit nejčastěji používané možnosti tisky, kterými jsou černobílý tisk a možnosti vybrat si mezi jednostranným a oboustranným tiskem.

Kapitola 7

Závěr

Cílem bakalářské práce bylo uvést potenciálního vývojáře do problematiky vývojářských nástrojů pro nejpoužívanější operační systémy nositelných zařízení a demonstrovat použití vývojářských nástrojů na vývoji konkrétní aplikace.

V první části práce byl vysvětlen princip nositelné elektroniky a tato zařízení byla rozdělena do několika kategorií. V další části byly probrány známé operační systémy, které se na nositelných zařízeních vyskytují a které jsou zajímavé pro vývojáře z důvodu dostupných vývojových prostředků. Byly zde popsány možnosti a principy internacionalizace aplikací, přístup vývojářů k aplikačním rozhraním pro důležité komunikační metody a byl vysvětlen základní princip distribuce aplikací do těchto systémů. Následující část již popisuje samotné vývojářské nástroje dostupné pro vývoj aplikací pro jednotlivé operační systémy. Bylo zde popsáno jaké vývojové prostředí můžeme využít, jaké prostředky můžeme použít pro ladění a testování vyvíjených aplikací a na závěr byly tyto vývojářské prostředky porovnány dle několika kritérií.

Poslední část práce se týkala implementace terminálové aplikace pro systém YSoft SafeQ. Tato terminálová aplikace byla implementována pro operační systém Android a Android Wear. Obě tyto aplikace zpřístupňují základní možnosti obsluhy systému YSoft SafeQ. Pomocí těchto aplikací můžeme detekovat okolní tiskárny, přihlásit se k nim, zobrazit si tiskové úlohy čekající na zpracování, změnit jim dodatečně některé možnosti tisku a následně tyto tiskové úlohy buď po jedné nebo hromadně vytisknout. Aplikace klade důraz na jednoduché použití uživatelského rozhraní. Byl zde i naznačen potenciální vývoj a vylepšení vyvinuté aplikace poté, jakmile bude operační systém Android Wear 2.0 dostupný na chytrých hodinkách. Vývoj terminálové aplikace YSoft SafeQ pro chytré hodinky bude dále pokračovat a bude rozšířen pro platformu watchOS, tedy pro všechny chytré hodinky společnosti Apple.

Práce se zdá být prospěšná, jelikož na jejím základě vznikl požadavek na další vývoj terminálové aplikace systému YSoft SafeQ pro nositelná zařízení. Práce přináší vývojářům možnost zamyslet se nad rozšířením platforem, které jejich aplikace podporují, i do segmentu nositelností. V neposlední řadě práce demonstruje implementaci aplikace pro chytré hodinky, která dokáže využít komunikačních rozhraní Bluetooth pro komunikaci s aplikací v mobilním telefonu a pro hledání Bluetooth majáků.

Literatura

[1] Araújo, A.: *Virtual Reality (VR) Vs Augmented - Which is Your Favorite Device?* [Online; navštíveno 29.2.2017].
URL http://www.choozurmobile.com/2016/07/augmented-reality-and-VR-headsets.html

[2] Azuma, R. T.: *A Survey of Augmented Reality. Presence: Teleoperators and Virtual Environments*, ročník 6, č. 4, Srpen 1997: s. 355–385, ISSN 1054-7460, doi:10.1162/pres.1997.6.4.355, [Online; navštíveno 2.3.2017].
URL http://dx.doi.org/10.1162/pres.1997.6.4.355

[3] Bouhnick, G.: *A List of All Operating Systems Running on Smartwatches*. Březen 2015, [Online; navštíveno 14.2.2017].
URL http://www.mobilespoon.net/2015/03/a-list-of-all-operating-systems-running.html

[4] Calvo, A.: *The Wearable Data Layer API*. Berkeley, CA: Apress, 2015, ISBN 978-1-4842-0517-4, s. 173–213, doi:10.1007/978-1-4842-0517-4_6.

