Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

Diplomová práce

Aplikace IQ House – ovládání inteligentního domu pomocí zařízení iPad

Plzeň, 2013 Jan Bouda

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 13. května 2013 Jan Bouda .....

Abstract

IQ House Application – intelligent building managing for iPad

Intelligent living is quite new branch of architecture, energy efficiency and mainly IT. There are few standardized products from companies with world wide range, but there are also small companies with research and development department creating their own products. Thus, every product needs unique approach of control.

Main goal of this thesis is to create application for Apple devices, such as iPhone and iPad, allowing managing intelligent house system IQ house. IQ house is a product of Pilsen Company NetPro Systems s.r.o.

As always, there are more options for application architecture. In final application, usability and user-friendly GUI is the most important for end final user (costumer).

Obsah

1. Úvod

Trend inteligentního bydlení zažívá v posledních letech velký rozmach nejen na českém trhu, který je v tomto směru ještě zcela nenasycený. Zároveň je ale rozšíření povědomí mezi zákazníky bez technického vzdělání a představ dosti složitá. Když si například jdeme koupit auto, máme jasnou představu o svých požadavcích a dokážeme i odhadnout koncovou částku. Co si ale představit pod pojmem inteligentní bydlení? Ve stručnosti inteligentní bydlení umožňuje ovládat ve vlastním bytě či domě téměř všechno elektronické vybavení. A nejde jen o samotné ovládání, ale také o měření, regulaci a zabezpečení s možností vzdáleného přístupu k osvětlení, vytápění, spotřebičům, žaluziím a prvkům zabezpečení. Stačí jen mít inteligentní instalaci – tedy potřebný hardware a spolehlivý software, který je dostatečně jednoduchý a intuitivní, aby ho rády používaly všechny věkové kategorie od dětí po seniory, či osoby s omezenou schopností pohybu.

Finanční náročnost se vždy odvíjí od rozsáhlosti instalace a vybraného dodavatele (řádově od deseti, do čtyřiceti procent celkových nákladů na elektroinstalaci v objektu).

Zároveň s rozšířením chytrých mobilních zařízení, jako jsou smartphony a tablety od firmy Apple a jiných, vzrostla i poptávka pro optimalizované aplikace pro tato zařízení, jelikož ovládat vše přes webové stránky je někdy komplikované a nepohodlné.

Cílem diplomové práce je analyzovat aktuální nabídku těchto aplikací v AppStore a navrhnout novou mobilní aplikaci pro ovládání inteligentního systému IQ house(C) podle aktuálně používané webové aplikace IQ House.

Zadání diplomové práce vzniklo ve spolupráci s plzeňskou firmou NetPro Systems s. r. o. (www.netpro.cz). Výsledkem bude program, který se dostane k většímu množství koncových zákazníků (vlastníků a správců inteligentních domů a bytů).

2. Analýza dostupných aplikací

Cílovou platformou aplikace, která vznikne v rámci diplomové práce je *iOS* [1]. Pro vytvoření přehledu, jaké funkce nabízejí podobné aplikace ostatních dodavatelů inteligentní instalace, jsem zvolil distribuční bod aplikací pro zařízení od firmy Apple - *AppStore*. AppStore je možné navštívit pomocí specializované aplikace *AppStore* (pokud vlastníme zařízení od firmy Apple), pomocí programu *iTunes*, nebo přes webové rozhraní.

2.1. Aplikace dostupné na marketu AppStore

Pro vytvoření přehledu aplikací konkurenčních firem jsem prošel výsledky vyhledávání v AppStore. Pro vyhledávání jsem použil klíčová slova: *smart/intelligent house, smart/intelligent building, smart/intelligent living*. Všechny zkoumané aplikace pro řízení inteligentních instalací byly ke stažení zdarma.

Porovnávané výsledky a subjektivní hodnocení uživatelské přívětivosti aplikací, které prakticky slouží stejnému účelu, ač ovládají často vzájemně nekompatibilní hardware, jsem shrnul do tabulky číslo 1. Různorodost designu a možnosti vyzkoušet si aplikaci v demo verzi (bez vlastnictví vlastního inteligentního objektu) mě překvapila. Pokud aplikace neobsahuje demoverzi, kterou by mohl potencionální zákazník vyzkoušet, považuji to za nedostatek, který může hrát rozhodující roli při výběru dodavatele inteligentní instalace.

Aplikace často pracují s půdorysy domů, nebo grafickými modely místností, které jsou převážně 2D, ale dají se nalézt i ve 3D variantě. Ty jsou podle mého názoru špatně přehledné. Z důvodu využití půdorysů, jak pro nalezení, tak pro indikaci stavu ovládatelných prvků, téměř žádná aplikace nepůsobí na první pohled jako nativní aplikace, ale spíše jako aplikace webová. Programátorské zpracování jednotlivých aplikací ovšem není možné posoudit a také to není cílem této práce.

Uživatelskou přívětivost bylo možné posoudit jen u aplikací, které nabízejí provoz v demo režimu. Testování jsem prováděl na dvou zapůjčených zařízeních *iPhone 3G* a *iPad 2*. Stejná zařízení jsem později využíval k ladění a testování vlastní vyvíjené aplikace.

2.1.1. Mobilní aplikace HAIDY

Mobilní aplikace HAIDY je jediným českým produktem v mém srovnání aplikací. Aplikace má pěkný design, který ale na první pohled působí nepřehledně, z důvodu velkého množství nastavení a možností ovládání. Vše je barevně sladěné, takže po delším užívání se uživatel v aplikaci zorientuje.

Mobilní aplikace obsahuje ovládání kamer bez možnosti zobrazení obrazu (alespoň v mnou testované demoverzi). Umožňuje ale zapnout a vypnout nahrávání kamer. Mobilní aplikace neobsahuje interkom (hlasovou komunikaci s terminálem u vstupu do objektu).

Ovládání aplikace pomocí půdorysu, který je více modelem reality než schématem objektu (viz obrázek číslo 2.1), je podle mého názoru nepřehledné. Některé ovladatelné prvky na půdorysu jsem na první pohled ani nezaregistroval. Otázkou pro mě zůstává, jestli existují „ikony“ pro všechny možné varianty zařízení. Pokud bych měl v domě například klasické topení s rozvodem horké vody a v některé místnosti elektrický přímotop, budou obě varianty na půdorysu vypadat stejně?

Obrázek 2.1 – Mobilní aplikace HAIDY

2.1.2. Mobilní aplikace MyHome (Control4)

Aplikace slouží k vládání systémů Control4, který je kompatibilní se systémem (standardem) KNX. Kompatibilita s takto světově rozšířeným systémem je podle mého velkou výhodou v záruce servisu reálné inteligentní instalace. Více informací o standardu KNX je možné získat na webových stránkách http://www.knx.org/.

Rozložení mobilní aplikace má velice příjemný design. Je uzpůsobena k využívání domácího mediálního centra, přes které je možné sledovat filmy, poslouchat rádio a uloženou hudbu. Ovládání přehrávače je vidět v dolní části obrázku číslo 2.2.

Pro ovládání inteligentního zařízení neobsahuje půdorys, což je hlavní odklon od nepsaného pravidla aplikací pro ovládání inteligentního domu. Ovládání pomocí ostatních vstupních prvků je ale dostatečně intuitivní a přehledné.

Obrázek 2.2 – Mobilní aplikace MyHome

2.1.3. Mobilní aplikace ThinKNX Tester

Mobilní aplikace ThinKNX Tester už ve svém názvu obsahuje odkaz na standard KNX. Jde tedy o aplikaci ovládající hardware tohoto systému.

Aplikace je zajímavě graficky řešena. Velice se mi líbí možnost zobrazit obraz z kamer v inteligentní instalaci. Je navíc možné zobrazit výstup z více kamer najednou. Ovládání dalších prvků je schematické a uživatelsky příjemné.

Zobrazení půdorysu je podle mého názoru zbytečně graficky přeplněné (viz horní část obrázku číslo 2.3). Neaktivní prvky jsou v modelu místnosti reprezentovány ikonami v šedých odstínech. Aktivní prvky jsou zobrazeny jako vybarvené. Pro každého zákazníka je potom nutné vytvořit modely místností na míru a stále je potřeba zvyknout si na rozložení zařízení, aby je nemusel pokaždé v obrázku hledat.

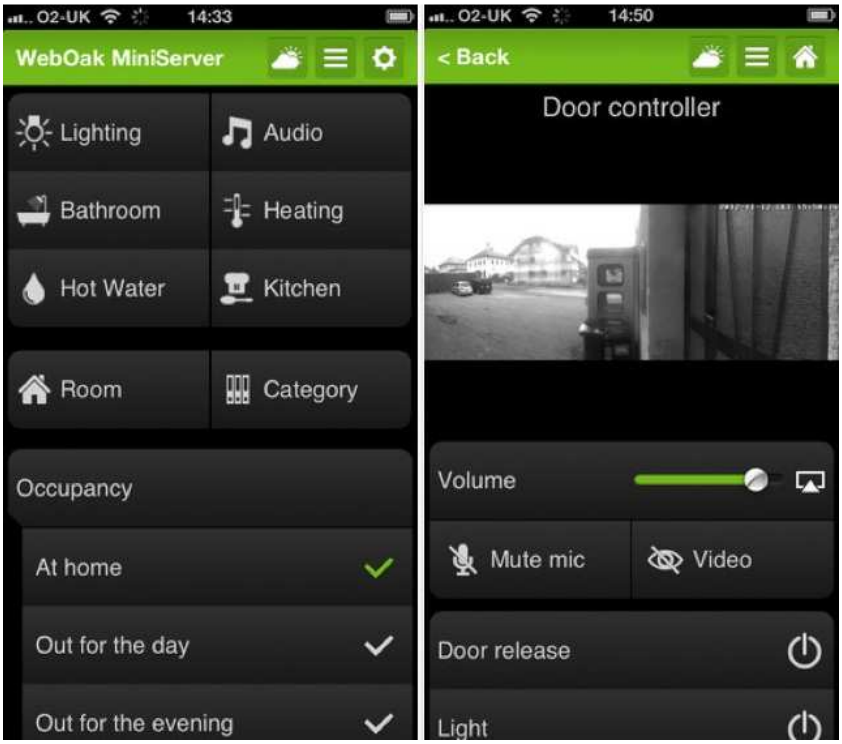
Obrázek 2.3 – Mobilní aplikace ThinKNX tester

2.1.4. Mobilní aplikace Loxone

Mobilní aplikace Loxone slouží k ovládání hardwaru firmy Loxone (centrální jednotkou je Loxone miniserver), která má široké zastoupení v Evropě a USA.

Mobilní aplikace pro zařízení iPhone obsahuje video interkom, který je možné plně ovládat. Ovládání a nastavení jednotlivých prvků je členěno na první pohled nepřehledně, delším užíváním je ale možné zorientovat se ve velkém množství nastavení.

Ovládání prvků je členěno do kategorií bez možnosti zobrazit je na půdorysu domu, případně místnosti. Kategorie jsou rozděleny v menu – viz levá část obrázku číslo 2.4.



Obrázek 2.4 – Mobilní aplikace Loxone

2.2. Shrnutí hodnocení mobilních aplikací

Každá nalezená aplikace umožňuje základní ovládání připojeného zařízení typu zapnout/vypnout – pro osvětlení, vytápění, zabezpečení, zásuvky a spotřebiče. Pro porovnání aplikací jsem proto zvolil následující parametry:

- Uživatelská přívětivost
  - Ovládání zároveň pomocí nabídek i půdorysů
  - Výrazná odezva dotykového ovládání (po použití každého ovládacího prvku na displeji dotykového zařízení následuje grafická či zvuková výstraha)
  - Přehledné rozdělení segmentů aplikace (osvětlení, vytápění a dalších)
  - Jednotný a přehledný design

- Kamery – některé systémy umožňují zobrazení obrazu, jiné jen dovolují zapnout a vypnout nahrávání

- Interkom – domovní interkom je dnes již běžnou součástí inteligentní instalace, jen některé aplikace ale umožňují jeho využití z mobilního zařízení a funkce je tedy dostupná jen na instalovaných zařízeních v domě (terminál u zvonků a terminál uvnitř objektu)

- Konfigurace automatických režimů – automatické režimy představují časování funkcí (vytápění, zabezpečení objektu) a konfigurace kompletních scén, čímž je myšleno kombinace funkcí různých segmentů (scéna „rodinný večer“ by mohla vypadat následovně: rozsvícená světla na intenzitu 60% v obývacím pokoji, zapnuté podlahové topení a zabezpečení garáže). Tyto funkce jsou zřejmě omezovány kvůli větší složitosti a z důvodů bezpečnosti při mylném nastavení časovaných funkcí. Naopak bývají dostupné na pevných terminálech v objektu, případně přes webovou aplikaci.

Přehled hodnocených aplikací spolu s porovnávanými parametry je umístěn v tabulce číslo 1.

Název aplikace	Zdroj	Demo	Ovládaný systém	Kamery	Interkom	Konfigurace automatických režimů
HAIDY	[2]	ANO	vlastní: HAIDY	Ovládání	NE	NE
MyHome	[3]	ANO	Control4 (kompatibilní s KNX)	Obraz + ovládání	ANO	NE
SmartRemote	-	NE	JUNG - KNX	NE	NE	-
iDom	-	NE	KNX	ANO	NE	-
Maestro	-	NE	KNX	ANO	NE	-
BEKO	-	NE	Jen spotřebiče	NE	NE	-
thinknx tester	[4]	ANO	KNX	Obraz	ANO	NE
Loxone Miniserver	[5]	ANO	vlastní: Loxone	ANO	ANO	ANO
iRidium	-	NE	KNX	NE	NE	-

Tabulka 1 - Vybrané aplikace inteligentních instalací

3. Webová aplikace IQ House

Cílem diplomové práce je vytvořit mobilní aplikaci podle vzoru aktuální webové aplikace IQ House, kterou využívají zákazníci firmy NetPro Systems s. r. o. pro ovládání svého objektu. Tato webová aplikace je naprogramována v jazyce Java převážně s využitím frameworku *Vaadin* [6]. Spolupráci na vývoji této webové aplikace, zejména potom optimalizaci pro zobrazení na mobilních zařízeních jsem se věnoval v období před diplomovou prací.

3.1. Funkční analýza webové aplikace IQ House

Systém inteligentní instalace IQ House není dodáván jako unifikovaný krabicový systém a je vždy upraven podle požadavků zákazníka. Každá instalace je tedy do určité míry originální. Od použité inteligentní instalace se následně odvíjí dostupné funkce webové aplikace. Jako vzorová konfigurace aplikace byla vybrána aplikace klienta *npiqhouse*, což je identifikátor inteligentní instalace ve firemním objektu v Plzni (Studentská 50). Na této instalaci probíhá testování nových prvků systému IQ House, vývoj a testování aplikací a prezentace pro zákazníky.

Tato instalace poskytuje uživatelům následující možnosti ovládání:

- Osvětlení – webová aplikace umožňuje ovládat světelná zařízení – zapínat, vypínat a nastavovat intenzitu u svítidla, které to umožňuje.

- Vytápění – topení je možné ovládat pomocí termostatu, který umožňuje manuálně nastavit teplotu a měnit topné režimy (manuální, automatický, temperovat a netopit), které jsou nastaveny v nadřazené aplikaci iqxnet. Zařízení typu topení a teploměr jsou v aplikaci pouze pro informaci. Ukazují, jestli fyzické zařízení topí/netopí a aktuální teplotu na teploměru.

- Zabezpečení – pomocí webové aplikace je možné spustit a zrušit zabezpečení objektu pomocí zadání přístupového kódu. Do kategorie zabezpečení se také počítají pohybová čidla a čidla na oknech a dveřích, která v aplikaci slouží pro zobrazení stavu sepnutí (klid/pohyb a otevřeno/zavřeno).

- Kamery – webová aplikace zobrazuje snímky z kamer získané z aplikace iqxnet, ve které byla původně implementována kompletní funkčnost pro ovládání kamer. V nadřazené aplikaci iqxnet, je možné zobrazit živý přenos z kamery, zapnout a časovat nahrávání obrazu z kamer i pomocí detekce pohybu.

- Grafy a spotřeby slouží k zobrazení vývoje spotřeby energií a teplot v čase. Grafický výstup z dat nadřazené aplikace je zajištěn zásuvným modulem *Google Chart Tools* [7].

Jednotlivá zařízení rozdělená do segmentů je možné zobrazit klasickým výpisem. Dále je ve webové aplikaci definován segment typu půdorys (*Přízemí* – zobrazeno na obrázku číslo 3.1), ve kterém jsou na obrázku půdorysu rozmístěny stavové ikony fyzických zařízení v dané oblasti.

