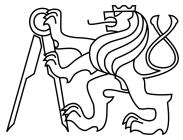
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Využití chytrých hodinek pro ovládání vzdálených zařízení
Student:	Jan Holub
Vedoucí:	Ing. Petra Pavlíčková, Ph.D.
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

Cílem práce je návrh a prototypová implementace technologie, která umožní pomocí chytrých hodinek ovládat vzdálená zařízení pomocí několika jednoduchých rozhraní: - true/false (např. otevři/zavři, ano/ne), vložení čísla (např. nastavení teploty na termostatu), poslání příkazu ke spuštění procesu (např. robot na uklízení začne uklízet) apod.

Pokyny pro vypracování:
1.Proveďte analýzu současných způsobů řešení.
2. Navrhněte vlastní řešení daného problému.
3. Navržené řešení implementujte.
4. Aplikaci otestujte a vytvořte dokumentaci.
5. Zhodnoťte výhody a nevýhody vašeho řešení.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 22. ledna 2017

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

Bakalářská práce

Využití chytrých hodinek pro ovládání vzdálených zařízení

Jan Holub

Vedoucí práce: Ing. Petra Pavlíčková, Ph.D.

16. května 2017

Poděkování

Chtěl bych poděkovat Ing. Petře Pavlíčkové, Ph.D. za vedení mé bakalářské práce a cenné rady. Děkuji taky své matce, která mi pomohla s finálními textovými úpravavami.

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 16. května 2017 .....

Odkaz na tuto práci

Holub, Jan. *Využití chytrých hodinek pro ovládání vzdálených zařízení*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017. Dostupný také z WWW: ⟨`https://aqueous-fortress-63326.herokuapp.com/`⟩.

Abstrakt

Tato bakalářská práce se zabývá vytvořením technologie, která umožní pomocí chytrých hodinek ovládat ostatní zařízení, která k tomu budou přizpůsobena. Práce obsahuje analýzu, návrh a samotnou implementaci. Výsledkem této práce je prototypová aplikace pro chytré hodinky, serverová aplikace a několik zkušebních virtuálních zařízení, která lze ovládat.

Klíčová slova chytré hodinky, ovládání vzdálených zařízení, phoenix framework, android

Abstract

This bachelor thesis fucuses on process of creation technology which allows to smart watches to control other devices which are adapted for it. The thesis contains analysis, design and implementation. Result of this thesis is prototype aplication for smart watches, server aplication and some trial virtual devices which can be controlled.

Keywords smart watches, remote device control, phoenix framework, android

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V dnešní době je stále více a více zařízení možno ovládat elektronicky. V mnoha situacích je nutné, aby uživatel ovládající tato zařízení byl v jejich těsné blízkosti. Existuje ovšem i velké množství případů, kdy přítomnost uživatele není potřeba, ba naopak je zbytečná. Přesně na taková zařízení bude tato práce zaměřena. K ovládání těchto zařízení budou využity chytré hodinky. Kvůli malým rozměrům hodinek [1] bude při návrhu aplikace kladen důraz na jednoduchost ovládacích rozhraní.

Zvětšujícím se trendem v oblasti chytrých zařízení se stává chytrá domácnost [2], která uživateli poskytuje možnost pohodlně ovládat jednotlivá zařízení připojená k řídící jednotce. Chytrá domácnost je většinou ovládána pomocí chytrých mobilů a tabletů. Existují ovšem také řešení, která pracují na bázi rozpoznávání hlasových příkazů.

Toto téma bakalářské práce jsem si zvolil, protože mě zaujala myšlenka jednoduše ovládat vzdálená zařízení pomocí chytrých hodinek. Tato technologie by měla být přínosem spíše pro jednotlivce a malé skupiny osob.

Kapitola 1

Cíle práce

Prvním cílem práce je provést analýzu současných způsobů řešení tématu ovládání vzdálených zařízení.

Druhým cílem je navrhnout vlastní řešení daného problému, provést analýzu formou funkčních a nefunkčních požadavků a následné sestavení a popsání několika detailních scénářů případů užití.

Dalším cílem je implementovat jednotlivé části technologie. První část je serverová aplikace, která obsahuje logickou a datovou vrstvu. Následně vytvořit zkušební ovládaná zařízení ve formě několika programů komunikujících se serverovou částí. Poté bude vytvořena část pro chytré hodinky a telefon s důrazem na jednoduchost jednotlivých ovládacích rozhraní.

Dalším cílem je otestovat tuto technologii jako celek a vytvořit dokumentaci.

Posledním cílem je zhodnotit výhody a nevýhody tohoto řešení.

Kapitola 2

Rešeršní část

2.1 Chytré hodinky

Chytré hodinky jsou zařízení, které patří do rodiny chytrých zařízení. Z dnešního pohledu se jedná o analogii chytrého telefonu v podobě hodinek.

Jedná se o zařízení, jenž umožňuje jejich nositeli interagovat se svým chytrým telefonem, který dále zajistí další zdroje jako např. připojení k internetu nebo určení polohy na mapě. Ke komunikaci mezi hodinkami a telefonem se používá technologie Bluetooth.

Pomocí chytrých hodinek lze telefonovat, odesílat zprávy, řídit svůj kalendář a využívat další funkce, které poskytuje chytrý telefon.

Hlavní výhodou chytrých hodinek je jednoduchost ovládání a rychlost přístupu k informacím a prostředkům, jenž chce uživatel najít.

Nevýhodou hodinek v současné době je malá kapacita baterie [3].

Jedná se o relativně novou technologii, proto není zatím moc rozšířená mezi běžnými uživateli.

2.1.1 Apple Watch

Apple Watch jsou náramkové chytré hodinky společnosti Apple s operačním systémem watchOS. Sdružují v sobě monitor fyzické aktivity a snaží se o sledování zdraví nositele (tep a jeho porovnání s dalšími údaji [4]).

Apple Watch spolehají na stálé bezdrátové připojení s iPhonem 5 nebo novějším.

2.1.2 Android Wear

Chytré hodinky s operačním systémem Android nabízí dobrou integraci s aplikacemi společnosti Google a přináší tím svému uživateli značný komfort a stabilní prostředí s jistotou dalšího budoucího rozvoje [5].

Mezi výrobce chytrých hodinek patří:

- LG
- Motorola
- Asus
- Sony
- Huawei
- Samsung

2.2 Existující řešení

V současné době existují podobná řešení problematiky ovládání zařízení pomocí chytrých technologií. Většinou se jedná o ovládání pomocí chytrých telefonů a tabletů. Jedním z rozšířených řešení je chytrá domácnost [6]. Řešení tohoto typu umožňuje zařízení ovládat i z velkých vzdáleností.

2.2.1 Chytrá domácnost

V domě je umístěna řídící jednotka, jenž je propojena s ostatními přístroji, které lze ovládat. Při vzdáleném přístupu uživatel komunikuje s touto řídící jednotkou, která nadále ovládá daná zařízení. Tato zařízení musí být ovšem v blízkosti řídící jednotky. Pokud jsou daleko od této jednotky, tak lze použít opakovač, který zesílí dosah. Když je řeč o dosahu řídící jednotky, jedná se o maximálně o pár set metrů.

Takové řešení je vhodné v případě, kdy je více přístrojů umístěných blízko sebe. Pokud je více zařízení rozmístěných daleko od sebe, tak toto řešení není vhodné.

2.2.1.1 Google Home

Google Home [7] je bezdrátový reproduktor s virtuálním asistentem ovládaný hlasem. Kromě poslechu hudby umožňuje vyhledávání informací, práci s kalendářem, ovládání ostatních zařízení v rámci chytré domácnosti.

Podpora více účtů Nová verze softwaru pro Google Home řeší připojení více účtů [8] Google k jednomu zařízení Home, který se tak napojí na data jednotlivých uživatelů. Aby vše fungovalo tak, jak má, dokáže systém rozpoznat i jednotlivé uživatele dle jejich hlasu. Tato vlastnost je extrémně důležitá při využívání těchto chytrých stanic v domácnostech, protože každá osoba má jinou historii a zájmy. Google používá personalizaci napříč různými službami, včetně hledání, takže v tomto případě jde jen o správnou hlasovou detekci toho, kdo se ptá a následného přizpůsobení odpovědí a reakce, stejně jako by to dělal

jednotlivý uživatel na jeho osobním mobilním telefonu. Rozpoznáváním uživatele podle hlasu je zabraňováno nežádoucímu zneužití této technologie, ke kterému již několikrát došlo [9].

Ovládání dalších přístrojů Společnost Google myslela také na komunikaci a ovládání dalších přístrojů v domácnosti. Skvělým příkladem je chytré osvětlení Philips Hue [10], které má přímou podporu Google Home. Spárování je nutné provést skrze oficiální aplikaci.