[5] Endrle, P.: *Zabezpečení standardu 802.11 a jeho možnosti*. 2009, [Online; navštíveno 15.4.2017].
URL http://hdl.handle.net/11012/11071

[6] Firtman, M.: *Getting Started With Wearables: How To Plan, Build And Design*. Říjen 2015, [Online; navštíveno 11.2.2017].
URL https://www.smashingmagazine.com/2015/10/getting-started-wearables-plan-build-design/

[7] Fowler, M.: *Inversion of Control Containers and the Dependency Injection pattern*. Leden 2004, [Online; navštíveno 10.03.2017].
URL https://martinfowler.com/articles/injection.html

[8] Hara, B.: *IEEE 802.11 handbook : a designer's companion*. New York, NY: IEEE, 2005, ISBN 978-0738144498.

[9] Haselsteiner, E.; Breitfu, K.: *Security in Near Field Communication (NFC)*. In *In Workshop on RFID Security*, Citeseer, 2006.

[10] de Icaza, M.: *Announcing Xamarin*. Květen 2011, [Online; navštíveno 2.3.2017].
URL http://tirania.org/blog/archive/2011/May-16.html

[11] International, E.: *Standard ECMA-335 - Common Language Infrastructure (CLI)*. Geneva, Switzerland, páté vydání, Prosinec 2010, ISBN 0-321-15493-2.

[12] Jaygarl, H.; Luo, C.; Choi, E.; aj.: *Professional Tizen Application Development*. Wrox Press, Wiley, 2014, ISBN 9781118809266.

[13] Katz, D.: *Getting Started with Apple WatchKit and WatchOS 2*. Říjen 2015, [Online; navštíveno 24.2.2017].
URL https://www.programmableweb.com/news/getting-started-apple-watchkit-and-watchos-2/how-to/2015/10/12

[14] Kosir, S.: *Wearables in Healthcare*. Duben 2015, [Online; navštíveno 11.2.2017].
URL
https://www.wearable-technologies.com/2015/04/wearables-in-healthcare/

[15] Lamkin, P.: *Smartwatch timeline: The devices that paved the way for the Apple Watch*. Březen 2015, [Online; navštíveno 17.2.2017].
URL https:
//www.wareable.com/smartwatches/smartwatch-timeline-history-watches

[16] Li, L.: *A history of .NET/Mono/Xamarin*. [Online; navštíveno 5.3.2017].
URL http://corefx.strikingly.com/

[17] Meglio, F. D.: *Native RTL support in Android 4.2*. Březen 2013, [Online; navštíveno 24.2.2017].
URL https://android-developers.googleblog.com/2013/03/native-rtl-support-in-android-42.html

[18] Melnychuk, B.: *Nearables Wearables: Connecting Beacons with Smartwatches for Indoor Positioning*. [Online; navštíveno 10.3.2017].
URL http://elekslabs.com/2015/09/nearables-wearables-connecting-beacons-with-smartwatches.html

[19] Minar, N. B.-N. I.; Tarique, M.: *Bluetooth security threats and solutions: a survey*. *International Journal of Distributed and Parallel Systems*, ročník 3, č. 1, 2012: str. 127, [Online; navštíveno 10.03.2017].
URL https:
//pdfs.semanticscholar.org/8872/521819c79505ac20e5da8dd14f8c41eb3f07.pdf

[20] Nadoba, A.: *Developing your first Tizen Galaxy Gear2 App*. [Online; navštíveno 14.3.2017].
URL http:
//blog.scalac.io/2014/07/30/developing-your-first-galaxy-gear-app.html

[21] Page, V.: *Wearable Technology*. [Online; navštíveno 9.2.2017].
URL http://www.investopedia.com/terms/w/wearable-technology.asp

[22] Phillips, B.; Stewart, C.: *Android Programming: The Big Nerd Ranch Guide*. Pearson Education, 2015, ISBN 9780134171500.