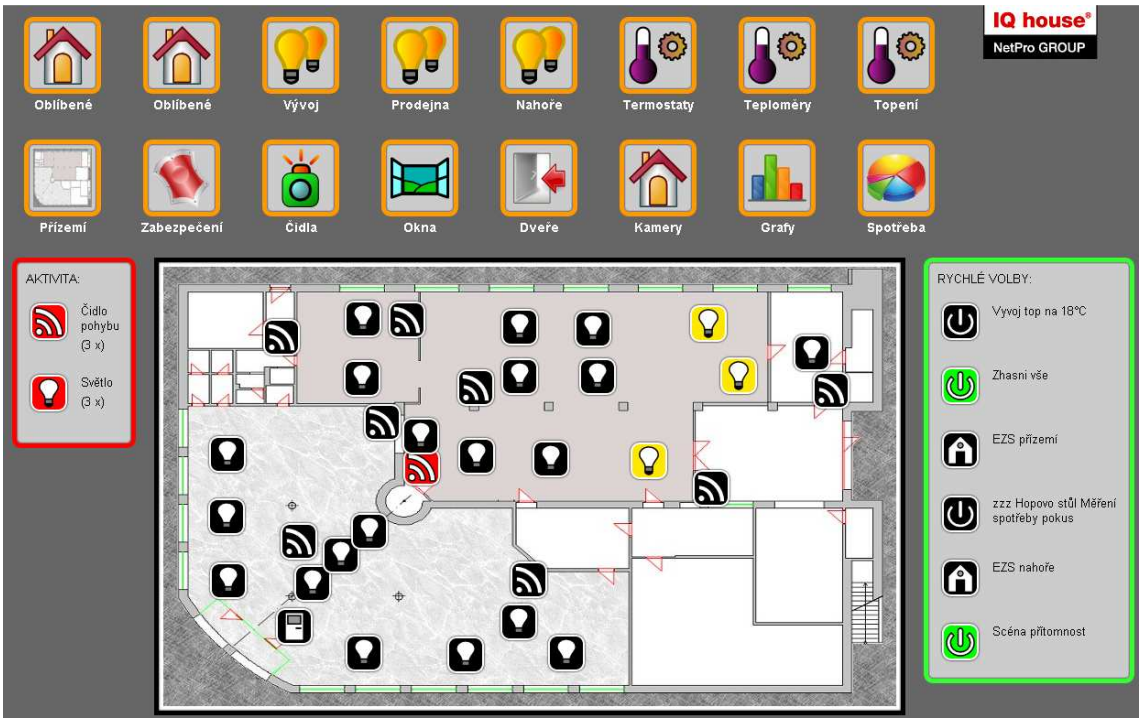
Podle požadavků by měla mobilní aplikace v rámci diplomové práce implementovat všechny výše popsané funkce webové aplikace.

3.2. Rozložení komponent webové aplikace IQ House

Stejně jako funkce webové aplikace je unikátní i obsah grafického rozhraní přístupné webové aplikace. Ta je navržena dostatečně obecně a dynamicky, aby bylo možné podle konfigurace v databázi generovat uživatelsky unikátní webové rozhraní.

Vzhled webové aplikace je rozdělen vertikálně na dvě části. V horní části se nachází menu, zde jsou vypsány segmenty aplikace podle priority. V dolní části se nacházejí samotné ovládací prvky jednotlivých segmentů.

Pokud je v menu vybrán segment typu půdorys (jako na obrázku číslo 3.1), je uprostřed dolní části zobrazen obrázek půdorysu, na kterém jsou rozmístěna tlačítka, jejichž ikona indikuje stav zařízení. V případě vybrání jiného segmentu je ve střední části zobrazena nabídka jednotlivých zařízení s indikací stavů.



Obrázek 3.1 - Webová aplikace npiqhouse

Na levé straně dolní části je zobrazen červený box „Aktivita“, ve kterém jsou zobrazeny aktuálně aktivní prvky v objektu, seskupené podle typu. Kliknutím na ikonu typu aktivního zařízení dojde k výpisu aktivních zařízení ve střední části. Kliknutím na jednotlivé zařízení ve střední části (ve výpisu, nebo půdorysu) dojde k načtení kontroleru zařízení na pravé straně dolní části webové aplikace (zelený box). Kontroler je specifický podle typu zařízení – jinak vypadá kontroler pro osvětlení, jinak kontroler pro vytápění. Některé kontrolery obsahují jen detailnější informace stavu zařízení, například *čidlo pohybu: klid/pohyb*.

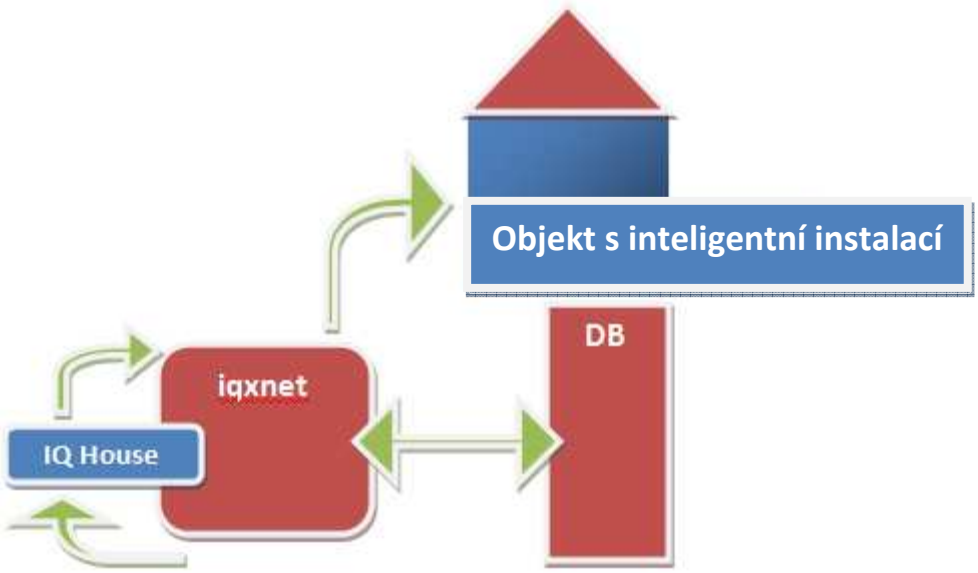
Aktualizace ikon podle aktuálního stavu zařízení je řízena automaticky časovačem, který je nastavený na patnáct sekund. Pokud po delší dobu (tři cykly časovače) neměníme kontroler zařízení, je obsah zeleného boxu nahrazen výpisem rychlých voleb. Rychlé volby jsou předdefinované akce inteligentní instalace, které mohou zahrnovat více zařízení z různých segmentů. Například akce „Zhasni vše“, nebo „Scéna přítomnost“, která zapne světla a vytápění.

3.3. Programátorská analýza webové aplikace IQ House

Jak již bylo zmíněno, GUI webové aplikace IQ House je naprogramováno v javovském frameworku Vaadin verze 6 [6]. Nad tímto uživatelským rozhraním je nadřazena aplikace iqxnet, která plní administrační funkci, obsahuje konfiguraci celého systému a slouží jako databázové rozhraní.

Aplikace iqxnet je naprogramována v jazyce Java, pomocí frameworku Wicket [8]. Pro objektově-relační mapování je použit framework Hibernate. Použitá perzistentní databáze je MySQL. Iqxnet slouží jako webový server, zajišťuje správu cookies, autentifikaci a poskytuje webové stránky podle požadavku (requestu). Aplikace uživatelského rozhraní IQ House s ní pravidelně komunikuje a tím zajišťuje plynulý běh GUI s indikací změny stavu zařízení.

Na aplikaci iqxnet je následně nasměrována kontrolní jednotka, která je pevnou součástí inteligentní instalace v objektu (v domě). Ta si v pravidelných intervalech načítá příkazy z fronty a dále zajišťuje jejich distribuci ke koncovým zařízením pomocí certifikované sběrnice IQ House Bus. Komunikace aplikací je znázorněna na obrázku číslo 3.2.

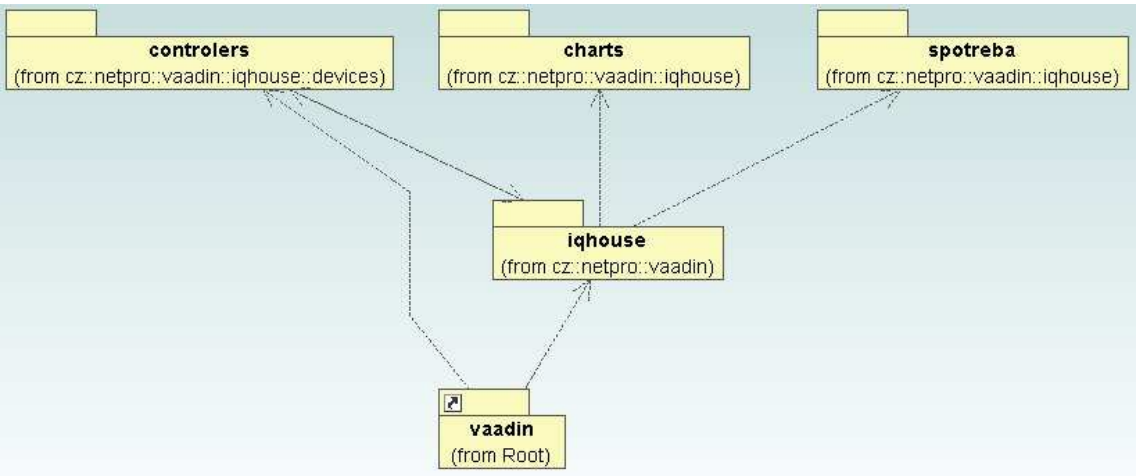


Obrázek 3.2 - Diagram hierarchie aplikace IQ House

3.3.1. Popis tříd webové aplikace IQ House

Framework Vaadin slouží převážně k definici grafického (webového) uživatelského rozhraní pomocí programovacího jazyka Java. Obsahuje definované komponenty a šablony vzhledů, které je možné skládat do složitějších celků pomocí definovaných layoutů. Hierarchie balíků, které obsahují zdrojové kódy webové aplikace napsané ve vaadinu je znázorněna na obrázku číslo 3.3.

Kořenový balík „vaadin“ obsahuje hlavní třídu zajišťující spuštění frameworku (VaadinApplication.java). Dále obsahuje třídu sloužící k řízení aktuálně zobrazeného obsahu (ViewManager.java). Tato třída uchovává instance aplikace pro dvojí rozložení komponent – pro klasické zobrazení (například na stolním počítači) a pro zobrazení aplikace na mobilním zařízení. Třídy pro mobilní zařízení mají podobnou strukturu jako pro zobrazení standardní, proto jsem je z důvodu přehlednosti do diagramů nezahrnul. V případě mobilního rozložení jen přibývá tlačítko „Zpět“ a dochází ke skrývání kontrolerů tak, aby byl na obrazovce viditelný vždy jen jeden aktuálně požadovaný.



Obrázek 3.3 - Webová aplikace IQ House - balíky

Na druhé úrovni se nachází balík „iqhouse“, který obsahuje všechny potřebné třídy pro zobrazení kompletní webové aplikace. Popis vybraných tříd, důležitých pro řízení zobrazování a vzhledu webové aplikace, se nachází na obrázku číslo 3.4.

Tyto třídy jsou dále rozčleněny do dalších balíků:

- Controlers – balík obsahuje kontrolery pro každý typ zařízení užívaného v inteligentní instalaci systému IQ House (světlo, topení, termostat, teploměr, zásuvka, čidlo pohybu a další).

- Charts – balík obsahuje třídy sloužící k vykreslování grafů ze statistik běhu systému IQ House. Pro tuto funkčnost je využíván plugin (ve vaadinovské terminologi *Add-on*) do frameworku vaadin – *VisualizationsForVaadin* [9], který využívá *Google Chart Tools* [7].

- Spotřeba – balík obsahuje třídy pro funkčnost zobrazení spotřeby energií v systému IQ House.

Třída zajišťující layout uživatelského rozhraní vertikálně, tedy rozdělení na menu segmentů v horní části a ovladatelný obsah v dolní části se nazývá *MainLayout.java*.

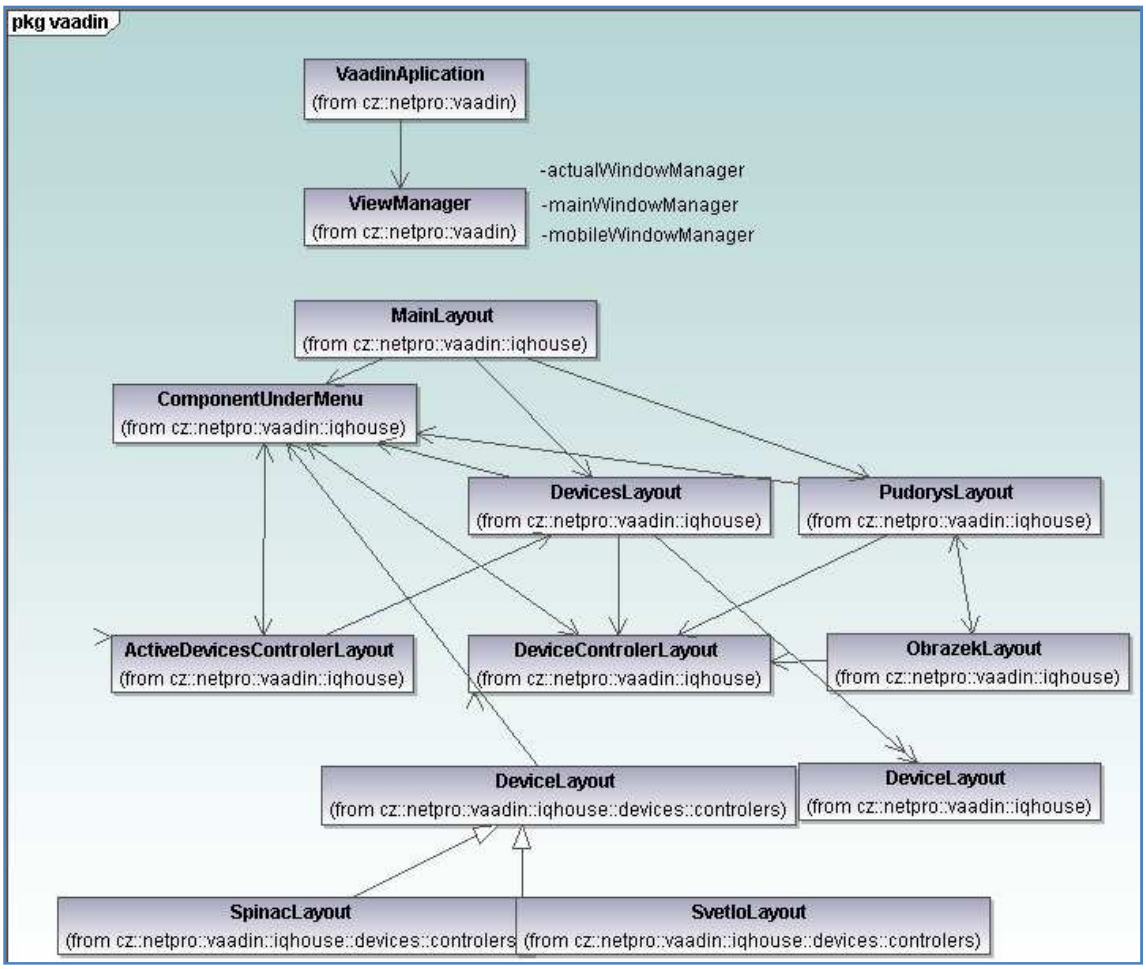
Zobrazení ovladatelného zobrazení v dolní části layoutu stránky dále řeší třída *ComponentUnderMenu.java*, která obsahuje další komponenty:

- *ActiveDevicesControlerLayout.java* – box na levé straně zobrazující aktuálně aktivní zařízení v systému IQ House, zobrazuje jen tlačítka s ikonou typu zařízení a popis s názvem typu zařízení a součtem aktivních zařízení daného typu (*Label*). Po kliknutí na tlačítko dojde k výpisu aktivních zařízení daného typu ve střední části pomocí třídy *DevicesLayout.java*.

- *DeviceControlerLayout.java* – box na pravé straně plní dvojí funkci. Zobrazuje kontroler vybraného zařízení (kontrolery všech typů zařízení jsou potomky třídy *AbstractDeviceControler.java*) a při delší nečinnosti uživatele nahradí kontroler zařízení rychlými volbami. Rychlé volby jsou definovány staticky pomocí tříd *DeviceLayout.java*. Odesílání příkazů pro ovládání fyzických zařízení do nadřazené aplikace iqxnet si zajišťují jednotlivé kontrolery zařízení.

Pokud v menu vybereme segment typu půdorys, je ve střední části zajištěno zobrazení půdorysu (třídou *PudorysLayout.java*), což je potomek třídy *AbsoluteLayout.java* z balíku tříd frameworku vaadin. Do tohoto layoutu je načten obrázek a následně jsou rozmístěny ikony zařízení podle souřadnic definovaných v databázi (tabulka *iqh3segmentDevice*).

Pokud vybereme segment klasického typu, jsou ve střední části vypsány zařízení z daného segmentu pomocí třídy *DevicesLayout.java*, která je potomkem jednoduchého *VerticalLayout.java* z balíku tříd frameworku vaadin.