2.2.1.2 Amazon Echo

Amazon Echo [11] je bezdrátový reproduktor s hlasovým ovládáním a virtuální asistentkou jménem Alexa, který byl vytvořen společností Amazon. Přístroj dokáže například odpovídat na hlasové příkazy, přehrávat hudbu a ovládat chytrou domácnost.

Ve článku [12] má při porovnání ovládání chytrých domácností Amazon Echo stále výhodu před Google Home.

2.3 Internet věcí

Internet věcí (IoT) je termínem pro moderní přístroje ovladatelné i na dálku pomocí internetu. Jdou snadno obsloužit pomocí chytrého telefonu nebo je jednoduše připojít k centrální jednotce.

Propojení zařízení by mělo být zejména bezdrátové a mělo by přinést nové možnosti vzájemné interakce nejen mezi jednotlivými systémy a též přinést nové možnosti jejich ovládání, sledování a zajištění pokročilých služeb. Problémem jsou různé existující standardy pro komunikaci různých skupin výrobců.

2.3.1 Využití IoT

Patří sem nejen lednička, která vás upozorní na docházející jídlo a případně jej i objedná, ale i osobní lékařské přístroje (třeba skryté v hodinkách), které vám například průběžně měří puls apod.

Největšího nasazení se však IoT dočká v profesionální sféře jako třeba ve zdravotnictví, průmyslu včetně automobilového, energetice, obchodu [13].

2.3.2 Budoucnost IoT

Společnost BI Intelligence provedla průzkum, který říká, že do roku 2020 bude počet IoT zařízení okolo 34 miliard [14]. To jsou přibližně čtyři IoT zařízení pro každého člověka.

Kapitola 3

Analýza

3.1 Zvolené řešení

Rozhodl jsem se použít řešení, kdy nebudou chytré hodinky a ovládaná zařízení mezi sebou komunikovat přímo. Veškerá zařízení budou posílat dotazy na server (viz obrázek 3.1), který bude obsahovat logickou a datovou vrstvu této aplikace. Základním předpokladem tedy je, že všechna zařízení budou muset mít přístup k tomuto serveru.

3.1.1 Výhody a nevýhody zvoleného řešení

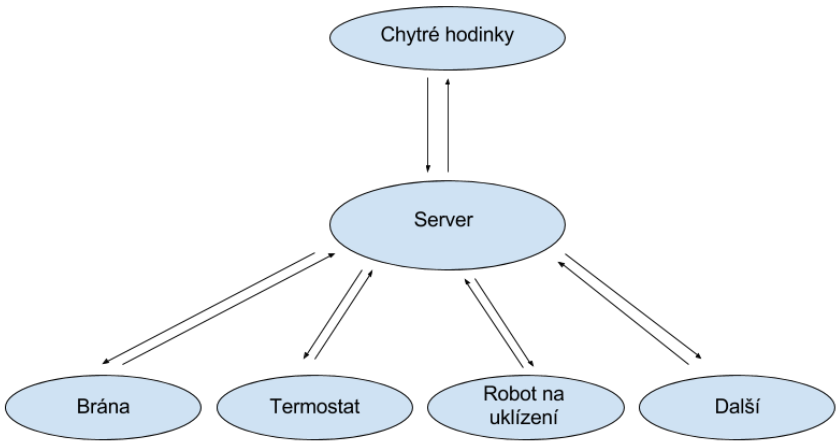
Mezi výhody zvoleného řešení patří, že zařízení mohou být umístěna kdekoliv. Nemusí být tedy umístěny v blízkosti řídící jednotky, jak je tomu u chytrých domácností.

Další výhodou je, že pro ovládání zařízení není nutné použít chytré hodinky. Komunikovat se serverem mohou jakákoliv zařízení, která umí používat API serveru. Samotné ovládání zařízení je možné přímo z webového rozhraní serveru, pokud bude tato funkčnost v budoucnu doplněna.

Hlavní nevýhodou je, že každé zařízení musí být samostatně schopno komunikovat se serverem. Zařízení nelze ovládat například pomocí technologie Bluetooth.

3.2 Typy rozhraní

Tato kapitola pojednává o typech ovládacích rozhraní, které budou implementovány. Kvůli velikosti a praktičnosti chytrých hodinek bude možné zařízení ovládat pomocí několika jednoduchých rozhraní.



Obrázek 3.1: Architektura zvoleného řešení.

3.2.1 Vložení celočíselné hodnoty

Toto rozhraní mají zařízení, kterým uživatel nastavuje stav pomocí celočíselných hodnot. Může se jednat např. o termostat umístěný v rodinném domě.

3.2.2 Vložení logické hodnoty

Tato zařízení lze ovládat pomocí dvou logických hodnot: true a false. Toto rozhraní je tedy vhodné pro zařízení, která nabývají dvou stavů. Typickým představitelem této skupiny mohou být vstupní dveře.

3.2.3 Spuštění procesu

Toto rozhraní je vhodné pro zařízení, která jsou schopna samostatně vykonávat nějaký proces. Díky tomuto rozhraní zařízení dostanou pouze impulz k zahájení činnosti. Vhodným příkladem by mohl být automatický robot na uklízení.

3.3 Požadavky

3.3.1 Funkční požadavky pro server

- F1: Registrace a přihlášení uživatele
- F2: Zobrazení přehledu zařízení
- F3: Přidávání zařízení

- F4: Spárování chytrého zařízení
- F5: Aktivace a deaktivace zařízení
- F6: Editace informací o zařízení
- F7: Zobrazení detailu jednotlivých zařízení
- F8: Nastavení nového stavu u zařízení

3.3.2 Funkční požadavky aplikace pro chytré hodinky

- F9: Zobrazení seznamu ovládaných zařízení
- F10: Zobrazení detailu zařízení
- F11: Odeslání dotazu na server pro změnu stavu zařízení

3.3.3 Nefunkční požadavky

- Připojeni k internetu
- Webový prohlížeč
- chytré hodinky s Android OS
- Jednoduchost ovládání

3.4 Případy užití

Pro jednodušší pochopení procesů je vhodné je popsat pomocí případů užití. Kromě aktérů konkrétního procesu a jejich zájmů jsou zde detailně popsány scénáře a případné alternativy nebo výjimky. Následující případy užití jsou psány podle doporučení uvedených v knize „Writing effective use cases“ [15].

3.4.1 U1: Registrace uživatele

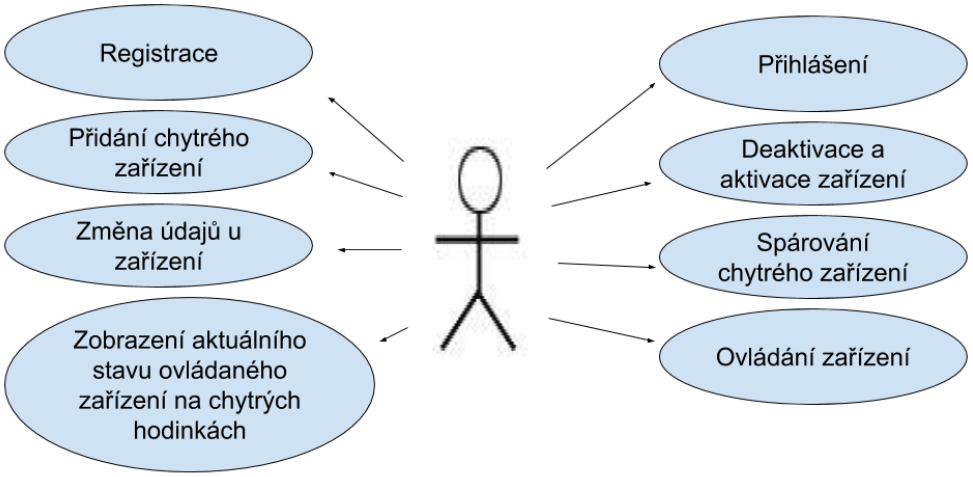
Hlavní účastník: Uživatel

Zúčastněné strany a zájmy:

- Uživatel - registrovat se.
- Server - vložit uživatele do databáze.

Úspěšné plnění: Uživatel se úspěšně zaregistruje.

Spouštěč: Požadavek uživatele



Obrázek 3.2: Případy užití.

Scénář úspěšného plnění:

1. Uživatel ve webovém rozhraní otevře položku Register.
2. Uživatel vloží do formuláře email, heslo, jméno, příjmení a odešle formulář.
3. Server uloží nového uživatele do databáze.

Vyjímky:

- **Zadaný email již existuje**
  Uživatel vloží jiný email a odešle formulář znovu.

3.4.2 U2: Přihlášení uživatele

Hlavní účastník: Uživatel

Zúčastněné strany a zájmy:

- Uživatel - přihlásit se.
- Server - kontrola přihlašovacích údajů a přihlášení uživatele.