[23] Pražák, D.: *watchOS*. Září 2016, [Online; navštíveno 17.2.2017].
URL https://www.letemsvetemapplem.eu/2016/09/13/novinky-watchos-3/

[24] Rawassizadeh, R.; Price, B. A.; Petre, M.: *Wearables: Has the Age of Smartwatches Finally Arrived? Commun. ACM*, ročník 58, č. 1, Prosinec 2014: s. 45–47, ISSN

0001-0782, doi:10.1145/2629633, [Online; navštíveno 18.3.2017].
URL http://doi.acm.org/10.1145/2629633

[25] Rogerson, J.: *Android Wear: everything you need to know*. Leden 2016, [Online; navštíveno 9.2.2017].
URL http://www.techradar.com/news/wearables/google-android-wear-what-you-need-to-know-1235025

[26] Rouse, M.: *augmented reality (AR)*. [Online; navštíveno 24.2.2017].
URL http://whatis.techtarget.com/definition/augmented-reality-AR

[27] Sawh, M.: *Smart clothing: The biggest benefits*. Leden 2017, [Online; navštíveno 1.3.2017].
URL https://www.wareable.com/smart-clothing/smart-clothing-what-are-the-benefits-2016

[28] Sawh, M.: *The best smart clothing: From biometric shirts to contactless payment jackets*. Leden 2017, [Online; navštíveno 1.3.2017].
URL https://www.wareable.com/smart-clothing/best-smart-clothing

[29] Statler, S.; Audenaert, A.; Coombs, J.; aj.: *Beacon Technologies: The Hitchhiker's Guide to the Beacosystem*. Berkely, CA, USA: Apress, první vydání, 2016, ISBN 1484218884, 9781484218884.

[30] Sung, D.: *What is wearable tech? Everything you need to know explained*. Srpen 2015, [Online; navštíveno 9.2.2017].
URL http://www.wareable.com/wearable-tech/what-is-wearable-tech-753

[31] Swider, M.; Peckham, J.: *Apple watchOS 3 and watchOS 3.1 features and updates*. Duben 2017, [Online; navštíveno 10.4.2017].
URL http://www.techradar.com/news/software/operating-systems/apple-watch-os-3-release-date-news-and-rumors-1323096

[32] Tehrani; Kiana; Michael, A.: *Wearable Technology and Wearable Devices: Everything You Need to Know*. Březen 2014, [Online; navštíveno 10.2.2017].
URL http://www.wearabledevices.com/what-is-a-wearable-device/

[33] Theodoropoulos, G.: *Working with Localization in iOS 8 and Xcode 6*. Listopad 2014, [Online; navštíveno 24.2.2017].
URL http://www.appcoda.com/localization-tutorial-ios8/

[34] Vivo, M. V.: *Android Wear: Accessing the Data Layer API*. Září 2015, [Online; navštíveno 11.3.2017].
URL https://medium.com/@manuelvicnt/android-wear-accessing-the-data-layer-api-d64fd55982e3

[35] *Android Open Source Project*. [Online; navštíveno 1.3.2017].
URL https://source.android.com/

[36] *Apple documentation*. [Online; navštíveno 8.3.2017].
URL https://developer.apple.com/develop/

[37] *Apple Pay*. [Online; navštíveno 9.3.2017].
URL http://www.apple.com/apple-pay/

[38] *Cochlear Implants*. [Online; navštíveno 13.2.2017].
URL http://www.asha.org/public/hearing/Cochlear-Implant/

[39] *Google Cardboard*. [Online; navštíveno 29.2.2017].
URL https://vr.google.com/cardboard/

[40] *Google Developer*. [Online; navštíveno 9.2.2017].
URL https://developer.android.com/

[41] *GSMArena*. [Online; navštíveno 11.03.2017].
URL http://www.gsmarena.com/

[42] *Material Design*. [Online; navštíveno 17.2.2017].
URL https://material.io/guidelines/

[43] *Microsoft HoloLens*. [Online; navštíveno 29.2.2017].
URL https://www.microsoft.com/microsoft-hololens/en-us