Obrázek 3.4 - Webová aplikace IQ House - hierarchie vybraných tříd

3.3.2. Popis vybraných databázových objektů webové aplikace IQ House

Na obrázku číslo 3.5 je zobrazen ERA model tabulek vybraných z databáze systému IQ House. Tabulky jsem vybral pro jejich bezprostřední využití ve webové aplikaci IQ House a jejich následnou potřebu při realizaci mobilní aplikace IQ House. Jedná se o tabulky:

- IQH3UnitBlock – vazební tabulka definující, který komunikační kanál inteligentního systému IQ House obsahuje jaký blok proměnných (každý kanál má jen jeden blok proměnných).

- IQH3UnitDef – definiční tabulka zařízení – slouží pro multimediální zařízení (kamery, multimediální server) a virtuální zařízení.

- IQH3DeviceType – číselník typu zařízení, obsahuje kód a název zařízení

- IQH3Device – tabulka shromažďující veškeré zařízení v instalaci inteligentního systému IQ House. Pro zobrazení v aplikacích je důležitá vazba na segment a hodnota sloupce *priorita* – ve webové vaadinovské aplikaci jsou zařízení podle priorit zobrazovány v přehledu zařízení, v rychlých volbách a položce menu „Oblíbené“ (v budoucnu toto rozdělení převezme kompletně příslušnost k segmentu).

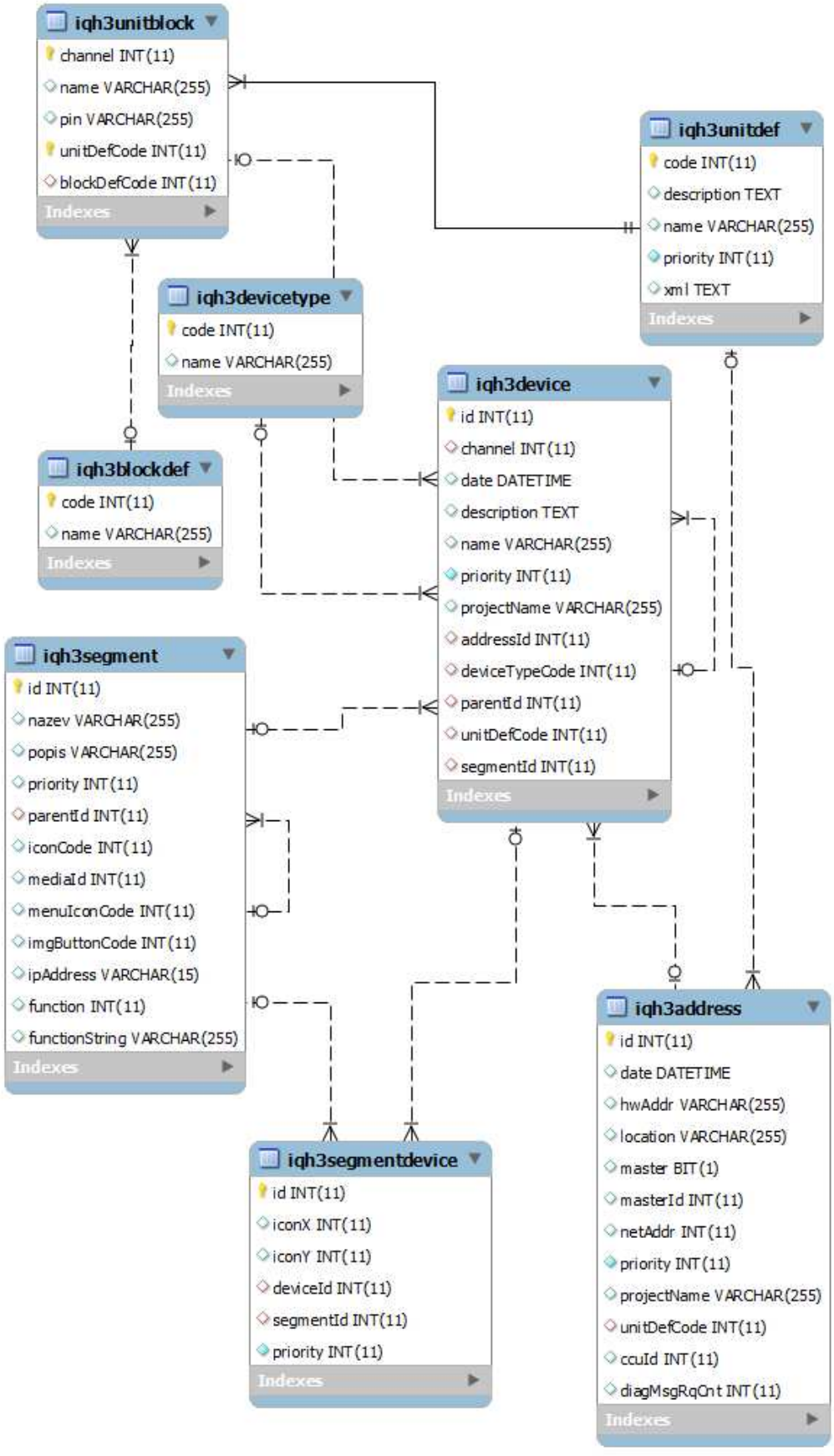
- IQH3BlockDef – číselník typu směrování zařízení. V systému IQ House jsou využívány vstupní zařízení, výstupní zařízení a další, které kombinují vstupní i výstupní přístup (termostat, zabezpečení).

- IQH3Segment – tabulka slouží k rozřazení zařízení do segmentů, které bývá intuitivní podle typu zařízení, případně podle lokalizace (obývák, osvětlení zahrada a podobně). V budoucnu, v rámci optimalizace pro mobilní aplikaci IQ House, budou do segmentů přidány položky oblíbené, rychlé volby a aktivita (vybrané typy zařízení, u kterých chceme hlídat jejich aktivitu). Pokud je u

segmentu vyplněno *imgButtonCode*, jde o segment, který bude zobrazen v menu aplikace. Pokud je vyplněno *mediaId*, jde o segment typu půdorys.

- IQH3Address – tabulka obsahuje adresy (síťové adresy, hardwarové adresy) a lokace všech zařízení v inteligentní instalaci IQ House (vzhledem projektu i fyzické umístění v objektu – například: rozvaděč v prvním patře).

- IQH3SegmentDevice – vazební tabulka typu M:N pro zařízení a segmenty. Obsahuje navíc sloupce *iconX* a *iconY* pro uložení souřadnic vykreslení zařízení v půdorysu.



Obrázek 3.5 - Webová aplikace IQ House - ERA model vybraných objektů

4. Technologie PhoneGap

Pří hledání multiplatformního řešení mobilní aplikace jsem narazil na technologii PhoneGap (http://phonegap.com/). Jde o princip, kdy za použití čistě webových technologií (*HTML*, *CSS* a *JS*) vzniknou webové stránky, které jsou následně sestavené podle nativních pravidel (a případné úpravy nativního kódu) různých platforem do okna *webkitu* (*webview*), který je otevřen spuštěním aplikace.

PhoneGap vznikl jako *opensourse* projekt framework, který je od roku 2011 dotován firmou *Adobe-Nitobi*. Změnami licence zařazením do distribuce *Apache* došlo k definování nového názvu pro distribuci PhoneGapu firmou Adobe – *Cordova*. Cordova podle názvu ulice ve Vancouveru, kde mělo Nitobi kancelář (Cordova Street).

V zaběhnuté terminologii je PhoneGap distribucí projektu Apache Cordova, v praxi se stále hovoří o výsledných produktech vývojářů mobilních aplikací jako o phonegapových aplikacích. Tyto aplikace využívají javascriptové knihovny a knihovny v nativním kódu dané platformy, které mají v názvu slovo „cordova“, případně zkratku *CDV*.

Většina uživatelů nepozná rozdíl od používání graficky nestandardně zpracované nativní aplikace. Aplikaci sestavenou pomocí technologie PhoneGap je možné stáhnout z marketů (AppStore, *Google play* a dalších), na ploše zařízení následně přibude ikona pro spuštění aplikace. Po spuštění aplikace je otevřeno okno pro zobrazení webového obsahu (platformě závislý webkit) bez rámečku a funkcí webového prohlížeče, jako je vyhledávání, tlačítka a pole pro zadávání adresy, a dojde k načtení definované stránky (index.html), která je v aplikaci sestavena (není tedy vyžadováno internetové připojení).

4.1. Minimální požadavky na PhoneGap

Jediným předpokladem pro možnost programování phonegapové aplikace je stažení balíku knihovny Cordova v aktuální verzi ze stránky *http://phonegap.com/download*. Verze, pomocí které je sestavena mobilní aplikace IQ House, je *2.4.0*. Balík po svém rozbalení obsahuje knihovny v nativních kódech všech podporovaných platforem. V dokumentaci k frameworku [10] je dále možné najít

návody, jak vytvořit platformně závislý projekt ve standardním IDE zvolené platformy. Pro platformu iOS jsem popsal vytvoření projektu v kapitole číslo 4.3.

4.1.1. PhoneGap Build

Aplikaci pomocí phonegapu je možné sestavit dvěma způsoby. První a rychlejší možnost je zabalit zdrojové soubory (HTML, CSS a JS) do klasického ZIP archivu a nechat vše ostatní na utilitě „PhoneGap Build“, která je v beta verzi dostupná na adrese *https://build.phonegap.com/.* Po nahrání připraveného archivu dojde automaticky k sestavení aplikace pro všechny dostupné platformy [11]:

- iOS,
- Android,
- Windows Phone,
- Blackberry 5/6/7,
- webOS,
- Symbian.

Druhou možností je sestavit aplikaci pomocí SDK zvolené platformy. Tím navíc získáme možnost přidat další platformě závislé nastavení aplikace. Nevýhodou je, že v tomto případě potřebujeme pro každou platformu extra nejen SDK ale i nativní třídy (a tedy i základní znalost programování v daném jazyce), které zajistí spuštění aplikace a otevření okna (webkitu) s připravenou webovou aplikací. Pro potřeby diplomové práce je nutné mít jen vývojové prostředí Xcode.

V případě řešení aplikace sestavené z webových stránek, naprogramovaných pomocí technologií (HTML, CSS a JS), a z nativního kódu v jazyce Objective-C dojde k vytvoření obecné, rozšiřitelné a znovu použitelné aplikace, která bude s vynaložením minimálního úsilí použitelná na všech požadovaných platformách.

4.1.2. Pluginy pro PhoneGap

Aplikace sestavená pomocí PhoneGapu je standardně jen oknem webkitu. Je sice možné použít nativní kód ve větším rozsahu a webkit s webovou aplikací zařadit jen do určité vyhraněné části GUI. Tím by ovšem phonegapová aplikace ztratila na své jednoduchosti pro webové vývojáře. Převážně pro ně je tato technologie navržena.

Webkit phonegapové aplikace, stejně jako okno webového prohlížeče, nemá běžně z bezpečnostních důvodů přístup k vestavěným funkcím zařízení. U webového prohlížeče to kromě geolokace není ani očekávaná funkce. U phonegapové aplikace je ovšem nutné zajistit fungování podobně jako u aplikace nativní bez nutnosti povolení funkčnosti uživatelem při každém navštívení stránky, která funkčnost vyžaduje. Povolení je odsouhlaseno permanentně jen při instalaci aplikace.

K zpřístupnění fyzických funkcí zařízení phonegapové aplikaci byly vyvinuty nativní pluginy pro každou platformu, které zajišťují spolupráci například s akcelerometrem, fotoaparátem a dalšími funkcemi mobilního zařízení. Plugin tedy slouží jako zprostředkovatel informací a řídí komunikaci mezi javascriptem a nativním kódem phonegapové aplikace. Je také možné naprogramovat si vlastní plugin s libovolnou funkčností.

Přidání pluginů do phonegapové aplikace je nastaveno v souboru *config.xml* aplikace:

```
<plugins>
        <plugin name="Device" value="CDVDevice" />
        <plugin name="Logger" value="CDVLogger" />
        <plugin name="Compass" value="CDVLocation" />
        <plugin name="Accelerometer" value="CDVAccelerometer" />
        <plugin name="Camera" value="CDVCamera" />
        <plugin name="NetworkStatus" value="CDVConnection" />
        <plugin name="Contacts" value="CDVContacts" />
        <plugin name="Debug Console" value="CDVDebugConsole" />
        <plugin name="Echo" value="CDVEcho" />
        <plugin name="File" value="CDVFile" />
```

```
        <plugin name="FileTransfer" value="CDVFileTransfer" />
        <plugin name="Geolocation" value="CDVLocation" />
        <plugin name="Notification" value="CDVNotification" />
        <plugin name="Media" value="CDVSound" />
        <plugin name="Capture" value="CDVCapture" />
        <plugin name="SplashScreen" value="CDVSplashScreen" />
        <plugin name="Battery" value="CDVBattery" />
        <plugin name="Globalization" value="CDVGlobalization" />
        <plugin name="InAppBrowser" value="CDVInAppBrowser" />
    </plugins>
```

4.1.3. Nastavení nativní aplikace pro PhoneGap

Kromě přidání pluginů je i k ostatním nastavením určen soubor config.xml. Hlavním nastavením je povolení přístupu k serverům. Z důvodu bezpečnosti není možné volat javascriptem adresy mimo úložiště aplikace. Mobilní aplikace se ovšem potřebuje dotazovat na server.

Pokud chceme v okně phonegapové aplikace zobrazit cizí webovou stránku, je nutné ji také přidat jako povolený zdroj:

```
<access origin="http://www.iqhouse.cz/npiqhouse/" /> // iqxnet aplikace
klienta npiqhouse

<access origin="http://www.google.com/ " subdomains="true" /> // pridani
povoleni pro externi webovou stranku s povolenim subdomen

<access origin="http://192.168.0.157:8080/" /> // pro testovani na lokalnim
stroji
```

4.2. Přehled verzí

Projekt typu PhoneGap, neboli Cordova, vyžaduje pro své zprovoznění na fyzickém zařízení nainstalovaný operační systém iOS s minimální verzí 5.X. Ostatní požadavky na verzi vyplývají z politiky firmy Apple, kdy je umožněno nahrávat na AppStore jen aplikace vytvořené aktuálním softwarem.

Vzájemná provázanost hardwaru a softwaru, čímž je myšlen operační systém zařízení Mac a vývojového prostředí udává následující minimální požadavky na verze:

- OS X Lion (10.7.4) – operační systém počítačů Mac,

- Xcode 4.5 – vývojové prostředí (IDE) platformy Apple. V případě instalace verze 4.5.2 obsahuje automaticky iOS SDK verze 6,

- iOS SDK 6 – Software Development Kit umožňující programování nejmodernějších aplikací pro mobilní zařízení firmy Apple, jako jsou:
    - iPad (všechny verze),
    - iPhone 3GS a novější,
    - iPod Touch 3rd generation a novější.

4.3. Vytvoření phonegap projektu v Xcode IDE

Programování pro platformu iOS probíhá standardně ve vývojovém prostředí Xcode. V době začátku projektu diplomové práce nebyla v Xcode možnost založit projekt typu PhoneGap – Cordova přímo. Je otázkou, jestli někdy v budoucnu tato možnost bude.

Vytvořit projekt je možné pomocí skriptu z příkazové řádky (terminálu). Následující postup předvádí vytvoření projektu *HelloWorld*.

1) Rozbalíme stažený archiv „phonegap-2.4.0.zip“. Balík obsahuje podsložky určené jednotlivým platformám, pro které je možné sestavit phonegapovou aplikaci.

2) Spustíme aplikaci „Terminál“. Pomocí příkazů „cd“ přejdeme do adresáře rozbaleného balíku knihoven phonegapu: `/phonegap-2.4.0/lib/ios/bin`. Adresář obsahuje skripty pro vytvoření projektu do programu Xcode, který bude mít automaticky přilinkován knihovnu cordova ve verzi pro iOS.

3) Pomocí příkazu „`./create ~/[DST] [PCG] [NAME]`“ vytvoříme nový projekt. Vysvětlení parametrů:
   - [DST] – umístění budoucího projektu,
   - [PCG] – název kořenového balíku projektu,
   - [NAME] – jméno projektu.

   Projekt s názvem HelloWorld je tedy možné vytvořit v dokumentech pomocí příkazu:

```
./create ~/Documents/HelloWorld org.apache.cordova.HelloWorld HelloWorld
```

4) Po provedení příkazu vznikne v dokumentech nová složka HelloWorld s projektovými soubory potřebnými pro vývojové prostředí Xcode. Dvojklikem na soubor s koncovkou *xcodeproj* (HelloWorld.xcodeproj) dojde ke spuštění IDE Xcode a otevření námi vytvořeného projektu. Projekt bude mít strukturu souborů zobrazenou na obrázku číslo 4.1.

Obrázek 4.1 - Struktura projektu HelloWorld

Zahrnutý podprojekt „CordovaLib.xcodeproj“ obsahuje potřebné třídy a soubory pro běh phonegapové aplikace. Obsahuje také třídy vytvořených pluginů, které jsou součástí distribuce.