Předpoklady:

- Uživatel je již registrován a není přihlášen.

Úspěšné plnění: Uživatel je přihlášen.

Spouštěč: Požadavek uživatele

Scénář úspěšného plnění:

1. Uživatel otevře webové rozhraní, kde uvidí přihlašovací formulář.
2. Uživatel vloží do formuláře přihlašovací údaje a odešle formulář.
3. Server zkontroluje správnost údajů a přihlásí uživatele.

Vyjímky:

- **Špatné přihlašovací údaje**
  Uživateli se zobrazí hláška říkající, že vložil špatné přihlašovací údaje.

3.4.3 U3: Přidání chytrého zařízení

Hlavní účastník: Uživatel

Zúčastněné strany a zájmy:

- Uživatel - přidání nového chytrého zařízení.
- Server - vložit zařízení do databáze.

Předpoklady:

- Uživatel je již registrován a přihlášen.

Úspěšné plnění: Uživatel úspěšně vytvoří chytré zařízení.

Spouštěč: Požadavek uživatele

Scénář úspěšného plnění:

1. Uživatel se přihlásí ve webovém rozhraní.
2. Uživatel otevře formulář pro přidání chytrého zařízení.
3. Uživatel vyplní ve formuláři název, popis a odešle formulář.
4. Server uloží nové zařízení do databáze.

Vyjímky:

- **Uživatel nevloží název do formuláře**
  Uživateli je zobrazena hláška říkající, že má zadat název zařízení.

3.4.4 U4: Spárování chytrého zařízení

Hlavní účastník: Uživatel

Zúčastněné strany a zájmy:

- Uživatel - spárovat chytré zařízení se serverem.
- Server - zpracovat požadavek na spárování od uživatele a telefonu.
- Telefon - odeslat serveru požadavek na spárování.

Předpoklady:

- Chytré zařízení (telefon a hodinky) je vloženo v systému.
- Chytré zařízení není spárované.

Úspěšné plnění: Uživatel úspěšně spáruje zařízení se serverem.

Spouštěč: Požadavek uživatele

Scénář úspěšného plnění:

1. Uživatel zobrazí detail zařízení ve webovém rozhraní na serveru.
2. Uživatel vloží požadavek na spárování zařízení.
3. Server uloží požadavek a poskytne uživateli autorizační klíč, který je platný po dobu pěti minut.
4. Uživatel klíč zadá do telefonu a odešle formulář.
5. Telefon odešle požadavek na spárování serveru.
6. Server zpracuje požadavek a vrátí telefonu token.

Alternativní scénář:

- **Klíč odeslaný z telefonu je neplatný**

  Uživatel je informován, že zadal špatný klíč.

- **Vyprší lhůta pro vložení klíče**

  Uživatel je informován o neplatnosti klíče.

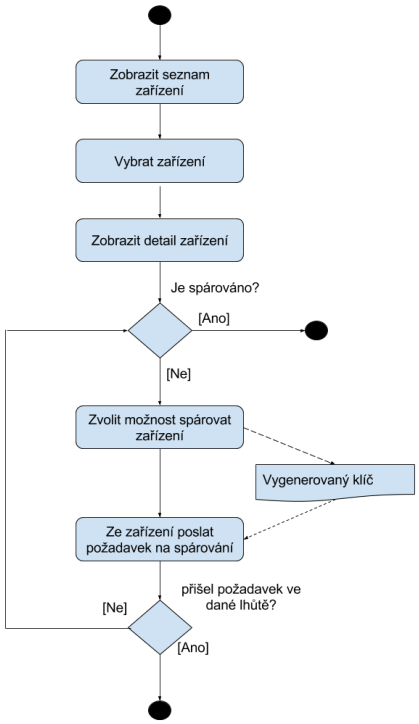
Vyjímky:

- **Chytré zařízení není vloženo v systému**

  Uživatel vloží chytré zařízení ve webovém rozhraní serveru a opakuje proces.

- **Chytré zařízení je již spárované**

  Uživatel nebude schopen vložit požadavek o spárování ve webovém rozhraní.



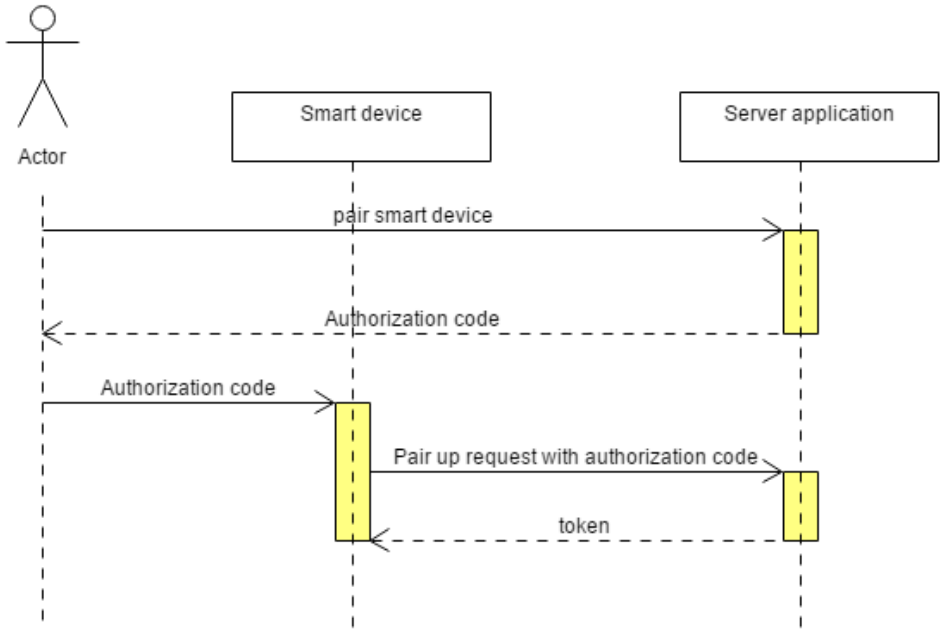
Obrázek 3.3: Spárování chytrého zařízení.

3.4.5 U5: Deaktivace a aktivace zařízení

Hlavní účastník: Uživatel

Zúčastněné strany a zájmy:

- Uživatel - deaktivace nebo aktivace chytrého zařízení.
- Server - povolit uživateli deaktivovat nebo aktivovat zařízení.



Obrázek 3.4: Sekvenční diagram spárování chytrého zařízení.

Předpoklady:

- Uživatel je již registrován a přihlášen.
- Editované zařízení je vloženo v systému.

Úspěšné plnění: Uživatel úspěšně deaktivuje nebo aktivuje zařízení.

Spouštěč: Požadavek uživatele

Scénář úspěšného plnění:

1. Uživatel otevře detail zařízení.
2. Uživatel klikne na tlačítko Deactivate device nebo Activate device.
3. Server deaktivuje nebo aktivuje zařízení.

3.4.6 U6: Změna údajů u zařízení

Hlavní účastník: Uživatel

Zúčastněné strany a zájmy:

- Uživatel - editovat údaje u zařízení.
- Server - zpracovat požadavek na změnu údajů.

Předpoklady:

- Uživatel je již registrován a přihlášen.
- Editované zařízení je vloženo v systému.

Úspěšné plnění: Uživatel úspěšně edituje zařízení.

Spouštěč: Požadavek uživatele

Scénář úspěšného plnění:

1. Uživatel najde v seznamu zařízení, které chce editovat, a zvolí možnost Edit.
2. Uživatel vloží do editačního formuláře nové hodnoty a odešle požadavek na změnu.
3. Server uloží nové hodnoty do databáze.

Vyjímky:

- **Uživatel ve formuláři nevyplní název nebo popis**

  Uživateli je zobrazena hláška, že zadané hodnoty nesmí být prázdné.

3.4.7 U7: Zobrazení aktuálního stavu ovládaného zařízení na chytrých hodinkách

Hlavní účastník: Uživatel

Zúčastněné strany a zájmy:

- Uživatel - zobrazit stav zařízení.
- Chytré hodinky - zobrazit stav zařízení.

Předpoklady:

- Chytré hodinky jsou vloženy v systému.
- Chytré hodinky jsou spárovány a aktivovány.
- Zařízení je vloženo v systému.
- Zařízení je spárováno a aktivováno.

Úspěšné plnění: Uživatel zobrazí stav zařízení.

Spouštěč: Požadavek uživatele

Scénář úspěšného plnění:

1. Uživatel otevře aplikaci v chytrých hodinkách.
2. Uživatel ze seznamu zařízení vybere zařízení.
3. Chytré hodinky zobrazí uživateli aktuální stav zařízení.