[44] *Mono*. [Online; navštíveno 6.3.2017].
URL http://www.mono-project.com/

[45] *Moto 360*. [Online; navštíveno 21.2.2017].
URL https://www.motorola.com/us/products/moto-360

[46] *Samsung Gear S3 Experience*. [Online; navštíveno 14.3.2017].
URL http://www.samsung.com/global/galaxy/gear-s3/experience/

[47] *Swift*. [Online; navštíveno 14.3.2017].
URL https://swift.org/

[48] *Tizen Developers*. [Online; navštíveno 14.3.2017].
URL https://developer.tizen.org/development

[49] *Wikipedia, the free encyclopedia*. [Online; navštíveno 9.3.2017].
URL https://en.wikipedia.org/

[50] *Xamarin Developer guidelines*. [Online; navštíveno 1.3.2017].
URL https://developer.xamarin.com/

Přílohy

Příloha A

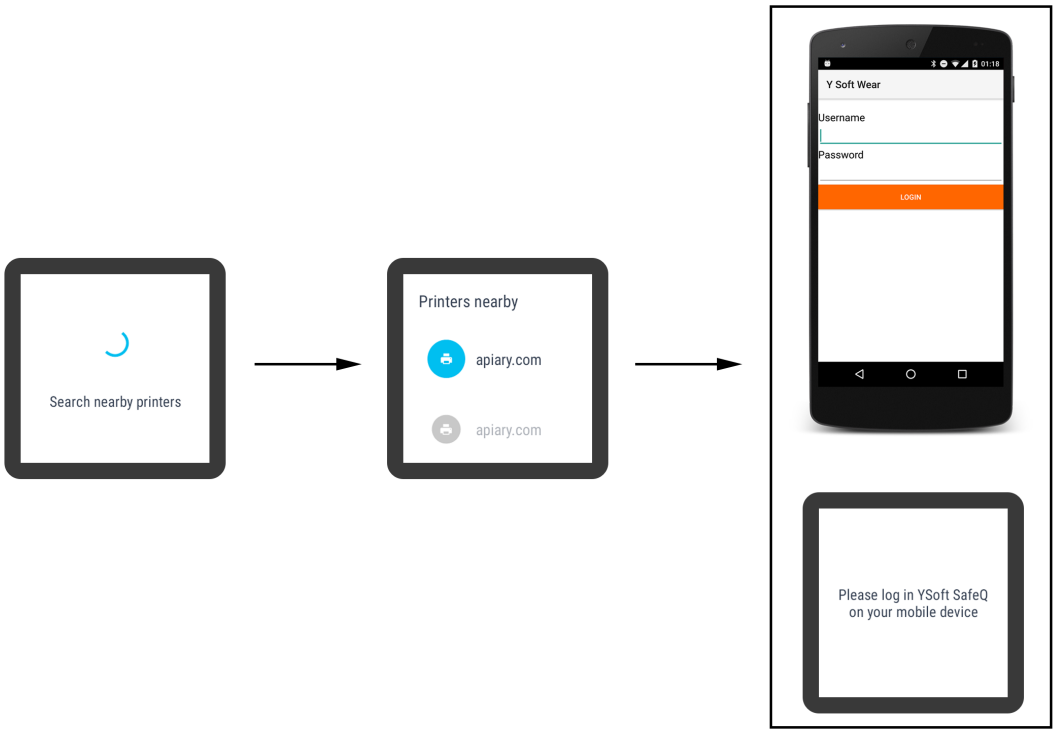
Tabulky

Zařízení	Systém	BlueTooth	WiFi	NFC	Chipset	RAM	Displej	Cena
Moto 360 1gen	Android Wear	4.0 LE	802.11 b/g	Ne	TI OMAP 3	512 MB	IPS 205ppi (320x290)	~5500 Kč
Moto 360 2gen	Android Wear	4.0 LE	802.11 b/g	Ne	Snapdragon 400	512 MB	IPS 263ppi (360x325)	~7000 Kč
Asus ZenWatch 2	Android Wear	4.1	802.11 b/g	Ne	Snapdragon 400	512 MB	AMOLED 278ppi (320x320)	~4000 Kč
Huawei Watch W1	Android Wear	4.1	802.11 b/g	Ne	Snapdragon 400	512 MB	AMOLED 286ppi (400x400)	~7500 Kč
Sony SmartWatch 3	Android Wear	4.0 LE	802.11 b/g	Ano	ARM A7	512 MB	AMOLED 283ppi (320x320)	~4000 Kč