Dalšími důležitými soubory vygenerovaného projektu jsou:

- Config.xml – konfigurační soubor phonegapové aplikace, slouží k nastavení povolených adres, na které může být phonegapová aplikace směrována, k připojení pluginů (zásuvných modulů) aplikace a k nastavení dalších proměnných.

- www (složka) – po automatickém vytvoření projektu obsahuje jen soubor „index.html“ s HelloWorld textem. Při dalším vývoji projektu zde budou uloženy všechny soubory nutné pro běh webové aplikace (HTML, CSS a JS).

- Classes/MainViewController.m – hlavní view kontroler phonegapové aplikace, zajistí otevření webview na celou plochu zařízení, na kterém phonegapová aplikace běží.

- Classes/AppDelegate.m – hlavní delegát iOS aplikace. Zajišťuje běh aplikace, udržuje instanci webview v okně aplikace. Při startu nastavuje webovou stránku, která bude ve webview otevřena – zde je možné rozhodnout, jaký *index* bude použit v závislosti na zařízení.

5) Spuštěním aplikace v některém ze simulátorů (na výběr jsou iPad a iPhone) pomocí nastavení schématu v horní části IDE dojde k zobrazení stránky *index.html* s výpisem APACHE CORDOVA – Device is ready.

5. Návrh mobilní aplikace IQ House

Cílem diplomové práce je vytvořit mobilní aplikaci pro zařízení iPad. Programování pro zařízení firmy Apple (u mobilních zařízení se jedná o platformu iOS) je podmíněno vlastnictvím stolního počítače, nebo notebooku od stejného výrobce (s procesorem Intel). Pokud už MAC (iMac, nebo MacBook) vlastníme, stačí si zdarma nainstalovat vývojové prostředí Xcode a můžeme začít pracovat. Spolu s IDE Xcode dojde i k nainstalování simulátorů zařízeni iPhone a iPad [1].

Ve chvíli, kdy chceme vyzkoušet program na reálném fyzickém zařízení, je potřeba zaplatit roční vývojářský poplatek a získat identifikační kód vývojáře pro zařízení Apple (v roce 2013 činí poplatek $99) [11].

Vývoj a testování aplikace v rámci diplomové práce následně probíhalo na notebooku MacBook Pro.

5.1. Výběr programovacího jazyka

Vytvoření aplikace pro platformu iOS je zásadě možné pomocí dvou principů. Vytvořit zcela nativní aplikaci pomocí programovacího jazyka *Objective-C* (případně *Objective-C++*), která bude použitelná pouze na platformě iOS. Druhou možností je vytvořit aplikaci v jiném programovacím jazyku, který ovládám a který je na platformě iOS podporován. Taková aplikace by byla využitelná s drobnými úpravami i pro jiné platformy. Mobilní aplikace v jiném než nativním kódu platformy iOS je nutné přeložit a případně doladit tak, aby na zařízeních firmy Apple fungovala bez poznání, že se nejedná o nativní aplikaci.

Jelikož jsem nikdy předtím v Objective-C neprogramoval, snažil jsem se nejprve zjistit možné alternativy, které by navíc splnily přání zaměstnavatele o pokrytí širšího spektra mobilních zařízení, zejména potom těch s operačním systémem *Android*. Pro systém IQ House existuje více aplikací naprogramovaných v různých jazycích a běžících na různém hardwaru (v jazyce Java - Vaadin, v jazyce C - *QT* a další). Vývoj aplikací ve firmě NetPro Systems přibližně kopíruje vývoj nového hardwaru do firemního portfolia inteligentní instalace. Dlouhodobým cílem je proto i sjednocení aplikací. Přidávat další aplikaci jen pro jednu platformu by tedy bylo kontraproduktivní.

Obrázek 5.1 – Kontrolní jednotky systému IQ House - CU a T1000

Hlavní řídící jednotky systému IQ House *T1000* a *CU* (viz obrázek číslo 5.1) mohou fungovat jako webserver a poskytovat HTML kód pro vykreslení webových stránek. Neposkytují ale podporu pro běh programů napsaných v jazyce Java, navíc webovou aplikaci naprogramovanou pomocí frameworku Vaadin už není v plánu dále rozvíjet kvůli netriviální migraci na novou sedmou verzi frameworku. Komponenty uživatelského rozhraní používané frameworkem Vaadin také zvládne nahradit schopný webový návrhář pomocí *HTML5* a *CSS3*.

Při zkoumání možností, jak navrhnout a realizovat mobilní aplikaci, jsem narazil na technologii *PhoneGap*. Tato technologie svým přístupem k vývoji mobilních aplikací zcela splňuje firemní požadavky a zároveň splňuje i zadání diplomové práce. Nejedná se o obecně rozšířenou technologii, i když využívá principů, které jsou ve vývoji mobilních aplikací známé. Více o této technologii je popsáno v kapitole číslo 4.

Z důvodu multiplatformní použitelnosti došlo k rozhodnutí vyvinout mobilní aplikace pomocí této technologie.

5.2. Požadované funkce mobilní aplikace

Hlavním úkolem při implementaci mobilní aplikace bude zajistit na mobilním zařízení stejnou funkčnost, jakou poskytuje stávající webová aplikace. Standardní funkce webové aplikace jsou popsány v kapitole 3.1.

Aplikace musí být navržena dostatečně obecně, aby ji bylo možné v budoucnosti umístit do distribuční sítě platformy Apple – do *AppStore*. Všechny potřebné grafické a

funkční komponenty bude obsahovat sestavené v sobě. Jejich vykreslení do výsledného GUI každého klienta proběhne až po zadání adresy nadřazené klientské aplikace iqxnet a po přihlášení uživatelským jménem a heslem, které bude stejné jako do webové aplikace IQ House. Podle uživatelského jména bude v budoucnu možné nastavit práva a dostupné funkce mobilní aplikace. Tato funkčnost není v rámci diplomové práce uvažována.

Pro implementaci těchto funkcí byly využity nástroje a úpravy popsané níže v této kapitole.

5.3. Výběr webových frameworků a knihoven

PhoneGap využívá pro implementaci GUI webových technologií, mezi které patří HTML, CSS a JS. Po rozhodnutí využít tuto technologii bylo nutné vybrat knihovny, které by usnadnily vývoj kompletní aplikace. Existuje nepřeberné množství hotového a dostupného kódu javascriptových knihoven a kaskádových stylů pro komplexní řešení. Důležité je ověřit jejich vzájemnou kompatibilitu v různých verzích a jejich chování po sestavení phonegapové aplikace.

Frameworky se většinou skládají z javascriptového souboru a ze souboru kaskádových stylů. V tomto případě zde uvádím soubory obou částí samostatně pro snazší orientaci.

5.3.1. Javascriptové frameworky

Hlavním frameworkem, který prostupuje celou aplikací je *AngularJS* [12], který slouží hlavně ke směrování stránek v aplikaci (routování) a k databindingu z přijatých souborů ve formátu JSON. Byl vybrán pro svoje zastřešení firmou Google a rozsáhlostí dokumentace. Navíc je velice snadné serializovat JSON z Hibernate databáze.

Další tři soubory, které nejsou součástí minimalizovaného kódu frameworku AngularJS a je tedy nutné je stáhnout a přidat samostatně jsou *Angular-cookies.js*, *Angular-resource.js* a *Angular-ui.js*.

Dalšími javascriptovými knihovnami, které aplikace využívá, jsou:

- Bootstrap.js [13] – javascriptová podpora pro šablonu grafického rozhraní Bootstrap. V aplikaci je přímo využíván pro zajištění modálních oken.

- Cordova.js [11] – knihovna zajišťující běh phonegapové aplikace v okně webkitu mobilního zařízení (každá platforma vyžaduje jiný javascriptový soubor, pro iOS je to *cordova-ios.js*).

- Hammer.js [14] – framework sloužící pro podporu dotykových akcí pro mobilní zařízení, jako jsou gesta a *multi-touch* akce (obsahuje soubory *hammer.js* a *gestures.js*).

- Jquery.js [15] – tento ve webdesignu hojně využívaný framework je dostupný na adrese http://jquery.com/, do aplikace je zahrnut jako prerekvizita pro knihovnu *jquery-ui.js.*

- Jquery-ui.js [16] – knihovna obsahující komponenty grafického uživatelského rozhraní. V aplikaci je přímo využíván *slider* pro nastavení intenzity osvětlení a teploty na termostatu.

- PHPjs [17] – balík javascriptových souborů, které implementují funkce z jazyka PHP. Vzhledem k nemožnosti použít pro phonegapovou aplikaci PHP funkce by se jistě hodilo využívat více funkcí z tohoto balíku.
    - md5.js – implementace funkce MD5, která v aplikaci slouží k zašifrování odesílaného hesla pro přihlášení ke vzdálené aplikaci iqxnet a pro zašifrování hesla při použití zařízení sloužícího k zabezpečení objektu,
    - utf8_encode.js – prerekvizita funkce md5.js.

5.3.2. Knihovny kaskádových stylů

Další soubory, které aplikace využívá, jsou knihovny pro vykreslení a umístění komponent:

- Bootstrap.css [13] – styl pro komponenty frameworku bootstrap,

- Darkstrap.css [18] – jde pouze o barevnou mutaci komponent z frameworku bootstrap do tmavých barev,

- Jquery-ui.css [16] – soubor stylů pro komponenty grafického webového rozhraní frameworku jquery-ui.

5.4. Úpravy nadřazené aplikace

Tak jako byla webová aplikace IQ House (naprogramovaná převážně pomocí frameworku Vaadin) úzce napojena na nadřazenou aplikaci iqxnet, bude i nová mobilní aplikace vyžadovat komunikaci s aplikací iqxnet. Komunikace s aplikací iqxnet bude nutná v budoucnu, kdy bude webová aplikace IQ House umístěna na stejném serveru jako aplikace iqxnet, i nyní, kdy bude pomocí phonegapu sestavena do samostatné mobilní aplikace. Pro zajištění komunikace s nadřazenou aplikací bude nutné zajistit její úpravu zejména na databázové úrovni a přidat funkci serializace tabulek do formátu JSON.

Vyvinutá mobilní aplikace pro systém IQ House pomocí phonegapu postupně nahradí aktuální webovou aplikaci napsanou v javě (potažmo ve vaadinu) webovou částí nově vznikající aplikace.

5.4.1. Úprava databázové vrstvy

Do databázové vrstvy, která je v aplikaci iqxnet zajištěna objekty a frameworkem hibernate, přibude další knihovna pro serializaci dat do formátu JSON. Touto knihovnou je *Jackson JSON Processor*, která je dostupná na adrese http://wiki.fasterxml.com/JacksonHome. Díky serializačním anotacím je možné do potřebného JSON souboru vybrat jen potřebná data a není nutné vždy exportovat kompletní informace o objektu, což značně zmenší výsledný soubor a urychlí síťovou komunikaci.

Pokud chceme při různých požadavcích (requestech) získávat různé JSON soubory s odlišnými daty, případně s jejich kombinací, je nutné definovat více serializačních pohledů.

Definice pohledu je vlastně prázdná třída v jazyce Java, na kterou se odkazujeme v samotném hibernate objektu. Příklad definice pohledu nad zařízeními IQ House (hibernate objekt IQH3Device):

```
package cz.netpro.zznplj.pojo.iqhouse3.jackson.views;
/**
 * Trida slouzici jako pohed nad zarizenimi IQ House
 * Slouzi prevazne pro tabulku IQH3Device
 */
public class DeviceView {
}
```

Pro potřeby mobilní aplikace budeme potřebovat následující pohledy:

- DeviceView – pohled pro získání *id*, *priority* a dalších informací o zařízení, jejichž součástí je i ikona a popis podle aktuálního stavu zařízení v systému IQ House.

- ActiveDeviceView – pohled pro získání aktuálně aktivních zařízení v systému.

- SegmentDeviceView – pohled pro získání informací z vazební tabulky mezi zařízeními a segmenty. Využívá se pro potřeby půdorysu.

- SegmentView – pohled pro získání segmentů.

- ActionView – pohled pro získání akcí, které je možné se zařízeními provádět. Podle akcí definovaných podle typu zařízení budou následně definovány akce v mobilní aplikaci.

- EmptyView – prázdný pohled využívaný při neplatném požadavku (requestu), aby zbytečně nedocházelo k výběrům z databáze a serializaci dat.

Použití těchto pohledů v hibernate objektu vypadá následovně:

```
@Root
@Entity
/**
 * Hibernate objekt reprezentujici zarizeni v instalaci IQ House
*/
public class IQH3Device implements Serializable, IPriority, IDate {
	@Id
	@GeneratedValue
	@Element(required = true)
	@JsonView({ DeviceView.class, SegmentDeviceView.class })
	private Integer id;

	@Column(nullable = false)
	@Element(required = false)
	@JsonView(DeviceView.class)
	@JsonProperty(value = "pri")
	private Integer priority = 1;
}
```

Důležité řádky jsem zvýraznil červenou barvou. První zvýrazněný řádek s anotací `@JsonView` obsahuje kromě pohledu DeviceView i SegmentDeviceView. Proměnná *id* je tedy serializována pro oba pohledy.

Druhý řádek slouží pro serializaci proměnné id jen pro pohled DeviceView.

Třetí řádek obsahuje další zajímavou anotaci `@JsonProperty`, která slouží k přejmenování názvu proměnné ve výsledném JSON souboru. To je velice užitečná funkce, pokud například název proměnné v hibernate objektu koliduje s klíčovým výrazem, případně jinou proměnnou v javascriptu, který JSON zpracovává.

5.4.2. Poskytování informací ve formátu JSON

Poskytování webový stránek řeší aplikace iqxnet pomocí frameworku wicket. Abychom mohli z mobilní aplikace volat webovou stránku poskytující JSON, musíme do aplikace iqxnet přidat novou třídu oddděním od *WebPage.java* (třídy frameworku wicket) a přidat tuto stránku jako nový modul webové aplikace iqxnet. Přidáním modulu definujeme klíčové slovo, které bude následně rozpoznáno v URL. V našem případě je to klíčové slovo *json*. Volání webové stránky bude tedy pomocí strukturovaného odkazu:

`iqxnet/json/PARAMETR/HODNOTA/PARAMETR/HODNOTA/….`

Zpracování parametrů probíhá následně v nově definované třídě, která je rozpoznána klíčovým slovem modulu – *json*. Podle parametrů dojde k rozhodnutí, jaký JSON pohled nad databází požadujeme. Instanci třídy *ObjectWriter* (z frameworku *Jackson JSON Processor*) je nastaveno mapování na požadovaný JSON pohled. Následně stačí provést klasický příkaz SELECT nad hibernate databází a vrátit v odpovědi (*respond)* serializovaná data.