3.4.8 U8: Ovládání zařízení

Hlavní účastník: Uživatel

Zúčastněné strany a zájmy:

- Uživatel - změnit nastavenou hodnotu u zařízení.
- Chytré hodinky - poskytnout uživateli žádané informace a odeslat serveru požadavek na změnu hodnoty.
- Server - zpracovat požadavek na změnu hodnoty od uživatele a telefonu.

Předpoklady:

- Chytré zařízení (telefon a hodinky) jsou spárovány se serverem.
- Ovládané zařízení je v sytému.

Úspěšné plnění: Uživatel úspěšně změní nastavenou hodnotu zařízení.

Spouštěč: Požadavek uživatele

Tabulka 3.1: Mapování případů užití na požadavky

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
U1	X										
U2	X										
U3		X	X								
U4		X		X			X				
U5		X			X		X				
U6		X				X					
U7									X	X	
U8								X	X	X	X

Scénář úspěšného plnění:

1. Uživatel v chytrých hodinkách otevře seznam zařízení.
2. Uživatel rozklikne zařízení, kterému chce nastavit hodnotu.
3. Uživatel zadá novou hodnotu a odešle ji.
4. Hodinky odešlou požadavek na serveru.
5. Server zpracuje požadavek.

Alternativní scénář:

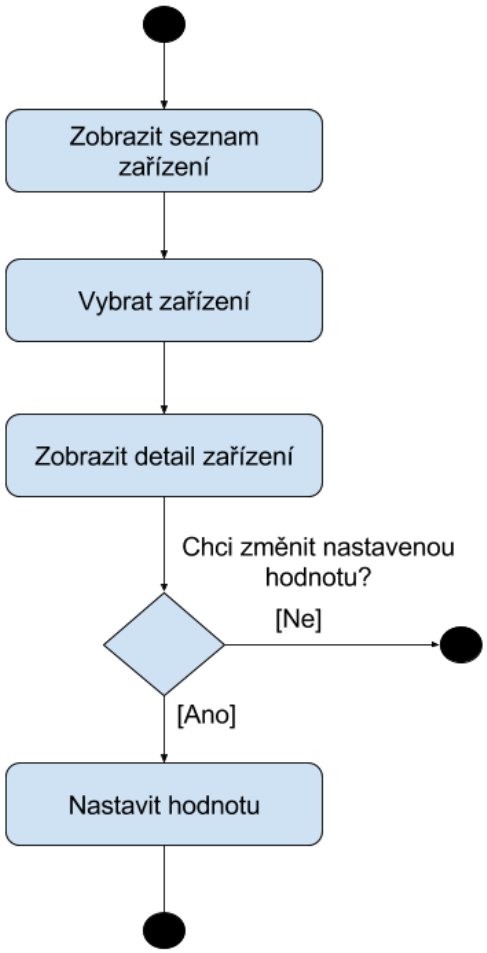
- **Zařízení již má nastavenou požadovanou hodnotu**
  Požadavek na změnu hodnoty se neodešle, protože není zapotřebí.
- **Zařízení není vloženo v systému**
  Uživatel vloží zařízení ve webovém rozhraní serveru a opakuje proces.

Vyjímky:

- **Chytré zařízení nemá přístup k serveru**
  Uživateli je zobrazena hláška, že nemá přístup k serveru.

3.5 Mapování případů užití na požadavky

V této podkapitole je zobrazeno mapování případů užití U na funkční požadavky F. Funkční požadavky F1 až F8 se vztahují k serverové aplikaci. Požadavky F9 až F11 souvisí s aplikací pro chytré hodinky.



Obrázek 3.5: Ovládání jednotlivých zařízení.

Kapitola 4

Realizace serverové aplikace

V prvním kroku jsem se rozhodl vytvořit serverovou část s webovým rozhraním, protože obsahuje veškerou logiku.

4.1 Použité technologie

4.1.1 Elixir

Elixir je dynamický funkcionální programovací jazyk navržený k budování škálovatelných a udržovatelných aplikací.

4.1.2 Phoenix

Phoenix je framework napsaný v elixiru pro tvorbu webových aplikací založený na MVC architektuře. Aplikace napsaná v phoenixu se skládá z několika vrstev.

Tyto vrstvy jsou:

- The Endpoint
- The Router
- Controllers
- Views
- Templates
- Channels
- PubSub

4.1.2.1 Live reload

Jednou z vlastností phoenix frameworku, která urychluje práci je live reload. Live reload znamená, že při upravení části zdrojového kódu a uložení tohoto souboru, se na pozadí kód zkompiluje a změny se projeví v prohlížeči bez nutnosti obnovení stránky. Tento proces je velmi rychlý a výsledek lze vidět téměř ihned.

4.1.2.2 Generators

Phoenix framework umožňuje pomocí příkazu mix generovat části svého kódu. Existuje několik typů generátorů, které generují různé části kódu.

4.1.2.3 Mix tasks

Při programování ve phoenix frameworku je nutné umět pracovat s příkazovou řádkou, kde využijeme příkaz mix. Tento příkaz má široké spektrum využití. Lze ho použít jak na vytvoření nového projektu, tak na generování částí kódu, migrací databáze, spuštění aplikace a dalším.

```
$ mix help | grep -i phoenix
mix local.phoenix        # Updates Phoenix locally
mix phoenix.digest       # Digests and compress static files
mix phoenix.gen.channel  # Generates a Phoenix channel
mix phoenix.gen.html     # Generates controller, model and views for an HTML based resource
mix phoenix.gen.json     # Generates a controller and model for a JSON based resource
mix phoenix.gen.model    # Generates an Ecto model
mix phoenix.gen.presence # Generates a Presence tracker
mix phoenix.gen.secret   # Generates a secret
mix phoenix.new          # Creates a new Phoenix v1.2.1 application
mix phoenix.routes       # Prints all routes
mix phoenix.server       # Starts applications and their servers
```

Obrázek 4.1: Příklady používaných mix příkazů.

4.1.2.4 Ecto

Ecto je technologie, kterou využívá phoenix ke komunikací s databází. Zajišťuje nám bezpečný přistup k databázi, kdy zabraňuje útokům jako je SQL injection a dalším.

Ecto je kompatibilní s následujícími databázemi:

1. PostgreSQL
2. MySQL
3. MSSQL
4. SQLite3

5. MongoDB

```
$ mix help | grep -i ecto
mix ecto.create          # Create the storage for the repo
mix ecto.drop            # Drop the storage for the repo
mix ecto.gen.migration   # Generate a new migration for the repo
mix ecto.gen.repo        # Generates a new repository
mix ecto.migrate         # Runs migrations up on a repo
mix ecto.rollback        # Reverts migrations down on a repo
```

Obrázek 4.2: Příklady používaných mix příkazů pro práci s ecto.

4.1.3 PostgreSQL

Phoenix s ecto modulem ve svém výchozím nastavení používá databázi PostgreSQL. PostgreSQL je plnohodnotným relačním databázovým systémem s otevřeným zdrojovým kódem. Má za sebou více než devatenáct let aktivního vývoje a také vynikající pověst pro svou spolehlivost a bezpečnost. Běží nativně na všech rozšířených operačních systémech. Plně podporuje cizí klíče, operace JOIN, pohledy, spouště a uložené procedury.

Obsahuje většinu SQL92 a SQL99 datových typů, např. INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL a TIMESTAMP. Nechybí ani podpora moderních datových typů jako je JSON nebo XML.

4.1.4 GrahpQL

GraphQL je dotazovací jazyk, ve kterém vymezíme, jaká data ze serveru chceme získat. V odpovědi jsou pouze tato data bez jakýchkoliv dalších informací.

Tento přístup má výhodu v tom, že data, u kterých potřebujeme k výpočtu více výpočetních zdrojů a nejsou zapotřebí, nemusíme vypočítávat. Server poskytne odpověď ve formátu JSON.

```
{
  human(id: "1000") {
    name
    height
  }
}
```

```
{
  "data": {
    "human": {
      "name": "Luke Skywalker",
      "height": 1.72
    }
  }
}
```

Obrázek 4.3: Příklad graphQL dotazu.

4.1.5 Verzovací systém

Pro správu verzí jsem se rozhodl použít systém Git, jelikož s ním mám nejvíce zkušeností. Společně s Gitem použiji veřejným repozitář, který poskytuje GitLab.

4.2 GUI

Po otevření webového rozhraní v prohlížeči je vyžadován email a heslo k přihlášení k účtu. Pokud uživatel nemá vytvořený účet, tak se musí zaregistrovat a následně se může přihlásit. Po úspěšném přihlášení uvidí své údaje a dvě tabulky.

První tabulka obsahuje seznam všech chytrých zařízení a několik základních informací o nich. Zde je také možné rozkliknout detail nebo editovat zařízení. V detailu zařízení lze spárovat a odpárovat zařízení. Je zde možnost deaktivovat zařízení

Druhá tabulka obsahuje seznam všech ovládaných zařízení a jejich aktuální nastavenou hodnotu. Stejně jako u první tabulky je možné rozkliknout detail nebo editovat zařízení. Po rozkliknutí zařízení se zobrazí seznam všech nastavených hodnot a akcí pro zařízení. Na této stránce lze také spárovat a odpárovat zařízení a popřípadě jej deaktivovat.