Apple Watch 42mm	watchOS	4.0 LE	802.11 b/g/n	Ano	Apple S1	512 MB	AMOLED 302ppi (390x312)	~11000 Kč
Pebble Time Round	Pebble OS	2.1 / 4.0	Ne	Ne	Cortex M4	128 KB	ePaper 254ppi (180x180)	~6500 Kč
Samsung Gear Fit 2	Tizen	4.2	Ne	Ne	Exynos 3250	512 MB	AMOLED 1,5"(216x432)	~5000 Kč
Samsung Gear S2	Tizen	4.1	802.11 b/g/n	Ano	Exynos 3250	512 MB	AMOLED 302ppi (360x360)	~6500 Kč

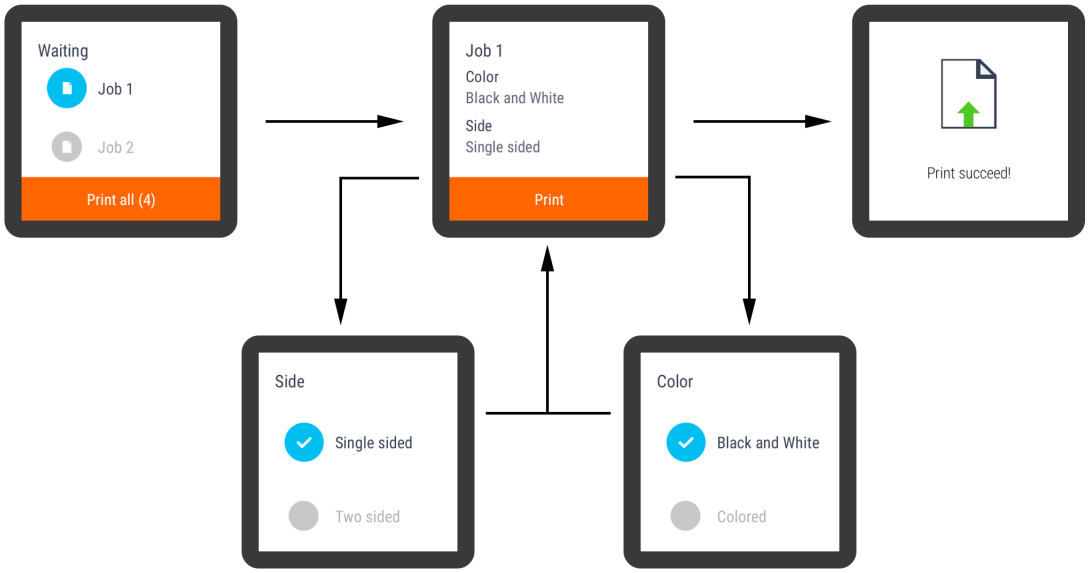
Tabulka A.1: Aktuální nabídka chytrých hodinek (specifikace čerpány z portálu gsmarena.com [41])

Příloha B

Obrázky



Obrázek B.1: Android Wear aplikace – 1. část



Obrázek B.2: Android Wear aplikace – 2. část

Příloha C

Manuál k překladu aplikace

C.1 Potřebné prostředí

Pro přeložení zdrojových souborů je zapotřebí mít k dispozici operační systém Windows a v něm nainstalované prostředí Visual Studio společně s balíčkem Xamarin. Visual Studio i s balíčkem Xamarin je možné stáhnout ze stránek projektu Xamarin[1]. Pokud již aplikaci Visual Studio máte, můžete po spuštění jejího instalátoru balíček Xamarin pouze doplnit do stávající instalace aplikace Visual Studio. Zda vše proběhlo správně můžete ověřit při vytváření nového projektu v aplikaci Visual Studio, kde by měla být k dispozici šablona aplikací pro systémy Android a Android Wear.