Ukázka metody pro nastavení mapování na požadovaný pohled:

```
/**
 * Metoda slouzici k nastaveni JSON pohledu mapperu ObjectWriter
 */
private ObjectWriter getObjectWriter(ObjectMapper mapper) {
	// v pripade prazdneho pozadavku
	if (StringUtils.isBlank(dataName)) {
		return mapper.writerWithView(EmptyView.class);
	// v pripade pozadavku na zarizeni
	} else if (StringUtils.equalsIgnoreCase(dataName, "devices")) {
			return mapper.writerWithView(DeviceView.class);
	// v pripade pozadavku na aktivni zarizeni
	} else if (StringUtils.equalsIgnoreCase(dataName, "activedevices")) {
		return mapper.writerWithView(ActiveDeviceView.class);
	// v pripade pozadavku na na segmenty
	} else if (StringUtils.equalsIgnoreCase(dataName, "segments")) {
			return mapper.writerWithView(SegmentView.class);
	// v pripade pozadavku na vazebni tabulku segmentu a zarizeni (pudorys)
	} else if (StringUtils.equalsIgnoreCase(dataName, "segmentDevices")) {
			return mapper.writerWithView(SegmentDeviceView.class);
	// v pripade pozadavku na na akce
	} else if (StringUtils.equalsIgnoreCase(dataName, "actions")) {
			return mapper.writerWithView(ActionView.class);
	}
	// v pripade nevalidniho pozadavku
	return mapper.writerWithView(EmptyView.class);
}
```

Metoda pro odeslání odpovědi na požadavek obsahuje kontrolu přihlášení uživatele, následně provede výběr z databáze (v metodě *getData()*) a výsledné objekty zapíše pomocí JSON mapování do response jako *String*.

```
@Override
/**
 * Metoda prekryvajici odpoved na pozadavek
 * Nastavi format odpovedi (respond), provede kontrolu prihlaseni a vrati
 * pozadovana data, nebo chybovou hlasku.
 */
public void respond(RequestCycle requestCycle) {
	ObjectMapper mapper = new ObjectMapper();
	mapper.disable(MapperFeature.DEFAULT_VIEW_INCLUSION);
	ObjectWriter writer = getObjectWriter(mapper);
	try {
```

```
            requestCycle.getResponse().setContentType("application/json;
charset=UTF-8");
            // kontrola prihlaseni uzivatele mobilni aplikace
            if (!userCheckForJSON.checkUser()) {
                  ArrayList<LoginView> result = new ArrayList<LoginView>();
                  result.add(new LoginView("loginRequired"));
      // navraceni chybove hlasky
      requestCycle.getResponse().write(mapper.writeValueAsString(result));
            } else {
      // navraceni dat
      requestCycle.getResponse().write(writer.writeValueAsString(getData()));
            }
      } catch (JsonProcessingException e) {
                  e.printStackTrace();
      }
}
```

5.4.3. Autentifikace uživatele mobilní aplikace

Aplikace iqxnet obsahuje stránku pro přihlášení a následně si uchovává informace o přihlášeném uživateli pomocí cookies. Aby mohlo přihlašování do mobilní aplikace fungovat samostatně, bylo nutné přidat do stránky poskytující JSON mechanizmus na založení nové cookie po úspěšném přihlášení uživatele. Každý následný požadavek vždy jen projde kontrolou autenticity a vrátí požadovaná data. Pokud nebude v aplikaci iqxnet nalezena validní cookie uživatele mobilní aplikace, bude jako odpověď (respond) odeslán JSON soubor obsahující řetězec „loginRequired“.

```
// v pripade pozadavku na prihlaseni uzivatele mobilni aplikace
if (StringUtils.equalsIgnoreCase(login, "logUser")) {
      String userName = params.getString("name");
      String password = params.getString("pass");
      ArrayList<LoginView> result = new ArrayList<LoginView>();
      if (NPAuthDataSession.get().signIn(userName, password)) {
            // Prida cookies s prihlasovacimi udaji.
            WebResponse response = (WebResponse)
getRequestCycle().getResponse();
            Cookie userId;
            USRUserAuth user = NPAuthDataSession.getUserAuthS();
            userId = new Cookie(USRUserAuth.COOKIE_NAME_USER_ID,
user.getLoginName());
            UserDAO.updateLoginTimeout(user);
            String hash = user.getToken(user.getLoginName());
            user.setLoginHash(hash);
            userId.setPath("/");
            int den = 60 * 60 * 24;
            userId.setMaxAge(den);
            // Ulozeni cookie
            response.addCookie(userId);
            // Ulozeni noveho Login timeoutu a Login hashe do DB.
            HibQU.save(user);
```

```
                result.add(new LoginView("loginOK"));

        requestCycle.getResponse().write(mapper.writeValueAsString(result));
        } else { // v pripade neplatneho prihlaseni je navracena chybova hlaska
                result.add(new LoginView("badLogin"));

        requestCycle.getResponse().write(mapper.writeValueAsString(result));
                cz.netpro.zznplj.modules.users.UserDAO.insertBasicNetProUsers();
        }
}
```

V mobilní aplikaci bude zajišťovat přihlášení a přeposílání cookie uživatele knihovna *angular-cookie*, kterou stačí zařadit do aplikace bez nutnosti dalšího programového kódu.

V javascriptových kontrolerech bude také zahrnut druhý stupeň kontroly přihlášení. Pokud přijde od aplikace iqxnet odpověď, že je uživatel nepřihlášen, dojde k přesměrování na stránku s odhlášením uživatele a dojde ke skrytí komponentů, do kterých jsou bindována data, aby nedošlo k zobrazování zastaralého obsahu.

V případě odchycení komunikace mezi mobilní aplikací IQ House a aplikací iqxnet je teoreticky možné ovládat inteligentní zařízení v domě. Reálně by to ale bylo možné, pouze pokud by narušitel znal identifikátory jednotlivých zařízení a příkazy, které je ovládají. Příkazy pro jednotlivé zařízení v inteligentní instalaci se liší podle typu. Navíc pro ovládání zařízení sloužícího k zabezpečení objektu je nutné zadat další kód (heslo), které je přenášeno zašifrované pomocí funkce MD5. Pro zabránění narušení zvenčí může probíhat komunikace mezi serverem, na kterém běží aplikace iqxnet, a mobilní aplikací pomocí šifrovaného protokolu *https*.

5.5. Rozložení mobilní aplikace

Cílem diplomové práce je navrhnout a realizovat mobilní aplikaci pro zařízení iPad. Aby byla tato aplikace použitelná zároveň i pro zařízení iPhone, je nutné počítat s dalšími úpravami, které se týkají převážně designu a rozvržení aplikace, kdy je k dispozici menší plocha displeje.

V případě návrhu mobilní aplikace pro zařízení iPad a iPhone v nativním kódu je standardním řešením použít dva soubory s koncovkou *XIB*. Tyto soubory slouží k definování GUI iOS aplikace pomocí vývojového prostředí Xcode (pomocí

intuitivního WYSIWYG editoru zakomponovaného do Xcode IDE – *Interface Builder Cocoa Touch* [19]). Pro každé zařízení je tedy definován zvláštní XIB soubor.

V případě řešení mobilní aplikace pomocí PhoneGapu jde vždy o design webové aplikace, která se má v okně webkitu fyzického zařízení zobrazit. V zásadě existují dvě možnosti, jak rozlišit typ zařízení – podle obecného rozlišení displeje a typu (názvu) zařízení.

5.5.1. Rozlišení displeje zařízení pomocí javascriptu

Rozlišení displeje, na kterém aktuálně webová aplikace běží, je možné zjistit pomocí javascriptových funkcí:

```
var viewport = {
    width  : $(window).width(), // funkce pro zjisteni sirky zobrazeneho webu
    height : $(window).height() // funkce pro zjisteni vysky zobrazeneho webu
};
```

Následně je možné načíst jiný soubor kaskádových stylů, případně použít úplně jiné komponenty a javascriptové funkce.

5.5.2. Rozlišení displeje zařízení pomocí direktivy kaskádových stylů

Použití různých kaskádových stylů podle velikosti zařízení je možné také bez javascriptu. Pomocí anotace *@media* dojde nejprve ke zjištění rozlišení zařízení, na kterém aplikace běží. Následně je načten příslušný kaskádový styl:

```
@media only screen and (device-width: 768px) {
  /* pro iPad zarizeni */
}
@media only screen and (min-device-width: 481px) and (max-device-width:
1024px) and (orientation:portrait) {
  /* pouze pro orientaci na vysku (portrait) */
}
@media only screen and (min-device-width: 481px) and (max-device-width:
1024px) and (orientation:landscape) {
  /* pouze pro orientaci na sirku (landscape) */
}
```

5.5.3. Rozlišení displeje podle názvu zařízení

V případě použití nativních tříd v jazyce Objective-C je možné rozpoznat zařízení, na kterém aplikace běží ještě před načtením souboru kaskádového stylu. V takovém případě je možné rozhodnout jaká webová stránka (jaký index.html) bude do webkitu načtena.

Můžeme tedy mít pro každé zařízení připravený *index* soubor, který bude importovat různou sadu kaskádových stylů a rozdílné javascriptové soubory. Tím, že bude použito více souborů, naroste mírně kompletní velikost mobilní aplikace. Na druhou stranu ale odpadne nutnost testovat mobilní aplikaci na všech typech zařízení i po triviální změně v kódu kaskádového stylu a ve zdrojovém kódu.

```
self.viewController.wwwFolderName = @"web"; // nastaveni zdrojoveho adresare
pro phonegapovou aplikaci (iOS)

self.viewController.startPage = @"index.html"; // nastaveni indexove stranky
pro phonegapovou aplikaci (iOS)

// v pripade ze jde o zarizeni iPhone
if ([[[UIDevice currentDevice] model] hasPrefix:@"iPhone"]) {
    self.viewController.startPage = @"index-phone.html";
}
else{ // v pripade ze jde o zarizeni iPad
    self.viewController.startPage = @"index.html";
}
```

6. Realizace mobilní aplikace IQ House

Mobilní aplikace IQ House vznikla v rámci diplomové práce pomocí technologie PhoneGap. Nativní třídy phonegapové aplikace zůstaly téměř beze změny ve stavu, jak byly vygenerovány pomocí skriptu *create* z knihovny cordova. Jediná změna, která slouží k rozhodnutí jakou webovou stránku *index.html* načíst, je popsána v kapitole číslo 5.5.3. Kód, který byl doplněn za účelem uživatelského pluginu, je popsán v příloze *Vytvoření vlastního pluginu do PhoneGapu*. Tento plugin slouží k perzistentnímu uložení uživatelského nastavení do textového souboru – v rámci diplomové práce je potřeba ukládat pouze URL adresu nadřazené klientské aplikace iqxnet, se kterou aplikace komunikuje.

Tato kapitola slouží k popisu webové části aplikace, která celá vznikla v rámci diplomové práce.

6.1. Funkce mobilní aplikace

Ovládání jednotlivých zařízení v mobilní aplikaci je řešeno pomocí modálních oken z pluginu Bootstrap [13] (v případě zařízení iPhone – v případě iPadu jsou některé akce zastoupené tlačítky přímo ve výpisu zařízení).

Obsah modálního okna je vždy před otevřením nastaven podle typu zařízení v kontroleru. Jednotlivé akce, které je se zařízením daného typu možné provádět jsou definovány v nadřazené aplikaci. Na straně mobilní aplikace jsou následně jen podle typu akce vykresleny příslušné ovládací prvky (pomocí technologií HTML, CSS, JS a potřebných pluginů).

Stav zařízení je v mobilní aplikaci indikován stavovou ikonou a popisem (v půdorysovém zobrazení pouze stavovou ikonou). Aktuálnost zobrazeného stavu hlídá časovač, který jednou za pět sekund provede natažení nového JSON souboru se stavem právě zobrazených zařízení (zařízení ze zobrazeného segmentu) a provede změnu ve *view* příslušného kontroleru.

Akce jsou v nadřazené aplikaci definovány typem front-endového prvku webového rozhraní, který se zobrazí v mobilní aplikaci. V rámci diplomové práce jsou číselníkem vymezeny čtyři typy akce:

- Tlačítko – Z hlediska GUI vždy po stisknutí tlačítka na dotykovém displeji dojde k jeho úplnému probarvení (při základním zobrazení obsahuje světlý gradient). Nadřazené aplikaci je odesíláno ID zařízení a hodnota povelu:
  - Zapnout,
  - Vypnout,
  - Spustit (slouží pro virtuální zařízení, které může definovat kombinace akcí různých zařízení v inteligentní instalaci).
- Slider – nadřazené aplikaci je odesláno ID zařízení, hodnota povelu přepnutí zařízení do manuálního režimu (v případě termostatu) a nastavená hodnota na slideru.
- Klávesnice zabezpečení – nadřazené aplikaci je odesláno ID zařízení, hodnota povelu (spustit/zrušit) a potvrzovací kód zadaný uživatelem na klávesnici.
- Kódovaný obrázek PNG – nadřazené aplikaci je odeslán požadavek na poskytnutí kódovaného obrázku podle IP adresy kamery v inteligentní instalaci – odpovědí je řetězec znaků reprezentující poslední snímek kamery.

Tyto akce jsou následně ve webové části mobilní aplikace rozlišeny *ng-view* podmínkou:

```
<span ng-repeat="action in modalActions"><!-- cyklus pres akce, definovane
v JSON souboru -->

<!-- akce reprezentovana tlacitkem s definovanou barvou a ikonou akce -->
  <span ng-show="{{action.type}} == 1">
    <button ng-click="command({{modalDevice.id}}, '{{action}}')" ng-
class="button" class="btn" style="background-image: -webkit-linear-
gradient(top, #FFFFFF, {{action.color}}); background-image: linear-gradient(to
bottom, #FFFFFF, {{action.color}}); background-color: {{action.color}}; ">
       <img alt="" src="img/dev/{{action.icon}}.png" />
                  {{action.name}}
    </button>
</span>

<!-- akce reprezentovana sliderem intenzity pro nastaveni osvetleni -->
<div style="margin-top: 10px;" ng-show="{{action.type}} == 2">
```

```
      <div style="padding: 10px; border-width: 1px; border-style: solid;
border-color: white; border-radius: 5px;">
            <div style="margin-bottom: 10px;">{{action.name}}: <input
type="text" readonly="readonly" ng-model="action.min"> </div>
            <div intensity-
slider="{'min':{{action.min}},'max':{{action.max}}}" ng-model="action"
devid="{{modalDevice.id}}" func="command(dev, action)"></div>
            </div>
      </div>

<!-- akce pro zadani zabezpecovaciho kodu – tlacitko, ktere otevre dalsi
modalni okno s klavesnici pro zadani kodu -->
  <div style="margin-top: 10px;" ng-show="{{action.type}} == 3">
      <button ng-click="openModalNumberInput({{modalDevice}}, {{action}});"
ng-class="button" class="btn" style="background-image: -webkit-linear-
gradient(top, #FFFFFF, {{action.color}}); background-image: linear-gradient(to
bottom, #FFFFFF, {{action.color}}); background-color: {{action.color}}; ">
            <img alt="" src="img/dev/{{action.icon}}.png" />
                              {{action.name}}
      </button>
  </div>

<!-- akce reprezentovana sliderem intenzitz pro nastaveni pozadovane teploty
na termostatu -->
<div style="margin-top: 10px;" ng-show="{{action.type}} == 4">
      <div style="padding: 10px; border-width: 1px; border-style: solid;
border-color: white; border-radius: 5px;">
         <div style="margin-bottom: 10px;">
            {{action.name}}
            <br>
            Požadovaná teplota:
            <input type="text" readonly="readonly" ng-model="action.min">
            </div>
            <div intensity-
slider="{'min':{{action.min}},'max':{{action.max}}}" ng-model="action"
devid="{{modalDevice.id}}" func="command(dev, action)"></div>
            </div>
      </div>

<!-- akce definovana pro zobrazeni snimku z kamery -->
<div style="margin-top: 10px;" ng-show="{{action.type}} == 5">
      <div style="padding: 10px; border-width: 1px; border-style: solid;
border-color: white; border-radius: 5px;">
            <div style="margin-bottom: 10px;">
            {{snimekError}}
            <img src="{{snimek}}">
            </div>
      </div>
</div>
```

6.1.1. Osvětlení

Mobilní aplikace umožňuje u každého zařízení typu osvětlení pomocí tří ovládacích prvků nastavit světlo:

- Zapnout – tlačítko pro odeslání příkazu pro rozsvícení nadřazené aplikaci.
- Vypnout – tlačítko pro odeslání příkazu pro zhasnutí nadřazené aplikaci.
- Intenzita – slider pro nastavení intenzity osvětlení (u zařízení, které neumožňuje regulaci je jakákoli intenzita větší než nula brána jako povel k rozsvícení). Po uvolnění doteku slideru je konečná intenzita odeslána nadřazené aplikaci k nastavení.

6.1.2. Vytápění

Vytápění je možné nastavovat pouze pomocí zařízení typu termostat. Zařízení typu topení a teploměr jsou v aplikaci pouze pro informaci o aktuálním stavu zařízení a prostředí. Zařízení typu termostat má v modálním okně čtyři ovládací prvky:

- Automatický režim – stisknutím dojde k zapnutí režimu, který je definován v nadřazené aplikaci iqxnet.
- Temperovat – aktivací režimu dojde k udržování teploty na 5°C.
- Netopit – tlačítko slouží k vypnutí topení.
- Manuální režim – pomocí slideru je možné nastavit požadovanou teplotu.

6.1.3. Zabezpečení

Po kliknutí na tlačítko „Zabezpečit“ nebo „Odzabezpečit“ dojde k otevření modálního okna s klávesnicí pro zadání kódu. Kód je následně odeslán nadřazené aplikaci. Jelikož jde o testovací verzi, není zatím tento přenos kódu šifrován. V dalším rozšíření bude nahrazen bezpečným přenosem do nadřazené aplikace.

6.1.4. Kamery

V rámci diplomové práce je funkcionalita kamer implementována v mobilní aplikaci shodná se stávající webovou aplikaci. Je tedy zobrazen pouze poslední snímek z kamery bez možnosti streamování a ovládání nahrávání.

Snímky z kamer jsou uloženy v nadřazené aplikaci v databázi ve formě bytového pole. Fyzicky soubor s obrázkem, na který by bylo možné z mobilní aplikace pouze odkázat, neexistuje. Aby nebylo nutné ukládat každý snímek v úložišti mobilní aplikace a tak zabírat uživatelský prostor, zvolil jsem postup, kdy je pole bytů na straně nadřazené aplikace zakódováno pomocí *Base-64* algoritmu a následně odesláno ve formě JSON souboru, jako ostatní data. Takto zašifrovaná data je možné vložit do HTML kódu pomocí následující konstrukce. Hodnota {{snimek}} představuje řetězec zašifrovaného obrázku.

```
<img src="{{snimek}}">
```

6.1.5. Grafy a spotřeby

Zobrazení grafů a spotřeb nebylo v rámci diplomové práce řešeno, jelikož se jedná pouze o informační funkcionalitu, kterou by navíc bylo nutné řešit dalším javascriptovým pluginem.

6.1.6. Půdorysové zobrazení

Na rozdíl od snímků z kamer je obrázek půdorysu uložen v nadřazené aplikaci. Pro jeho zobrazení v mobilní aplikaci je pouze nutné zjistit správnou adresu obrázku. Jelikož je webová část mobilní aplikace univerzální a může fungovat samostatně pouze ve webovém prohlížeči, stačí v tomto případě použít relativní URL.