4.3 Databázový model

Tato kapitola je zaměřena na návrh databázového modelu.

Každá tabulka bude obsahovat atributy o vytvoření záznamu a změně záznamu, přestože tyto atributy nejsou vyznačené v návrhu.

Pokud se uživatel rozhodne, že si nepřeje, aby bylo možno používat chytré zařízení nebo ovládané zařízení, může zařízení pouze deaktivovat. Až se rozhodne zařízení opět uvést do provozu, pouze ho aktivuje a bude jej moci opět využívat.

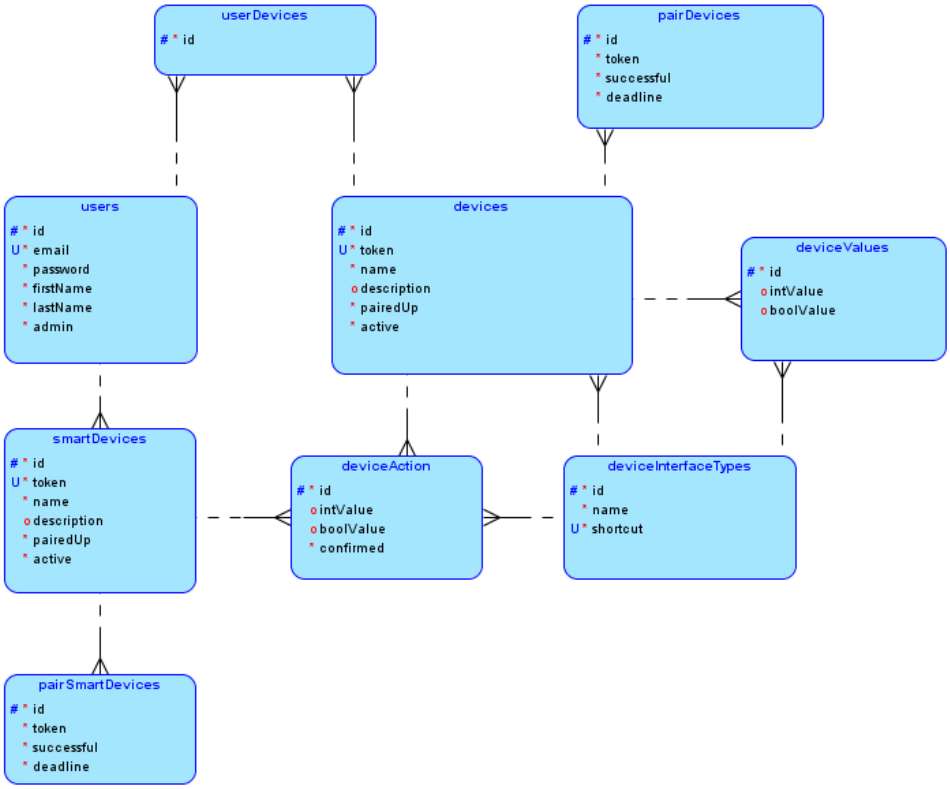
4.3.1 Users

Tabulka udržuje seznam registrovaných uživatelů. Každý uživatel musí být unikátní email, který slouží k přihlášení přes webové rozhraní.

Z logického návrhu databáze jde vidět, že každý uživatel může mít jak více smartDevices, tak více devices.

4.3.2 SmartDevice

Jedná se o zařízení, na kterém bude nainstalovaná aplikace pro ovládání ostatních zařízení. Nemusí to být pouze chytré hodinky. Může to být jakékoliv zařízení schopné komunikovat se serverem.



Obrázek 4.4: Návrh databáze.

Atribut token slouží pro identifikaci zařízení při odeslání dotazu na server. Pouze pokud je atribut active nastaven na true, tak server vyhovoví požadavkům na ovládání zařízení.

4.3.3 pairSmartDevice

Pomocí SmartDevice nebude možné ovládat žádná zařízení, dokud nebude správně spárované se serverem. Pro uložení informací o pokusech o spárování zařízení slouží tabulka pairSmartDevice.

Počet pokusů o spárování není omezen, avšak je možné mít pouze jeden pokus o spárování konkrétního zařízení v jednom okamžiku.

Atribut successful udává výsledek pokusu o spárování.

4.3.4 Device

Jedná se o zařízení, které je možno ovládat. Může se jednat o jakékoliv zařízení schopné komunikovat se serverem. Databázový návrh umožňuje, aby device bylo ovládáno více uživateli.

Atribut token slouží pro identifikaci zařízení při odeslání dotazu na server. Pouze pokud je atribut active nastaven na true, tak lze ovládat toto zařízení.

4.3.5 UserDevice

Mezi entitami users a devices je vztah M:N, který je realizován pomocnou tabulkou userDevices.

4.3.6 pairDevice

Aby bylo možné ovládat konkrétní device, je nutné ho nejprve spárovat se serverem. Pro uložení informací o pokusech o spárování zařízení slouží tabulka pairDevice. Pokud spárování proběhlo v pořádku, je atribut successful nastaven na true.

4.3.7 DeviceAction

Tato entita je určena k uchování všech požadavků od chytrých zařízení o změnu hodnoty jednotlivých ovládaných zařízení.

Jelikož mohou jednotlivá zařízení nabývat hodnot odlišných datových typů, tak je pro každý tento datový typ vytvořen atribut. Ostatní atributy mají hodnotu null. Pro uchování hodnot součastných rozhraní stačí atributy intValue (celočíselná hodnota) a boolValue (logická hodnota).

Entita obsahuje také cizí klíč, který se odkazuje do tabulky DeviceInterfaceType, abychom mohli zjistit, ke kterému rozhraní se hodnota vztahuje, pokud bude více rozhraní používat stejný datový typ.

Dále je zde přítomen atribut confirmed, který říká, zda byla nastavená hodnota detekována konkrétním ovládaným zařízením. Tento atribut není důležitý u nastavení např. číselné hodnoty zařízení, ale je důležitý při spouštění akcí, kde zajišťuje, aby nebyla nějaká akce opětovně spuštěna po jejím dokončení.

4.3.8 DeviceValue

Tato entita uchovává historii všech hodnot, které kdy jednotlivá zařízení nabyla.

Je možné zvolit řešení, kdy by v entitě Device byla uložená aktuální nastavená hodnota. Řešení s uchováním historie hodnot je zvoleno, protože průběžně uchovaná data mohou být následně použita k analýze historie jednotlivých zařízení.

4.3.9 DeviceInterfaceTypes

Jedná se o seznam typů rozhraní, která mohou mít zařízení. Entita má unikátní atribut shortcut, který jednoznačně určuje typ rozhraní.

4.4 Architektura

Jednotlivé moduly, ze kterých se skládá aplikace ve phoenix frameworku jsou rozděleny do několika sekcí.

4.4.1 Models

Pro každou entitu z databázového modelu je vytvořen model. Pro vytváření takových modelů má phoenix framework generátory, které usnadní a urychlí práci. O generátorech se zmiňuji v kapitole o phoenix frameworku.

4.4.2 Controllers

Veškeré kontrolery, které reprezentují logickou vrstvu aplikace jsou uloženy zde. Při dotazu na server router přesměruje dotaz na konkrétní kontroler, který má za úkol zpracovat dotaz a vrátit výsledek.

4.4.2.1 DeviceActionController

Tento kontroler přijímá a zpracovává dotazy, které pracují s požadavky na změnu stavu zařízení.

setAction Dotaz poslaný z chytrého zařízení, který uloží do databáze požadavek na změnu stavu

getAction Dotaz odeslaný z ovládaného zařízení, který vrátí poslední požadavek na změnu hodnoty

4.4.2.2 Další kontrolery

1. DeviceController
2. DeviceValueController
3. PageController
4. SessionController
5. SmartDeviceController
6. UserController

4.4.3 Moduls

Zde jsou umístěny pomocné moduly, které obstarávají další funkcionalu.

4.4.3.1 Generator

Tento modul obsahuje sadu funkcí pro generování různých výstupů. Je zde obsažena např. funkce generující náhodný řetězec, která je využita při spárování zařízení a generování tokenu pro zařízení.

4.4.3.2 Hash

Modul obsahující hashovací funkci, která je použita pro ukládání hesel do databáze.

4.4.3.3 Session

Modul má na starost práci se session.

4.4.4 Plugs

Plug je označení pro modul, jehož funkce call je volána dříve, než je dotaz předán kontroleru. Ve funkci call lze dotaz upravit, zrušit nebo zavolat jinou funkci. Plug se přiřazuje ke konkrétní množině dotazů v routeru.