V rámci instalace balíčku Xamarin se automaticky stáhnou aktuální vývojové prostředky pro systém Android (Android SDK), které jsou zapotřebí pro překlad přiložené aplikace. Pokud by jejich instalace selhala, je možné je stáhnout ze stránek pro vývojáře aplikací systému Android[2]. V poslední řadě je zapotřebí nainstalovat Java Development Kit (JDK). Instalační balíček JDK je možné stáhnout ze stránek Oracle[3].

C.2 Stažení NuGet balíčků a překlad zdrojových souborů

Po instalaci potřebného prostředí je možné otevřít soubor `YSoftTerminal.sln` v prostředí Visual Studio. Po načtení projektu je zapotřebí stáhnout NuGet balíčky. To učiníme volbou `Restore NuGet Packages` v průzkumníku souborů prostředí Visual Studio. Poté již můžeme projekt přeložit volbou `Build Solution`, která přeloží zdrojové soubory a vygeneruje výsledný balíček s aplikací pro mobilní telefon a chytré hodinky.

[1] https://www.xamarin.com/download
[2] https://developer.android.com/studio/index.html
[3] http://www.oracle.com/technetwork/java/javase/downloads/

Příloha D

Manuál k Android a Android Wear aplikaci

D.1 Potřebná zařízení

Pro plně funkční chod aplikace jsou zapotřebí zařízení se systémem Android a Android Wear. V případě zařízení se systémem Android se může jednat o jakýkoliv mobilní telefon či tablet, který obsahuje systém verze 4.3 nebo vyšší. V případě zařízení se systémem Android Wear se může jednat o libovolné hodinky obsahující tento systém s verzí Android Wear 1 či Android Wear 2. Tato dvě zařízení musí být navzájem spárována pomocí aplikace `Android Wear`[1] v mobilním telefonu či tabletu.

D.2 Instalace vygenerovaného balíčku

Výsledný aplikační balíček s názvem `ysoft.apk` je možné nalézt v adresáři `app` na přiloženém nosiči. Tento soubor je zapotřebí zkopírovat do mobilního telefonu či tabletu například pomocí USB kabelu. Následně je zapotřebí jej v mobilním telefonu či tabletu nalézt pomocí jakéhokoliv průzkumníka souborů. Při pokusu o otevření tohoto souboru bude spuštěn instalátor aplikací systému Android a v něm bude možné odsouhlasit dále již bezobslužnou instalaci. Po instalaci aplikace do mobilního telefonu se automaticky započne instalace aplikace i do chytrých hodinek.

D.3 Hledání tiskáren

Aplikace po startu automaticky začne vyhledávat okolní tiskárny, ke které by se mohl uživatel připojit. Pro demonstrační účely je v seznamu okolních tiskáren pevně definovaná položka reprezentující neexistující tiskárnu pomocí výše zmíněné služby Apiary[2]. Tato tiskárna je sice pouze virtuální, ale díky aplikačnímu rozhraní služby Apiary jsou zasílány obdobné odpovědi, které by přišli z reálného serveru spravující konkrétní tiskárnu. Díky tomu je možné projít aplikaci i bez tiskárny s nakonfigurovaným Bluetooth majákem v blízkém okolí.

[1] https://play.google.com/store/apps/details?id=com.google.android.wearable.app
[2] https://apiary.io/

Příloha E

Obsah nosiče

- app/ – adresář s vygenerovaným instalačním balíčkem
- src/ – adresář obsahující zdrojové kódy aplikace
- tex/ – adresář obsahující zdrojové soubory k písemné zprávě
- projekt.pdf – text bakalářské práce
- video.flv – prezentační video výsledné terminálové aplikace