Sestavení URL obrázku je zajištěno javascriptem, kde je do URL stránky s půdorysovým označením vložen parametr *mediaId*, který určuje, který půdorys požadujeme:

```
// sestaveni adresy obrazku pudorysu
if($scope.serverPath == '..'){ // pokud je web spusten na serveru s aplikaci
iqxnet
   var urlRoot =
window.location.href.substring(0,window.location.href.indexOf("web"));
   $scope.pudorysURL = urlRoot + 'cmsres/mediaid/' + $routeParams.mediaId;
}
else{ // v pripade phonegap aplikace
   $scope.pudorysURL = $scope.serverPath + '/cmsres/mediaid/' +
$routeParams.mediaId;
}
```

6.2. Adresářová struktura webové části mobilní aplikace

Kořenová složka obsahující zdrojové soubory se jmenuje „web“. K přejmenování ze standardního názvu phonegapové aplikace (www) došlo pro potřebu integrace do aplikace iqxnet.

Složka obsahuje dva soubory, které představují *index* webové aplikace. Jeden z indexů je načten v případě, že je aplikace spuštěna na zařízení iPad (index.html). Druhý soubor je načten v případě, že je mobilní aplikace spuštěna na zařízení iPhone (index-phone.html).

Popis podsložek webové aplikace:

- css – složka obsahuje soubory kaskádových stylů, soubor „styles.css“, je načítaný souborem „index.html“ - slouží jako styl pro mobilní aplikaci spuštěnou na zařízení iPad a „styles-phone.css“, který je načítaný souborem „index-phone.html“ a slouží jako styl pro aplikaci spuštěnou na zařízení iPhone. Složka dále obsahuje kaskádové styly používaných frameworků a knihoven.

- img – složka obsahuje obrázky využívané v mobilní aplikaci – v kořenové složce je uloženo logo firmy a *favicon* (ikona webové stránky určená pro funkce prohlížeče, jako jsou záložky, historie a další)
    - but – složka obsahuje obrázky využívané v menu aplikace jako tlačítka (buttons),
    - dev – složka obsahuje obrázky využívané jako ikony pro zařízení systému IQ House, obrázky jsou ve více barevných variantách pro různé stavy zařízení (devices).

- js – složka obsahující javascriptové soubory. V kořenové složce jsou uloženy funkce, které spolu s knihovnou AngularJS řídí webovou aplikaci:
    - *app.js* – soubor definuje nastavení modulu pro framework AngularJS, konfiguraci *route provideru* (řídí zobrazování stránek webové aplikace) a první funkci, která je volána při spuštění webové aplikace, obsahující framework AngularJS.

- *app-phone.js* – soubor pro definici modulu pro framework AngularJS – oproti „app.js“ se liší změnou v načítání webových stránek *route providerem.*
- *directives.js* – zde jsou definovány uživatelské direktivy pro framework AngularJS. Direktivy jsou funkce, které pomáhají vázat (bindovat) hodnoty proměnných na prvky GUI a zároveň hlídají změnu ovladatelného prvku – například při posunutí *slideru* teploty odešlou požadovanou teplotu.
- *services.js* – soubor sdružující služby pro modul definovaný v souboru „app.js“. Jde o tovární funkci (*factory*), která usnadňuje definování složitějších funkcí. V mobilní aplikaci se služby využívají k definování požadavků na soubory typu JSON.
    - controlers – složka obsahuje jednotlivé javascriptové kontrolery pro ovládání a bindování dat do GUI. Platí pravidlo, že každá šablona (část webové stránky) z adresáře „tpl“ má svůj vlastní kontroler, který zajišťuje data pro její *scope* (kontext).
    - plugins – složka obsahuje javascriptovou část phonegapových pluginů, vyvinutých v rámci diplomové práce (více v příloze *Vytvoření vlastního pluginu do PhoneGapu*).

- lib – složka obsahuje všechny využívané javascriptové knihovny (tedy jejich javascriptovou část):
    - phpjs – složka obsahuje vybrané funkce z balíku PHPJS.

- tpl – složka obsahuje html *šablony*. Šablona je část stránky, která je pomocí route provideru modulu angularu načítána do webové aplikace, a zároveň je jí přiřazen javascriptový kontroler.

6.3. Popis integrace frameworku AngularJS do mobilní aplikace IQ House

AngularJS, zkráceně angular, je javascriptový MVC framework od firmy Google, který primárně zajišťuje funkci mobilní aplikace IQ House.

MVC je název vzoru softwarové architektury, která odděluje vrstvu prezentující data od uživatelské interakce s nimi. *Model* obsahuje aplikační data. *View* (pohled) může být definován jako jakýkoli výstup zpracovaných dat. Jde většinou o graf, diagram, nebo tabulku. *Controller* slouží ke zprostředkování uživatelského vstupu pro funkce modelu a pohledu, obsahuje tedy logiku fungování aplikace a další funkce [20].

- Model – model je z pohledu angularu reprezentován jakýmkoli objektem předaným do určitého scope kontroleru. V případě mobilní aplikace IQ House jsou to data ve formátu JSON, které jsou získávány z aplikace iqxnet.
- View – pohled je z hlediska angularu zastoupen HTML šablonami. HTML šablony jsou části webové stránky v souboru s příponou „html“, které jsou v mobilní aplikaci IQ House uloženy ve složce „tpl“. Pomocí route provideru jsou šablony vkládány do DOM (*Document Object Model*), do <*div*> elementu označeného direktivou angularu *ng-view*. Následující kód je v mobilní aplikaci IQ House umístěn na stránce „index.html“:

```
<div ng-view></div> <!-- vyuziti direktivy ng-view -->
```

- Controller – kontroler je ve frameworku AngularJS zastoupen funkcí v jazyce JavaScript. Přiřazení kontroleru HTML šabloně z kategorie view probíhá v aplikaci IQ House zpravidla v souboru „app.js“ pomocí route provideru. Při použití šablony je zároveň nastaven kontroler. Kontext kontroleru (scope) je následně přístupný ve všech elementech šablony.

Proměnné angularu mají prefix „$“ kvůli odlišení od programátorem definovaných proměnných v javascriptu. Nejdůležitější, a v mobilní aplikaci IQ House nejvyužívanější, proměnné angularu jsou *$rootScope* a *$scope*:

- $scope - do češtiny přeloženo jako *kontext*, *rozsah*, *prostor* nebo *rámec*. Jedná se prostor pro proměnné a funkce jednotlivých kontrolerů využívaných v aplikaci. Každý kontroler má svůj vlastní scope, v němž definované proměnné a funkce jsou přístupné z HTML šablony, která kontroler využívá.

- $rootScope – hlavní scope modulu definovaném v aplikaci využívající framework AngularJS. RootScope je společný předek všech scope definovaných

v javascriptových kontrolerech webové aplikace. Proměnné v tomto společném scope aplikace jsou viditelné ve všech HTML šablonách (v každém view).

6.3.1. Direktivy angularu

Angular přidává do HTML vlastní direktivy, pomocí kterých zajišťuje fungování aplikace. Hlavní direktivou je „ng-app“. Tato direktiva se zapisuje do tagu „<html>“ a podle hodnoty direktivy zajistí spuštění aplikačního modulu. Pro mobilní aplikaci IQ House vypadá direktiva následovně: *ng-app='“iqhouse“*. Kde iqhouse je název aplikačního modulu, definovaného v souboru *app.js*.

Dalšími používanými direktivami jsou zejména:

- {{*výraz*}} - Použití výrazu (obsahujícího zpravidla název proměnné) {{*název-proměnné*}}, slouží k vložení hodnoty proměnné do HTML kódu v HTML šabloně. Proměnná musí být definována ve scope kontroleru, kterému přísluší daný HTML objekt, jinak se v HTML místo hodnoty objeví text „undefined“.

- ng-model – tato direktiva slouží k nastavení dvoucestného bindingu dat v HTML šabloně. Využívá se například ve formulářích, kdy chceme nastavit hodnotu vstupního prvků z kontroleru a zároveň zjistit hodnotu zadanou uživatelem po odeslání formuláře:

```
<input class="span2" style="width: 100px;" type="text" ng-model="portCU"
placeholder="port"> <!-- vyuziti direktivy ng-model -->
```

- ng-submit – direktiva slouží k definování funkce ve scope, která se má vykonat po odeslání formuláře. Odeslání formuláře se provede standardně pomocí tlačítka:

```
<form ng-submit=“saveUserSettings()“> <!-- vyuziti direktivy ng-submit -->
```

- ng-show – direktiva sloužící k podmíněnému zobrazení části HTML kódu v šabloně. Podobně jako u direktivy ng-model se využívá proměnná ze scope kontroleru HTML šablony, případně proměnná z rootScope:

```
<div ng-show=“logged“>…</div> <!-- vyuziti direktivy ng-show -->
```

- ng-hide - direktiva sloužící k podmíněnému skrytí části HTML kódu v šabloně. Podobně jako u direktivy ng-model se využívá proměnná ze scope kontroleru HTML šablony, případně proměnná z rootScope:

```
<div ng-hide="logged">…</div> <!-- vyuziti direktivy ng-hide -->
```

- ng-repeat – velice užitečná direktiva, která funguje jako cyklus *for-each* v HTML šabloně nad jednotlivými objekty definovanými ve scope kontroleru. Objekty jsou v mobilní aplikaci IQ House většinou čistou kopií přijatého JSON souboru.

  Příklad vykreslení menu mobilní aplikace v HTML šabloně:

```
<div class="menu-buttons" ng-controller="MenuSegmentsCtrl">
  <!-- vyuziti directivy ng-repeat jako cyklu v menu segmentu -->
  <a ng-repeat="menusegment in menusegments"
href="#/{{menusegment.segmentDetailURL}}">
    <!-- vlozeni ikony segmentu do menu -->
    <img src="img/but/{{menusegment.imgButtonCode}}.png">{{menusegment.nazev}}
  </a>
</div>
```

- ng-include – podle svého klíčového slova slouží direktiva k zahrnutí HTML šablony do DOMu webové aplikace. Jde o druhou možnost, jak vložit šablonu do stránky mimo použití route provideru. Zároveň s ng-include je nastaven i kontroler šablony. Z toho vyplývá, že příkaz definuje nový scope, i když jde o zanořený objekt HTML stránky. V mobilní aplikaci IQ House se využívá direktiva ng-include na vložení menu, jehož zobrazení neřídí route provider, a pro zobrazení modálních oken. V kontroleru vnořeného prvku je nutné k hodnotám bindovaných z HTML šablony přistupovat pomocí klíčového slova *this*. Použití klasické konstrukce *$scope*.NAZEV_PROMENNE z důvodu definice nového scope nestačí.

- ng-controller – direktiva slouží k nastavení javascriptového kontroleru přímo v HTML šabloně. Není nutné používat route provider. Kontroler je tak možné přiřadit libovolnému úseku HTML šablony.

6.3.2. Nastavení aplikačního modulu angularu

Aplikační modul angularu je nejdůležitější část mobilní aplikace IQ House (a každé webové aplikace, které využívá framework AngularJS). Modul mobilní aplikace IQ House nese název „iqhouse“.

Modul *iqhouse* je vlastně javascriptová proměnná (pole proměnných a funkcí), která v sobě ukládá všechna potřebná nastavení pro řízení webové aplikace pomocí frameworku AngularJS. Modul je definován v souboru „app.js“.

Ukázka nastavení knihoven, které modul využívá je uvedena v následujícím odstavci. Obsahuje také nastavení *compileProvide*, které slouží k povolení protokolů užitých ve webové aplikaci. Phonegapová aplikace využívá protokol „file“ a „http“. Ostatní protokoly jsou ponechány v nastavení pro další plánované rozšíření aplikace.

```
// nastaveni modulu iqhouse (vycet pouzitych frameworku)
var iqhouse = angular.module('iqhouse', [ 'ngResource', 'ui.bootstrap',
'ngCookies', 'ui' ]).config(
function($compileProvider) { // nastaveni compile provideru
	$compileProvider.urlSanitizationWhitelist(/^\s*(https?|ftp|mailto|file|t
el):/);
});
```

Následující ukázka představuje nastavení route provideru, který řídí jakou HTML šablonu načíst při změněně URL a jaký javascriptový kontroler jí přiřadit.

```
// nastaveni route provideru pro modul iqhouse
iqhouse.config([ '$routeProvider', function($routeProvider, $locationProvider,
$rootScope) {
	$routeProvider.when('/home', { // v pripade pozadavku na stranku HOME
		templateUrl : 'tpl/devicesLayout.html', // nacteni sablony
		controller : DevicesCtrl // prirazeni kontroleru
	});
	$routeProvider.when('/s/:segmentId', { // v pripade pozadavku na stranku
zarizenich jednoho ze segmentu

		templateUrl : 'tpl/devicesLayout.html', // nacteni sablony
		controller : DevicesCtrl // prirazeni kontroleru
	});
	…
	$routeProvider.otherwise({// v pripade pozadavku na nedefinovanou
stranku

		redirectTo : '/' // presmerovat na korenovy adresar webu
	});
} ]);
```

Poslední využité nastavení modulu iqhouse je funkce *run()*. Tato funkce je volána zároveň s načtením aplikačního modulu iqhouse při startu aplikace.

Metoda run obsahuje podmínku, kde je podle protokolu zapsaného v adrese aplikace zjištěno, jestli je webová část aplikace spuštěná pomocí phonegapu jako mobilní aplikace, nebo byla spuštěna pomocí klasického webového prohlížeče pod aplikací iqxnet. Informace o variantě spuštění aplikace je následně uložena do proměnné `$rootScope.phonegap` (true/false).

```
// hlavni funkce modulu iqhouse
iqhouse.run(function($rootScope, $location) {
	// v pripade ze je web zpusten z mobilni aplikace (phonegap)
	if (window.location.href.indexOf("file") == 0) {
		$rootScope.phonegap = true; // nastaveni promenne urcujici, ze se
jedna o phonegapovou aplikaci

	} else { // webova aplikace spustena z prohlizece (lokalne pod aplikaci
iqxnet)
		$rootScope.phonegap = false;
		$rootScope.serverPath = '..';
	}

// sluzba angularu zajistujici broadcast do vsech definovanych kontroleru
	$rootScope.$on('refreshSubmenu', function(event, args) {
		$rootScope.$broadcast('repaintSubmenu', args);
		// nalezeni kontextu citace a jeho nasledne vynulovani
		var e = document.getElementById('counterScope');
		if (e != null) {
			var counterScope = angular.element(e).scope();
			counterScope.resetCounter();
		}
	});
});
```

Metoda dále nastavuje akci po přijetí signálu „refreshSubmenu“ a vynulování čítače pro obnovování informací pomocí JSON. Po zachycení signálu „refreshSubmenu“ rozešle metoda pomocí broadcastu název funkce spolu s argumenty všem definovaným kontrolerům (všem scope) v aplikaci. Ty následně provedou spuštění své funkce, která může být v každém scope definována jinak. Signál s parametrem je pomocí kontroleru vytvořen konstrukcí:

```
// metoda zajistujici vyvolani sluzby modulu iqhouse
$scope.$emit('refreshSubmenu', {
		submenu : 'tpl/menusegments.html' // nastaveni parametru submenu
});
```

6.3.3. Služby angularu

Výraz tovární třída v programátorském významu znamená třídu, která pomocí své instance vytváří instance dalších tříd (objektů) se kterými program pracuje. Tovární třídy jsou často využívaným konstrukčním prvkem převážně kvůli možnosti nastavit parametry nové instance vždy na jednom místě.

V případě frameworku AngularJS se hovoří o službách (anglicky *services*), což jsou vlastně tovární metody, které jsou přidány do modulu angularu. Služba je vytvořena při startu aplikace a je dále možné ji volat odkazem a předávat jí parametry.

V mobilní aplikaci IQ House jsou služby definovány v souboru „services.js“ a slouží k poptávání souborů typu JSON s různým obsahem od aplikace iqxnet. V následující ukázce je vidět definice služby pro získání segmentů. Službě je z kontroleru nastaveno URL, na které požadavek směruje. Parametry požadavku jsou nastaveny přímo ve službě (*data* a *type*).

```
// nastaveni tovarni metody modulu iqhouse (sluzby)
iqhouse.factory('SegmentJsonFactory', function($resource, $rootScope) {
	return $resource(':url/json', {url: '@url'}, { // nastaveni URL sluzbe
		'getSegments' : {
			method : 'GET', // metoda zaslani pozadavku
			params : { // nastaveni parametru pozadavku
				data : 'segments',
				type : 'menu',
			},
			isArray : true // odpoved pozadavku bude pole
		},
	});
});
```

6.3.4. Data binding

Framework angular slouží jako každý framework primárně k usnadnění vývoje a ušetření práce při programování stále stejných konstrukcí. Tímto usnadněním je určitě také data-binding, který je umožněn angularem jako jednocestný i dvoucestný (*one/two way binding*).