4.4.4.1 Authenticate

Tento plug je přiřazen k dotazům, jež přistupují k datům, která jsou určena pouze pro přihlášené uživatele. Plug zavolá funkci modulu Session a zjistí, zda je uživatel přihlášen. Pokud uživatel není přihlášen, tak přesměruje dotaz na přihlašovací stránku. Pokud uživatel je přihlášen, tak dotaz neupravuje.

4.4.5 Templates

Zde jsou umístěny soubory obsahující html kód, který je následně vykreslen v prohlížeči.

Kapitola 5

Realizace ovládaných zařízení

Jako simulaci opravdových zařízení jsem vytvořil pár jednoduchých virtuálních zařízení. Tato zařízení jsem naprogramoval v javě.

Jejich funkčnost spočívá v tom, že odešlou na serverovou aplikaci dotaz, který jim vrátí informace o stavu, který má zařízení nabýt. Následně tento stav nasimulují a informaci o novém stavu odešlou zpět na server. Poté se vlákno tohoto procesu uspí na dobu určitou. Po probuzení vlákna se cyklus opakuje.

Jednotlivá zařízení lze spouštět dle potřeby z příkazové řádky. Je možné nastavit jako argument délku spaní vlákna v milisekundách. Pokud nebude hodnota nastavena samotným uživatelem, tak bude automaticky nastavena na jednu vteřinu. Délka uspání vlákna je omezena minimální a maximální časovou hodnotou, aby nedošlo k nastavení nežádoucích hodnot.

Pro testovací účely jsou vytvořený tři zařízení:

1. termostat (vkládání celočíselné hodnoty)
2. dveře (vkládání logické hodnoty)
3. robot na uklízení (odeslání příkazu ke startu procesu)

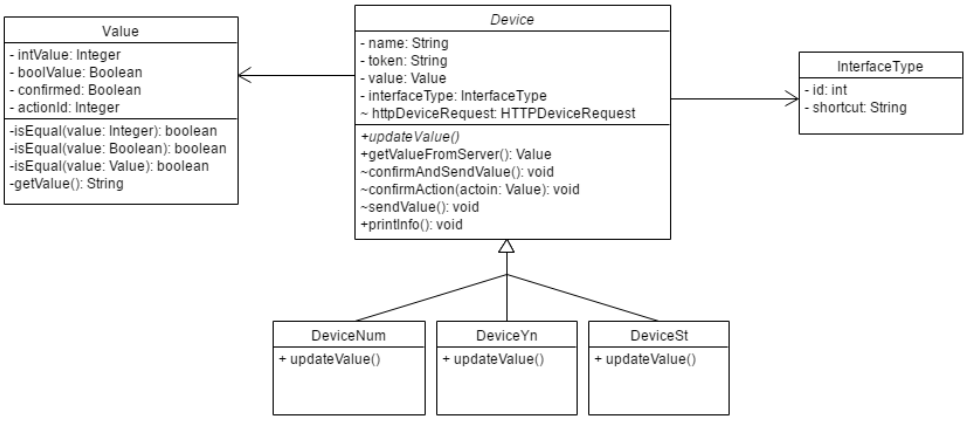
5.1 Návrhový model tříd

Tato kapitola se zabývá a popisuje návrhem tříd pro ovládaná zařízení.

5.1.1 HTTPDeviceRequest

Třída, která se stará o veškerou komunikaci se serverem. Poskytuje metody, které posílají tři typy dotazů:

1. Zjištění informace o hodnotě, jaké má zařízení nabýt
2. Odeslání aktuální hodnoty zařízení



Obrázek 5.1: Návrhový model tříd ovládaných zařízení.

3. Odeslání potvrzení o přijetí hodnoty, na kterou má být zařízení nastaveno

5.1.2 Value

Třída reprezentuje nastavenou hodnotu pro zařízení. Obsahuje mimo jiné také proměnnou, která udává, zda bylo na server odesláno potvrzení o přijetí této hodnoty. Proměnná actionId obsahuje číselný identifikátor záznamu v databázi, který nastavuje tuto hodnotu. Tento identifikátor je využit při odesílání dotazu o potvrzení přijetí hodnoty.

5.1.3 InterfaceType

Tato třída reprezentuje typ rozhraní u zařízení.

5.1.4 Device

Abstraktní třída reprezentující konkrétní zařízení. Potomci této třídy musí definovat metodu updateValue, která udává, jakým způsobem zařízení reaguje na novou hodnotu přijatou ze serveru.

5.1.5 DeviceNum

Tato třída je první z potomků třídy Device, která používá typ rozhraní, které pracuje s celými čísly. V metodě updateValue nejprve zkontroluje, zda je nová přijatá hodnota jiná než je aktuální nastavená hodnota. Pokud je podmínka splněna, tak se nastaví hodnota dle následujícího pseudokodu:

if(value < newValue) value++; else value–;

Pro zjednodušení v příkladu předpokládáme, že value a newValue jsou celá čísla, ikdyž představují objekty a musíme k jejich hodnotám přistupovat pomocí metod.

5.1.6 DeviceYn

Další potomek třídy Device používající rozhraní založené na vkládání logické hodnoty, řeší metodu updateValue jednoduše. Pokud se nastavená hodnota neshoduje s novou hodnotou, tak je hodnota pouze nahrazena hodnotou novou.

5.1.7 DeviceSt

Tento potomek třídy Device reprezentuje zařízení s typem rozhraní spuštění procesu. Zde je uvedená metoda updateValue pokrývající více případů:

1. Proces běží Pokud u nové příchozí hodnoty ze serveru není potvrzeno přijetí zařízením, tak se nejprve odešle potvrzení o přijetí. Když je proces ukončen, tak je informace o novém stavu odeslána na server. Rozhodnutí o ukončení daného procesu má na starost metoda isProcessFinished, která využívá náhodný generátor a ukončí proces s pravděpodobností 5% při každém zavolání.

2. Proces neběží Pokud je aktuální hodnota stejná jako nová přijatá hodnota, tak nedojde k žádné změně. Pokud je aktuální hodnota různá od přijaté hodnoty a přijatá hodnota není potvrzena, spustí se proces a je tato informace odeslána na server.

Kapitola 6

Realizace aplikace pro chytré hodinky

Při vývoji aplikace pro chytré hodinky je nutné také naprogramovat část pro mobilní telefon. Projekt je rozdělen do několika modulů. Zdrojové kódy aplikace pro mobilní telefon se ukládají do modulu mobile a zdrojové kódy pro chytré hodinky se ukládají do modulu wear. Jelikož oba moduly využívají některé společné funkce aplikace, tak je vytvořen další modul, do kterého je uložena společná logika. Tento modul se nazývá myCommonLibrary.

6.1 Použité technologie

6.1.1 Java

Java je všeobecně použitelný vysokoúrovňový, objektově orientovaný programovací jazyk. Zdrojový kód je překládán do tzv. bajtkódu, který je následně interpretován virtuálním strojem JVM (Java Virtual Machine). Java Virtual Machine označuje jak specifikaci popisující architekturu virtuálního počítače, tak i vlastní program provádějící implementaci bajtkódu.

6.1.2 JSON

JSON je způsob zápisu dat (datový formát) nezávislý na počítačové platformě, určený pro přenos dat, která mohou být organizována v polích nebo agregována v objektech.

Příklad:

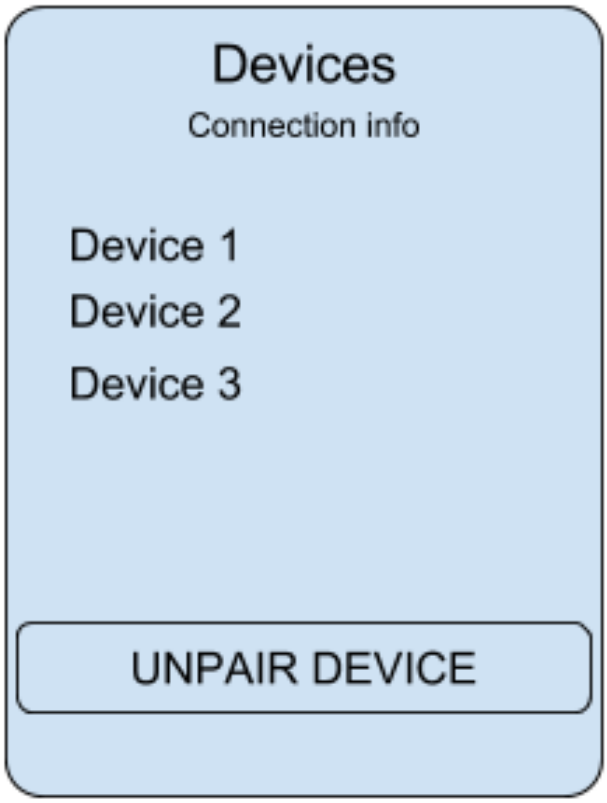
{ “name”: “Jan”, “age”: 22, “boolean”: true }

6.2 GUI

Tato kapitola se zaměřuje na návrh grafického uživatelského rozhraní pro mobil a chytré hodinky.

6.2.1 Návrh GUI pro mobil

Jelikož mobil slouží pouze jako prostředník mezi serverem a hodinkami, tak obsahuje pouze seznam ovládaných zařízení a možnost spárovat a odpárovat se od serveru.



Obrázek 6.1: Návrh GUI pro mobil.

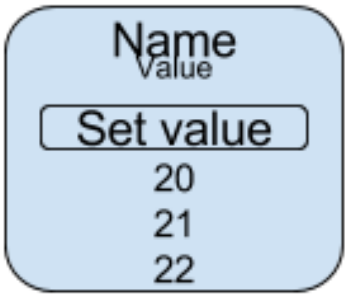
6.2.2 Návrh GUI pro hodinky

Kvůli malým rozměrům chytrých hodinek je při návrhu kladen důraz na jednoduchost jednotlivých obrazovek. Na úvodní obrazovce je zobrazen seznam zařízení, která lze ovládat. Při kliknutí na zařízení je otevřena obrazovka s detailem tohoto zařízení.

Obrázek 6.2: Návrh GUI pro hodinky.

6.2.2.1 Wireframe pro vložení celočíselné hodnoty

Tato obrazovka obsahuje název, nastavenou hodnotu a aktuální tohoto zařízení. Pro změnu nastavené hodnoty si lze vybrat novou hodnotu ze seznamu a slisknou tlačítko Set value.



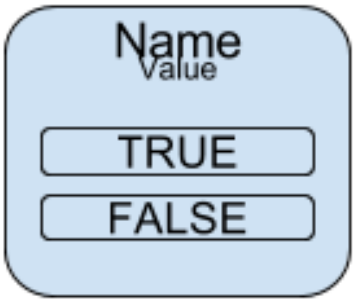
Obrázek 6.3: Wireframe pro vložení celočíselné hodnoty.

6.2.2.2 Wireframe pro vložení logické hodnoty

Na obrazovce se nachází název a aktuálně nastavená hodnota. Po zmáčknutí tlačítek TRUE nebo FALSE se odešle požadavek na změnu aktuální hodnoty tohoto zařízení.

6.2.2.3 Wireframe pro spuštění procesu

Tato obrazovka obsahuje název zařízení. Pokud zařízení vykonává proces, je zobrazena hláška, že proces běží. Pro spuštění procesu je zde umístěno tlačítko START.



Obrázek 6.4: Wireframe pro vložení logické hodnoty.

Obrázek 6.5: Wireframe pro spuštění procesu.

6.3 Modul myCommonLibrary

Tento modul obsahuje třídy, které jsou společné jak pro modul pro mobil, tak pro modul pro chytré hodinky.

6.3.1 ConnectionInfoShortcuts

Tato třída obsahuje společné konstanty mezi dalšími moduly. Aplikace běžící na mobilu a na chytrých hodinkách si mezi sebou posílají zprávy. Posílaná zpráva obsahuje identifikátor, aby bylo jasné, o jaký typ zprávy se jedná. V této třídě jsou konstanty, které se používají jako výše zmíněné identifikátory zpráv.

6.3.2 Device

Aplikace pro mobil i pro hodinky potřebuje pracovat s touto třídou, která reprezentuje ovládané zařízení.

6.3.3 InterfaceType

Tato třída reprezentuje typ rozhraní pro zařízení.

6.3.4 Value

Jedná se o třídu pro ukládání hodnot jednotlivých zařízení.

6.3.5 JSONParser

Tato třída obsahuje statické metody, které pracují s daty ve formátu JSON.

6.3.5.1 JSONToDevice

Metoda zpracuje vstupní argument, jenž je řetězec ve formátu JSON obsahující informace o jednom zařízení. Výstupem této metody je objekt Device.

6.3.5.2 JSONObjectToDevice

Metoda převádí vstupní argument typu JSONObject na objekt typu Device.

6.3.5.3 DeviceToJSON

Tato metoda převede objekt typu Device na řetězec ve formátu JSON.

6.3.6 Helper

Tato třída obsahuje pomocné metody určené převážně pro ukládání dat do SharedPrefereences. Jedná se o kolekce typu klíč-hodnota, která umožňuje ukládat data.

Třída také obsahuje několik metod pro zobrazení informativních hlášek na obrazovku.

6.3.6.1 Metody pracující s tokenem

Token je unikátní řetězec, kterým se zařízení identifikuje serveru. Pokud zařízení nemá token, tak se musí poslat požadavek na server pro spárování. V případě úspěšného spárování zařízení obdrží token.

Třída Helper implementuje následující metody pro práci s tokenem

1. saveToken
2. removeToken
3. getToken
4. isPairedUp

isPairedUp Metoda vrací logickou hodnotu určující, zda je zařízení spárováno se serverem.

6.3.6.2 Metody pro práci se třídou Device

Následující metody poskytují rozhraní základních operací pro práci s třídou Device:

1. saveDevice
2. getDevice
3. getAllDevices
4. updateDevice
5. removeDevice
6. removeAllDevices

6.4 Modul mobile

Tento modul obsahuje zdrojové kódy pro mobilní telefon.

Pokud je telefon spárovaný se serverem, tak úvodní stránka obsahuje seznam zařízení, které můžeme pomocí hodinek ovládat. Je zde také možnost odpárovat zařízení.

V případě, že telefon není spárovaný se serverem, je zobrazeno textové pole pro zadání spárovacího klíče a tlačítko pro odeslání dotazu o spárování.

Veškerá komunikace se serverem probíhá skrz tento modul a následně je přeposílána z mobilu do chytrých hodinek.

6.4.1 MainActivity

Tato třída se stará o inicializaci a vykreslení komponent na obrazovku. Spouští také vlákno, které volá metodu pro získání informací o ovládaných zařízeních ze serveru. Po obdržení informací o zařízeních dále posílá tyto informace hodinkám. Požadavek na server o aktuální informace o ovládaných zařízeních je posílán každé 2,5 sekundy.

Třída obsahuje také Listener, který přijímá zprávy od hodinek a reaguje na ně. Hodinky posílají mobilu dva typy zpráv:

UPDATE_NETWORK_INFO Po přijetí této zprávy je zpět hodinkám poslána zpráva, zda má mobil přístup k serveru.

DEVICE_ACTION_REQUEST Zpráva obsahuje požadavek pro server o změnu nastavené hodnoty konkrétního zařízení. Mobil tento požadavek odešle na server.

6.4.2 ConnectionHelper

Třída obsahuje metody pro komunikaci se serverem a hodinkami.

6.4.2.1 Komunikace se serverem

Pro komunikaci mezi mobilem a serverem jsou použity následující metody.

pairUpDevice Metoda pošle na server dotaz pro spárování mobilu. Pokud je dotaz úspěšný, tak server mobilu pošle token, který si mobil uloží pro další komunikaci se serverem.

getDevices Tato metoda pošle serveru dotaz, který vrátí seznam zařízení, které je možné ovládat pomocí chytrých hodinek. Seznam těchto zařízení je dále přeposlán hodinkám.

sendDeviceActionRequest Metoda slouží k odeslání dotazu o změnu hodnoty zařízení.

isConnected Metoda vrací logickou hodnotu, zda má mobil přístup k serveru.

6.4.2.2 Komunikace s chytrými hodinkami

Ke komunikaci mezi chytrými hodinkami a mobilem slouží MessageAPI, které umožňuje posílání zpráv. Každá zpráva obsahuje identifikátor a obsah. Poslání takové zprávy je realizováno metodou sendMessage.

6.5 Modul wear

Tento modul obsahuje aplikaci pro chytré hodinky.

6.5.1 MainActivity

Tato třída se stará o vykreslení úvodní obrazovky, která obsahuje seznam všech zařízení, která lze ovládat. Po kliknutí na vybrané zařízení se spustí nová aktivita, která otevře novou obrazovku s detailem zařízení. v závislosti na typu rozhraní vybraného zařízení se spustí jedna z následujících aktivit:

ShowDeviceNumActivity Třída reprezentuje aktivitu, která vykreslí rozhraní pro vložení celého čísla vybraného zařízení.

ShowDeviceStActivity Třída reprezentuje aktivitu, která vykreslí rozhraní pro vložení logické hodnoty vybraného zařízení.

ShowDeviceYnActivity Třída reprezentuje aktivitu, která vykreslí rozhraní pro spuštění procesu vybraného zařízení.