K jednocestnému bindingu dat jsou v mobilní aplikaci IQ House využívány služby modulu iqhouse. Například služba *SegmentJsonFactory* je volána z kontroleru *MenuSegmentsCtrl*, který je definován v souboru „menuCtrl.js“ a slouží ke stažení segmentů ve formátu JSON a následnému vykreslení v šabloně „menusegments.html“.

Na zjednodušené ukázce je vidět definování proměnné *request* (nastavení URL pro službu) a definování dvou metod. První slouží k uložení přijatých dat do scope v případě úspěšného provedení požadavku. Druhá slouží k akci v případě neúspěchu požadavku.

```
var request = { // nastaveni promenne pozadavku (prirazeni parametru)
    url: $rootScope.serverPath
};
var data = ""; // pole pro odpoved na pozadavek
data = SegmentJsonFactory.getSegments(request, function($rootScope) {
    $scope.menusegments = data;

}, function() { // chybova hlaska v pripade neuspesneho pozadavku
       $rootScope.message = "Chyba komunikace se serverem, zkontrolujte stav
připojení k síti.";
       });
```

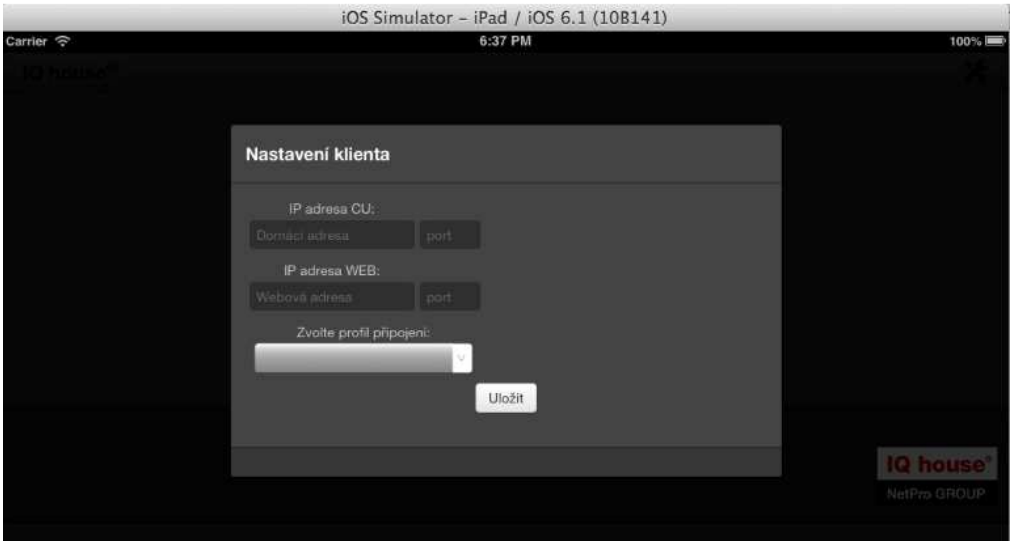
7. Uživatelský manuál a workflow mobilní aplikace IQ House

V této kapitole je popsáno workflow aplikace od prvního spuštění až po komunikaci mezi mobilní aplikací a nadřazenou aplikací iqxnet. Zároveň jsou popsány akce, které se dějí na pozadí bez vědomí uživatele.

Po nainstalování mobilní aplikace IQ House do zařízení iPad a jejím prvním spuštění dojde k zobrazení modálního okna pro zadání uživatelského nastavení. Toto modální okno je znázorněno na obrázku číslo 7.1. Pomocí uživatelského nastavení je zařízeno připojení k nadřazené aplikaci iqxnet, která běží s klientskou instalací na serveru www.iqhouse.cz.

7.1. Uživatelské nastavení klienta

Nastavení je po svém potvrzení uloženo do souboru a při dalším spuštění aplikace se již neobjeví. Dojde jen k načtení nastavení ze souboru do javascriptových proměnných. Nastavení je možné změnit pomocí odkazu „nastavení“ v horní části aplikace. Souborové operace probíhají bez nutnosti potvrzení uživatelem zcela nepozorovaně.



Obrázek 7.1 - Mobilní aplikace IQ House - nastavení klienta

Uživatelské nastavení obsahuje následující vstupní pole:

- IP adresa CU – jde o IP adresu kontrolní jednotky v domě s inteligentní instalací. Kontrolní jednotka také umí poskytovat soubory typu JSON a v případě, že je uživatel doma (v dosahu kontrolní jednotky), je komunikace s kontrolní jednotkou na lokální síti rychlejší než v případě použití webového přístupu.

- IP adresa WEB – IP adresa webová aplikace iqxnet, se kterou mobilní aplikace komunikuje. Tato adresa je využívána v případě, že uživatel není v dosahu kontrolní jednotky (je například v práci). Vyplnění webové adresy je jediným povinným prvkem v nastavení.
    - Port – obě adresy umožňují nastavit port aplikace. Port může být u některých klientů využíván a záleží na nastavení konkrétní instalace systému IQ House.

- Profil připojení – zde je možné nastavit profil připojení. Do budoucna je plánováno rozšířit mobilní aplikaci o funkci detekce kontrolní jednotky v dosahu. Následně bude přepínání profilů připojení automatické. Na výběr jsou nyní následující možnosti:
    - Připojení přes internet,
    - Připojení k CU (doma).

7.2. Přihlášení do aplikace

Po vyplnění uživatelského nastavení nebo po načtení nastavení ze souboru dojde k zobrazení hlavní stránky mobilní aplikace, na které je zobrazen pouze formulář pro přihlášení do aplikace (v případě že je uživatel nepřihlášen). Toto přihlášení je vlastně přihlášení ke komunikaci s nadřazenou aplikací iqxnet. V horním panelu je zobrazena pouze ikona pro zobrazení uživatelského nastavení. Po přihlášení uživatele panel obsahuje další ikony – ikonu pro návrat na domovskou stránku aplikace a ikonu pro odhlášení od nadřazené aplikace iqxnet.

Odeslání formuláře (s chybně vyplněným jménem nebo heslem) pomocí dotazu na aplikaci iqxnet je následováno chybovou hláškou, která přijde ve formě JSON souboru. V případě úspěšného přihlášení k aplikaci je navrácen JSON oznamující úspěšné přihlášení. Zároveň je přijata i nová cookie od nadřazené aplikace iqxnet.

Obrázek 7.2 - Mobilní aplikace IQ House - přihlášení do aplikace

Po úspěšném přihlášení je automaticky odeslán požadavek na segmenty. Po přijetí segmentů v souboru JSON jsou přijatá data uložena do kontroleru a je vykresleno hlavní menu mobilní aplikace spolu se zařízeními ze segmentu „Oblíbené“.

7.3. Menu aplikace

Hlavní menu aplikace je výčet segmentů definovaných v databázi klientské instalace systému IQ House. Pod menu se nachází hlavní ovladatelná část aplikace, stejně jako tomu bylo v případě webové aplikace naprogramované ve frameworku Vaadin. Hlavní rozložení aplikace je na zařízení iPad je znázorněno na obrázku číslo 7.3.

Požadavek na zaslání zařízení jednotlivých segmentů je odeslán při kliknutí na tlačítko menu. Zároveň je odeslán i požadavek na definované akce, které je možné provádět se zařízením. Tyto akce se liší podle typu zařízení, který do jisté míry odpovídá segmentu (což není ale podmínkou).

Přijatá data jsou následně z formátu JSON uložena do kontrolerů a zpřístupněny HTML šablonám.

V případě, že je mobilní aplikace spuštěna na zařízení iPad je odeslán navíc ještě požadavek na aktuálně aktivní zařízení v systému IQ House. Na zařízení iPhone nedochází k zobrazení aktivních zařízení zároveň s výpisem zařízení jednotlivých segmentů, proto tento požadavek nevznikne.

Obrázek 7.3 - Mobilní aplikace IQ House - menu a výpis zařízení

Na levné straně dolní části jsou vypsány aktuálně aktivní zařízení v systému IQ House. V pravé části jsou potom vypsány zařízení ze segmentu. Na obrázku číslo 7.4 jsou zobrazena zařízení ze segmentu „Vývoj“, který obsahuje zařízení z kategorie osvětlení. Přímo ve výpisu zařízení je v mobilní aplikaci možné ovládat zařízeni tlačítky (Zapnout/Vypnout).

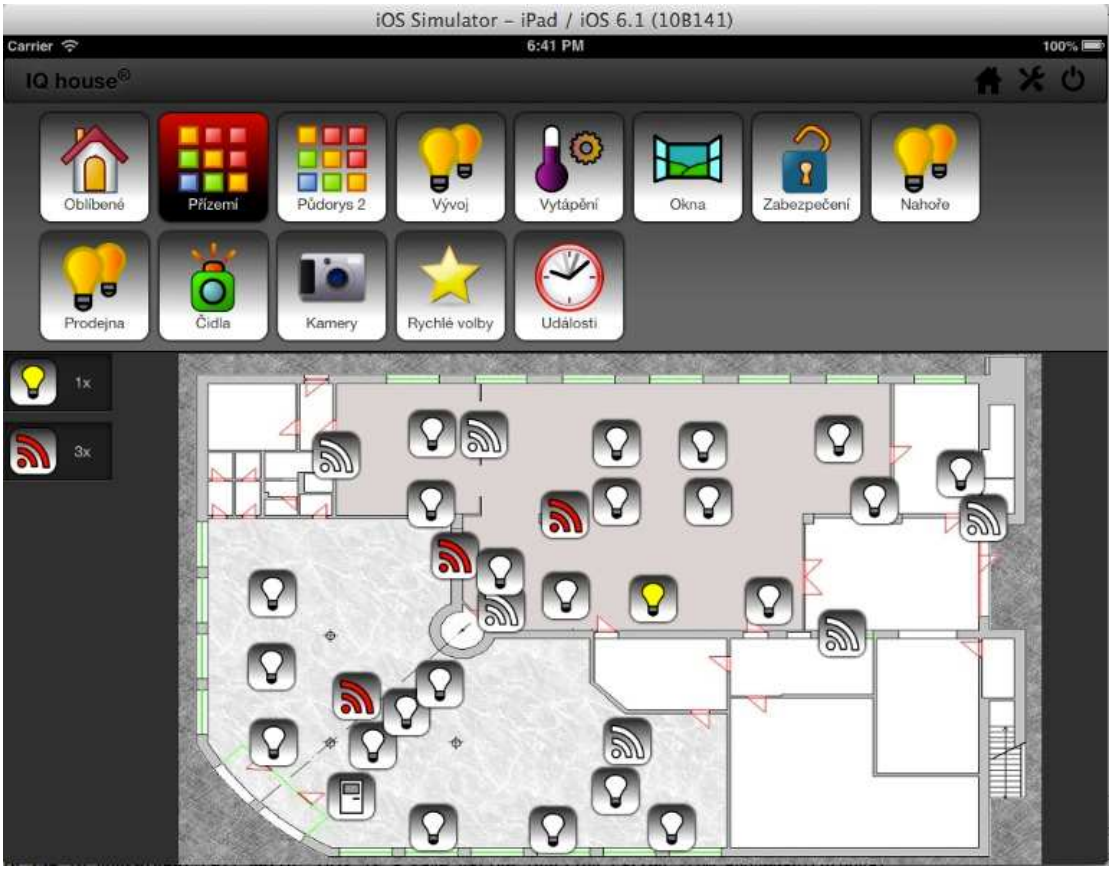
Kontroler zařízení, který byl ve webové aplikaci reprezentován zeleným boxem v pravé části stránky, je v mobilní aplikaci nahrazen modálním oknem. Modální okno

pro ovládání konkrétního zařízení je otevřeno kliknutím na stavovou ikonu zařízení (modální okno je popsáno v kapitole 7.5).

7.4. Půdorys

Pokud v menu zvolíme segment typu půdorys (pro testovaného klienta npiqhouse je to segment s názvem „Přízemí“), dojde k odeslání požadavku na zařízení spolu s vazbou na segment. Z této vazby jsou následně získány souřadnice pro vykreslení zařízení do plochy půdorysu.

Po získání dat z nadřazené aplikace iqxnet dojde nejprve v dolní části k vykreslení obrázku půdorysu a následnému rozmístění stavových ikon zařízení – viz obrázek číslo 7.4.



Obrázek 7.4 - Mobilní aplikace IQ House - zobrazení půdorysu

Ovládat zařízení na půdorysu je možné pouze kliknutím na stavovou ikonu zařízení. Po výběru zařízení je zobrazeno modální okno s možností nastavení zařízení.

7.5. Ovládání zařízení

Ovládání zařízení systému IQ House je v mobilní aplikaci zajištěno modálními okny. Kliknutím na stavovou ikonu zařízení dojde k otevření modálního okna (viz obrázek číslo 7.5), které obsahuje definované možnosti nastavení podle typu zařízení.

Po provedení akce uživatelem je sestaven příkaz, který je následně odeslán do nadřazené aplikace iqxnet ke zpracování. Po změně stavu fyzického zařízení vlivem akce provedené nadřazenou aplikací je mobilní aplikací natažen JSON soubor obsahující aktuální stav zařízení. Změna stavu se následně projeví na stavové ikoně zařízení v mobilní aplikaci.

Obrázek 7.5 - Mobilní aplikace IQ House - modální okno pro ovládání termostatu

8. Závěr

V rámci diplomové práce se podařilo implementovat mobilní aplikaci pro zařízení iPad a iPhone, která byla testována ve firemním objektu s inteligentní instalací IQ House v Plzni. Díky implementaci jsem si značně rozšířil přehled nejen v oblasti web designu, ale i v programování mobilních aplikací.

Webovou část mobilní aplikace je možné využívat i pomocí klasického webového prohlížeče bez nutnosti sestavení aplikace pro konkrétní platformu. Tímto řešením bude v budoucnu nahrazena původní webová aplikace.

Phonegapová aplikace pro platformu iOS byla dále v rámci oborového projektu sestavena pro platformu Android.

V případě rozšíření uživatelského nastavení by bylo dobré předělat aktuální ukládání dat do souboru na využívání datové vrstvy iOS platformy například pomocí knihovny tříd *CoreData*. Výměna informací ve formátu JSON by jistě šla také více optimalizovat. Pro aktuální nároky aplikace je ovšem realizované řešení dostačující.

V době odevzdání diplomové práce jsou v plánu další rozšíření aplikace, kterými bude v první řadě vytvoření uživatelské demoverze aplikace. Demoverze aplikace s imaginárním domem poslouží potencionálním zákazníkům k vyzkoušení systému a věřím, že přispěje kladně k jejich rozhodnutí.

Přehled zkratek

GUI – Graphical User Interface (grafické uživatelské rozhraní)

ZIP – koncovka komprimovaného souboru

HTML – Hypertext Markup Language (značkovací jazyk využívaný pro vývoj webových stránek)

CSS – Cascade Style Sheet (soubor kaskádových stylů)

JS – Java Script (programovací jazyk)

SDK – Software Development Kit (sada pro vývoj softwaru)

CDV – Cordova (produkt komunity Apache, firmy Nitobi)

URL – Uniform Resource Location (webová adresa)

XIB – X Interface Builder (koncovka souboru sloužícího k vývoji GUI na platformě iOS)

WYSIWYG – What You See Is What You Get (označení programu pro designování aplikací pomocí skládání komponent do výsledného vzhledu)

IDE – Integrated Development Environment (vývojové prostředí usnadňující programování)

DOM – Document Object Model (platformě nezávislý standard pro popis objektů)

JSON – Java Script Object Notation (formát objektů navržený původně pro javascriptové funkce)

Seznam obrázků

Použité zdroje

[1] MARK, David and NUTTING, Jack and LAMARCHE, Jeff and OLSSON, Fredrik. *Beginning iOS 6 Development.* Apress, 2013. ISBN 978-1-4302-4512-4.

[2] HAIDY a.s.. *HAIDY | Inteligentní chytrý dům domácnost bydlení* [online]. [cit. 2012-11-20]. Dostupné z: <http://www.haidy.cz>

[3] Control4 Corporation. *Control4 :: Improving the lives of our customers - Home Automation and Smart Home Control* [online]. Datum aktualizace 2012-12 [cit. 2013-01-15]. Dostupné z: <http://www.control4.com>

[4] Thinknx. *Thinknx - Discover ALVEO, the new home automation supervision systém* [online]. Datum aktualizace 2012-08-01 [cit.2O13-01-15].Dostupné z: <http://www.thinknx.com/en/>

[5] Loxone. *Levná domácí automatizace, ovládání domácnosti, inteligentní dům levně - Loxone* [online]. Datum aktualizace 2013-04-09 [cit. 2013-01-16]. Dostupné z: <http://www.loxone.com/pages/cz>

[6] Vaadin Ltd. *Vaadin - thinking of U and I - vaadin.com* [online]. Datum aktualizace 2012-12-02 [cit. 2012-12-02]. Dostupné z: <https://vaadin.com/home>

[7] Google Inc. *Google Chart Tools - Google Developers* [online]. Datum aktualizace 2013-04-03. Dostupné z https://developers.google.com/chart/.

[8] The Apache Software Foundation. *Apache Wicket - Welcome to Apache Wicket* [online]. Datum aktualizace 2012-11-16 [cit: 2012-12-02]. Dostupné z: <http://wicket.apache.org/>

[9] BARRETT, Phil. *VisualizationsForVaadin* [online]. Verze 1.1.2. Dostupné z: https://vaadin.com/directory#addon/visualizationsforvaadin:vaadin.

[10] The Apache Software Foundation. *PhoneGap API Documentation* [online]. Verze 2.4.0 [cit. 2013-02-20]. Dostupné z:

http://docs.phonegap.com/en/2.7.0/guide_plugin-development_index.md.html#Plugin%20Development%20Guide

[11] GHATEL, Rohit and PATEL, Yogesh. *Beginning PhoneGap Mobile Web Framework for JavaScript and HTML5*. Apress, 2012. ISBN13 978-1-4302-3903-1.

[12] Google Inc. *AngularJS — Superheroic JavaScript MVW Framework* [online]. Datum aktualizace 2012-12-1 [cit. 2013-02-13]. Dostupné z: <http://angularjs.org/>

[13] @mdo and @fat. *Bootstrap* [online]. Verze 2.3.1. Dostupné z: http://twitter.github.io/bootstrap/.

[14] TANGELDER, Jorik. *Hammer.js - A javascript library for multi-touch gestures* [online]. Verze 1.0.5. Dostupné z: http://eightmedia.github.io/hammer.js/.

[15] The jQuery Foundation. *jQuery* [online]. Verze 1.9.1. Dostupné z: http://jquery.com/.

[16] The jQuery Foundation. *jQuery UI* [online]. Verze 1.9.2. Dostupné z: http://jqueryui.com/.

[17] Kevin van Zonneveld and contributors. *php.js* [online]. Datum aktualizace 2012-11-26 [cit. 2013-01-22]. Dostupné z: http://phpjs.org/.

[18] NEUMANN, Dan. *Darkstrap* [online]. Verze 0.9.2. Dostupné z: http://danneu.com/bag/darkstrap/darkstrap.html.

[19] BUTTFIELD-ADDISON, Paris and MANNING, Jonathon. *Learning Cocoa with Objective-C, 3rd Edition.* O'Reilly Media, 2012. ISBN 978-1-4493-1849-9.

[20] GREEN, Brad and SESHADRI, Shyam. *AngularJS*. O'Reilly Media, 2013. ISBN 978-1-44934-485-6.

[21] NEUBURG, Matt. *Programming iOS 6, 3rd Edition - Fundamentals of iPhone, iPad, and iPod touch Development*. O'Reilly Media, 2013. ISBN 978-1-4493-6576-9.

Příloha A: Obsah přiloženého CD

- DP – adresář obsahující text diplomové práce (ve formátu PDF a DOC)

- PhoneGap – adresář obsahuje distribuci frameworku PhoneGap (Cordova) ve verzi 2.4.0

- iOS APP – adresář obsahující aplikaci vzniklou v rámci diplomové práce
    - Projekt – adresář obsahující soubory projektu IDE Xcode – obsahuje také nativní třídy pro phonegapovou aplikaci
    - web – adresář obsahující zdrojové kódy webové aplikace (soubory ve formátu HTML, CSS a JS)

- README.txt

Příloha B: Vytvoření vlastního pluginu do PhoneGapu

Pokud není ve frameworku PhoneGap (Cordova) zastoupený plugin, který potřebujeme pro naši aplikaci, je možné vytvořit vlastní plugin. Za plugin je možné považovat jak jednoduchou třídu, tak komplexní balík funkcí. Účel pluginu je vždy stejný – zprostředkování komunikace mezi nativním kódem zařízení (Objective-C) a kódem webové aplikace (JavaScript).

Návody na vytvoření jednoduchých pluginů pro různé phonegapové platformy jsou dostupné na webové adrese [10]. Pro platformu iOS je ovšem návod neaktuální a pro jeho reálné zprovoznění jsem musel dále hledat na vývojářských fórech a provádět úpravy. Proto jsem se rozhodl vytvořit v rámci diplomové práce vlastní tutorial na sestavení pluginu sloužícího k uložení uživatelského nastavení do souboru.

V mobilní aplikaci IQ House slouží plugin k ukládání, načítání a editaci uživatelského nastavení do souboru uloženého v kontextu aplikace (každá iOS aplikace má ve svém kontextu složku pro ukládání dokumentů).

iOS Cordova plugin – nativní kód

Nejprve se zaměříme na vytvoření té části pluginu, která je tvořena nativním kódem v jazyce Objective-C. Vytvoření pluginu předpokládá alespoň základní znalost programování v tomto jazyce.

1) Do složky obsahující hlavní třídu nativní aplikace v jazyce Objective-C (*main.m*) přidáme složku s názvem „Plugins“ (pomocí aplikace *Finder*).

2) Pomocí aplikace Xcode vytvoříme novou třídu s názvem *UserSettingsPlugin*. V levém sloupci vývojového prostředí (*Project Navigator*) klikneme na složku Plugins pravým tlačítkem myši a zvolíme možnost „New File…“. V levé části dialogu vybereme možnost „Cocoa Touch“ (v kategorii iOS), v pravé části dialogu

označíme možnost „Objective-C class“ a klineme na tlačítko *Next*. V dalším kroku zadáme jméno třídy, vybereme, od které třídy chceme oddědit: *Subclass of:* „CDVPlugin“ a klikneme na tlačítko *Next*. V posledním kroku vybereme umístění souborů třídy (bude vytvořen i hlavičkový soubor *UsertSettingsPlugin.h*) a klikneme na tlačítko *Create*.

3) Pro jednoduchý plugin na ukládání, načítání a editaci budeme potřebovat dvě metody. V hlavičkovém souboru [21] si připravíme jejich prototypy. Připravíme si také dva pevné řetězce, které bude plugin využívat: „FILE_NAME“ a „PROPERTY_SEPARATOR“.

```
#import <Cordova/CDV.h>
// nazev souboru pro ulozeni uzivatelskeho nastaveni
NSString *FILE_NAME = @"userConfig.txt";
// format oddelovace uzivatelskeho nastaveni (pro parsovani)
NSString *PROPERTY_SEPARATOR = @"=";

@interface UserSettingsPlugin : CDVPlugin
// metoda slouzici pro cteni souboru s uzivatelskym nastavenim
- (void) loadSettings:(NSMutableArray*)arguments
withDict:(NSMutableDictionary*)options;

// metoda slouzici pro zapis souboru s uzivatelskym nastavenim
- (void) saveSettings:(NSMutableArray*)arguments
withDict:(NSMutableDictionary*)options;

@end
```

4) Ve vlastním souboru třídy doplníme definici metod pro čtení a zápis do souboru. Pro ukázku zde uvedu jen ty řádky kódu, které jsou důležité pro fungování pluginu. Definice cesty k souboru uloženém (pokud neexistuje, dojde k jeho vytvoření) v dokumentech aplikace:

```
// zjisteni cesty k souboru s uzivatelskym nastavenim
NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
NSUserDomainMask, YES);

NSString *documentsDirectory = [paths objectAtIndex:0];

NSString *filePath = [documentsDirectory
stringByAppendingPathComponent:FILE_NAME];
```

Uložení callbackId z argumentů funkce

CallbackId je posláno zpět javascriptu spolu s výsledky metody. Javascript podle něj pozná, jakou metodu nativní kód vykonal:

```
NSString *callbackId = [arguments pop]; // ziskani callbackId z parametru
NSLog(@"Toto je nativni metoda pro cteni souboru - PhoneGap/Cordova!");
```

Odeslání návratové zprávy javascriptu v případě úspěchu a neúspěchu provádění metody:

```
CDVPluginResult *result;
result = [CDVPluginResult resultWithStatus:CDVCommandStatus_ERROR
messageAsString: @"Empty file!"]; // v pripade prazdneho souboru

[self writeJavascript:[result toErrorCallbackString:callbackId]];

result = [CDVPluginResult resultWithStatus:CDVCommandStatus_ERROR
messageAsString: @"Error :("]; // v pripade chyby cteni ze souboru

[self writeJavascript:[result toErrorCallbackString:callbackId]];
```

Spuštění pluginu a řízení výsledků pomocí javascriptu

V druhé části návodu se zaměříme na vytvoření té části pluginu, která je zastoupena javascriptovým kódem – tedy komunikací na straně webové aplikace.

1) Ve složce obsahující javascriptové soubory mobilní aplikace IQ House vytvoříme pomocí aplikace *Finder* složku „plugins“. Ve složce vytvoříme javascriptový soubor *UserSettingsPlugin.js*.

2) Javascriptový soubor musí obsahovat definici pluginu, což je vlastně javascriptové pole proměnných, které obsahuje funkce volající nativní funkce pluginu definované v první části návodu. Volání nativních metod je zařízeno funkcí *cordova.exec()*. Pro potřeby pluginu nám stačí funkce pro volání metod na načtení a uložení dat do souboru.

```
var UserSettingsPlugin = { // definice javascriptoveho pluginu
    // definici volani funkce pro cteni souboru
    callLoadFunction: function (success, fail, resultType) {
            cordova.exec( success, fail, // spusteni funkce
                    "UserSettingsPlugin", // nazev pluginu
                    "loadSettings",  // nazev metody nativniho kodu
```

```
                    [note]); // poznamka
    },
    // definici volani funkce pro zapis souboru
    callSaveFunction: function (success, fail, resultType) {
            cordova.exec( success, fail, // spusteni funkce
                    "UserSettingsPlugin", // nazev pluginu
                    "saveSettings",  // nazev metody nativniho kodu
                    [note]); // poznamka
    }
};
```

3) Javascriptový soubor dále obsahuje funkce, které jsou volané z javascriptových kontrolerů mobilní aplikace. V parametrech funkcí jsou přiřazeny handlery callbacků po vykonání metod nativního kódu. Každá funkce má tedy handler pro úspěšné a neúspěšné vykonání nativního kódu pluginu. Třetím parametrem je řetězec předávaný z javascriptových kontrolerů mobilní aplikace. Pro potřeby pluginu definujeme tři funkce přiřazující handlery pro uložení, načtení a editaci souboru v nativním kódu pluginu:

```
// javascriptova funkce prirazujici handlery obsluzne funkci pro zapis do
souboru
function callUserPluginSave( returnSuccess ) {
    UserSettingsPlugin.callSaveFunction( saveResultHandler, saveErrorHandler,
note );
}

// javascriptova funkce prirazujici handlery obsluzne funkci pro cteni souboru
function callUserPluginLoad( returnSuccess ) {
    UserSettingsPlugin.callLoadFunction( loadResultHandler, loadErrorHandler,
note );
}

// javascriptova funkce prirazujici handlery obsluzne funkci pro editaci
souboru
function callUserPluginEdit( returnSuccess ) {
    UserSettingsPlugin.callLoadFunction( editResultHandler, editErrorHandler,
note );
}
```

4) Jako poslední bod této části pluginu přidáme definici funkcí samotných handlerů. Handlery obsahující v názvu slovo „result“ slouží k obsluze úspěšného vykonání nativního kódu pluginu. Handlery obsahující ve svém názvu slovo „error“ slouží k obsluze chybového stavu nativní části pluginu. V handlerech vykonání nativního kódu (úspěšném i neúspěšném) dochází k vyhledání angular scope a volání metod z nalezeného scope. V těchto metodách je

dále definováno chování v případě úspěchu, či neúspěchu. Jedná se většinou o bindování načtených dat, nebo zobrazení chybové hlášky.

```
// obsluzny handler v pripade uspechu cteni ze souboru
function loadResultHandler (result) {
   var e = document.getElementById('loginScope'); // vyhledani elementu v DOM
   if(e != null){ // pokud element existuje v DOM
       var loginScope = angular.element(e).scope(); // vyhledni scope elementu
        loginScope.saveLoadedToScope(result); // volani funkce z nalezeneho
scope
       }
}

// obsluzny handler v pripade neuspechu cteni ze souboru
function loadErrorHandler (error) {
  var e = document.getElementById('loginScope'); // vyhledani elementu v DOM
  if(e != null){ // pokud element existuje v DOM
       var loginScope = angular.element(e).scope(); // vyhledni scope elementu
      loginScope.open(); // volani funkce z nalezeneho scope
       }
}

// obsluzny handler v pripade uspechu zapisu do souboru
function saveResultHandler (result) {
   //alert("SAVE SUCCESS: \r\n"+result );
   var e = document.getElementById('loginScope'); // vyhledani elementu v DOM
   if(e != null){ // pokud element existuje v DOM
       var loginScope = angular.element(e).scope(); // vyhledni scope elementu
        loginScope.close();// volani funkce z nalezeneho scope
       }
}

// obsluzny handler v pripade neuspechu zapisu do souboru
function saveErrorHandler (error) {
   alert("SAVE ERROR: \r\n"+error ); // vypsani chybove hlasky
}

// obsluzny handler v pripade uspechu editace souboru
function editResultHandler (result) {
   var e = document.getElementById('loginScope'); // vyhledani elementu v DOM
   if(e != null){ // pokud element existuje v DOM
       var loginScope = angular.element(e).scope(); // vyhledni scope elementu
        loginScope.fillForm(result); // volani funkce z nalezeneho scope
       }
}
// obsluzny handler v pripade neuspechu editace souboru
function editErrorHandler (error) {
  alert("EDIT ERROR: \r\n"+error ); // vypsani chybove hlasky
  var e = document.getElementById('loginScope'); // vyhledani elementu v DOM
  if(e != null){
       var loginScope = angular.element(e).scope(); // vyhledni scope elementu
      loginScope.open(); // volani funkce z nalezeneho scope
       }
}
```

Propojení nativního kódu a javascriptu

Poslední a nezbytná část návodu slouží k zajištění, aby plugin v phonegapové aplikaci fungoval podle očekávání. Vytvoření nativního kódu a kódu mobilní aplikace v javascriptu samo sobě nestačí.

1) Nejprve je potřeba přidat mapování pluginu do konfiguračního souboru phonegapové aplikace (*config.xml*):

```
<plugins>
  <plugin name="UserSettingsPlugin" value="UserSettingsPlugin" />
</plugins>
```

2) Následně můžeme z javascriptových kontrolerů volat funkce přiřazující handlery funkcím nativního kódu. Pokud chceme volat funkce pluginu hned při startu phonegapové aplikace, je nutné to učinit v javascriptovém handleru „ondeviceready“, jinak dojde k pádu phonegapové aplikace kvůli nekonzistenci.

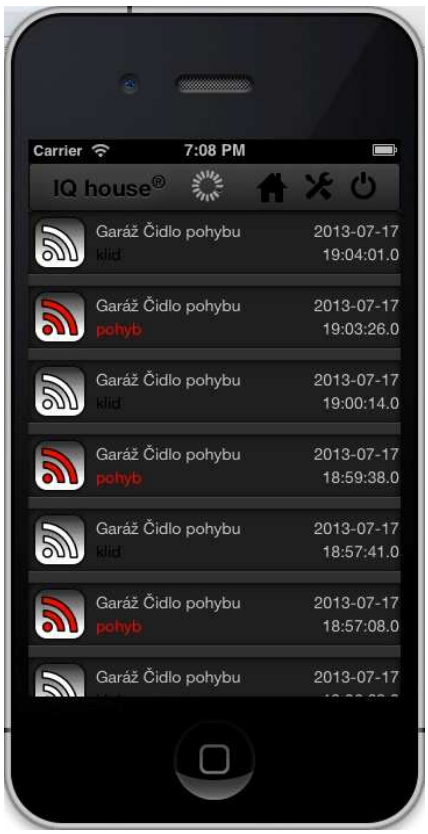
```
// v pripade ze je aplikace v mobilnim zarizeni pripravena k pouziti
document.addEventListener("deviceready", function() {
              // volani pluginu pro nacteni uzivatelskeho nastaveni
              callUserPluginLoad('firstLoad');
      }, false);
```

Příloha C: Screenshoty mobilní aplikace IQ House na zařízení iPhone


Carrier
7:01 PM
IQ house®
OVLÁDÁNÍ
Vývoj Světlo 1. a 2. od oken A
Zapnout
Vypnout
Intenzita:
0
Zavřít
Vývoj Světlo 2. od oken C


Carrier
7:01 PM
IQ house®


Carrier
7:02 PM
IQ house®
KAMERA
vzadu
Zavřít


Carrier
7:08 PM
IQ house®
Garáž Čidlo pohybu
klid
2013-07-17
19:04:01.0
Garáž Čidlo pohybu
pohyb
2013-07-17
19:03:26.0
Garáž Čidlo pohybu
klid
2013-07-17
19:00:14.0
Garáž Čidlo pohybu
pohyb
2013-07-17
18:59:38.0
Garáž Čidlo pohybu
klid
2013-07-17
18:57:41.0
Garáž Čidlo pohybu
pohyb
2013-07-17
18:57:08.0
Garáž Čidlo pohybu
2013-07-17