6.5.2 ConnectionHelper

Tato třída obsahuje metody pro odesílání a zpracování zpráv mezi hodinkami a mobilem. Po přijetí zprávy obsahující nové informace o ovládaných zařízeních se tato zařízení uloží pomocí metod třídy Helper z modulu myCommonLibrary a následně je překreslena aktuální spuštěná aktivita. Pokud příchozí zpráva obsahuje informaci o ztrátě spojení se serverem, tak se tato informace projeví na hlavní stránce tím, že bude vypsána hláška o ztrátě spojení. Dokud nebude spojení se serverem obnoveno, tak není možné odesílat dotazy na změnu stavu zařízení.

Kapitola 7

Testování

Tato kapitola je věnována testování prototypu aplikace. Díky testování byly nalezeny některé chyby, které byly následně odstraněny.

7.1 Použité technologie

7.1.1 Postman

Jedná se o nástroj s graficky uživatelským rozhraním, který je schopen odesílat různé typy dotazů na server.

Při odesílání dotazu se zadá adresa serveru, metoda a parametry. Výsledek, který nám odešle server zpět, lze zobrazit ve více formátech jako je např. JSON, XML, HTML.

Tento nástroj byl použit při testování částí serverové aplikace.

7.2 Testování serverové aplikace

Testování serverové části bylo z hlediska správné funkčnosti velice důležité, protože tato část obsahuje logickou a datovou vrstvu. Testování probíhalo manuálně pomocí již zmíněného programu Postman. Testovány byly následující dotazy:

- spárování chytrého zařízení
- spárování ovládaného zařízení
- vložení požadavku na změnu hodnoty zařízení
- nastavení nové hodnoty zařízení
- zjištění aktuálních informací o zařízení
- zjištění seznamu ovládaných zařízeni spolčně s jejich hodnotami

7.3 Testování aplikace pro android

Při vývoji aplikace pro telefon a chytré hodinky bylo jako IDE použito Andoid Studio. které umožňuje zobrazit výstup aktuálně běžících programů v zařízení. Tímto způsobem byly programy odlazeny. Po nasazení serverové aplikace na server byla aplikace otestována manuálně v provozu.

Kapitola 8

Další vývoj

V budoucnu by bylo možné tuto technologii vylepšit o celou řadu vylepšení, která nebyla zařazena do návrhu. Zde je uvedeno několik příkladů vylepšení:

Sekvence událostí Jedná se o funkčnost, která zajistí, aby se spustila sada příkazů pro zvolená zařízení. Uživatel by si mohl vybrat konkrétní zařízení, nastavit jejich hodnoty a tuto množinu uložit jako sekvenci. Spuštěním sekvence by se odeslaly nastavené hodnoty těchto zařízení ke zpracování na server.

Časové ovládání Uživatel by mohl nastavit u zařízení nejen hodnotu, na kterou se má zařízení nastavit, ale i čas, ve který má být uvedená hodnota nastavena.

Ovládání pomocí slovních příkazů Jelikož mají chytré hodinky zabudované rozpoznávání řeči, tak by dalším rozšířením mohla být možnost ovládat zařízení pomocí slovních příkazů. Uživatel by pouze řekl, které zařízení chce na jakou hodnotu nastavit.

Závěr

Cílem práce bylo analyzovat současná řešení, požadavky, navrhnout a implementovat jednotlivé části technologie, která umožní uživateli ovládat vzdálená zařízení pomocí chytrých hodinek. Následně byla technologie otestována a vytvořena dokumentace. Dokumentaci lze nalézt na přiloženém médiu. Posledním cílem bylo zhodnotit výhody a nevýhody zvoleného řešení.

V rešeršní části byla popsána současná podobná řešení, kterými jsou chytré domácnosti. Čtenář byl také seznámen se základními charakteristikami chytrých hodinek.

V analytické části je sepsáno zvolené řešení spolu s typy rozhraní, jež slouží k ovládání jednotlivých zařízení. V této kapitole jsou také umístěny funkční a nefunkční požadavky, které jsou zde zohledněny při tvorbě případů užití.

Následující kapitoly se zaměřují na realizaci jednotlivých částí aplikace. Jsou zde zmíněny použité technologie, které byly více či méně použity při tvorbě této práce.

Na základě vytvořeného návrhu byla vytvořena prototypová aplikace, jenž je schopná ovládat zařízení. Důraz při tvorbě této aplikace byl kladen na jednoduché ovládání zařízení s různými rozhraními, což se povedlo splnit.

Slabou stránkou této práce bylo testování, kterému bylo věnováno relativně málo času vůči ostatním částem.

Součástí výsledného produktu je také vytvořená serverová aplikace, se kterou komunikuje aplikace běžící na chytrých hodinkách a jednotlivá zařízení. Pro testovací účely bylo také vytvořeno několik virtuálních zařízení.

Všechny stanovené cíle se úspěšně podařilo splnit.

Literatura

[1] Shanklin, W.: 2016 Smartwatch Comparison Guide. září 2016. Dostupné z: http://newatlas.com/smartwatch-comparison-2016/45642/

[2] Mushtaq, N. U.: Smart Home. říjen 2016. Dostupné z: http://cctvinstitute.co.uk/smart-home/

[3] Mims, C.: What I learned from researching almost every single smart watch that has been rumored or announced. červen 2013. Dostupné z: https://qz.com/102646/takeaways-from-every-single-smart-watch/

[4] Your heart rate. What it means, and where on Apple Watch you'll find it. březen 2017. Dostupné z: https://support.apple.com/en-us/HT204666

[5] Frajer, T.: NEJLEPŠÍ CHYTRÉ HODINKY SE SYSTÉMEM ANDROID WEAR KTERÉ KOUPÍTE V ČR. prosinec 2014. Dostupné z: http://www.androidtip.cz/top-5-nejlepsi-chytre-hodinky-se-systememem-android-wear-ktere-koupite-v-cr/

[6] Woodford, C.: Smart homes and the Internet of Things. říjen 2016. Dostupné z: http://www.explainthatstuff.com/smart-home-automation.html

[7] Dolejš, J.: Google Home: Chytrá domácnost na dobré cestě. leden 2017. Dostupné z: https://www.svetandroida.cz/google-home-recenze-201701

[8] Martin, T.: How to add multiple accounts to Google Home. duben 2017. Dostupné z: https://www.cnet.com/how-to/how-to-add-multiple-accounts-to-google-home/

[9] Javůrek, K.: Burger King pomocí hlasu z televizní reklamy cíleně „hacknul" chytrý domácí reproduktor Google Home. duben 2017. Dostupné z: https://connect.zive.cz/bleskovky/burger-king-pomoci-

hlasu-ztelevizni-reklamy-cilene-hacknul-chytry-domaci-reproduktor-google-home/sc-321-a-187170/default.aspx

[10] Dolejš, J.: Philips Hue: Vybudujte si chytrou domácnost již dnes. listopad 2016. Dostupné z: https://www.svetandroida.cz/philips-hue-recenze-201611

[11] Javůrek, K.: Amazon Echo: chytrý reproduktor do domácnosti. listopad 2014. Dostupné z: AmazonEcho: chytrýreproduktordodomácnostiVícena:http://www.zive.cz/clanky/amazon-echo-chytry-reproduktor-do-domacnosti/sc-3-a-176069/default.aspx

[12] Gebhart, A.: Google Home vs. Amazon Echo. duben 2017. Dostupné z: https://www.cnet.com/news/google-home-vs-amazon-echo/

[13] Firmy bojují o ovládnutí internetu věcí. Uživatel může prodělat. březen 2015. Dostupné z: http://technet.idnes.cz/souboj-internetu-veci-iot-097-/hardware.aspx?c=A150316_161724_hardware_vse

[14] Meola, A.: Firmy bojují o ovládnutí internetu věcí. Uživatel může prodělat. prosinec 2016. Dostupné z: http://www.businessinsider.com/what-is-the-internet-of-things-definition-2016-8

[15] Cockburn, A.: *WRITING EFFECTIVE USE CASES*. Addison-Wesley, 2001, ISBN 02-017-0225-8. Dostupné z: http://alistair.cockburn.us/get/2465

Příloha **A**

Seznam použitých zkratek

IoT Internet of Things

MVC Model-view-controller

GUI Graphical user interface

JSON JavaScript Object Notation

XML Extensible markup language

HTML HyperText Markup Language

IDE Integrated Development Environment

API Application Programming Interface

Příloha B

Obsah přiloženého média

- `readme.txt` ..... stručný popis obsahu
- `exe` .....
  - `android` ..... adresář se spustitelnou formou implementace
  - `device` ..... adresář se spustitelnou formou implementace
- `src`
  - `android` ..... zdrojové kódy implementace
  - `device` ..... zdrojové kódy implementace
  - `server` ..... zdrojové kódy implementace a spuštění serveru
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF