Univerzita Hradec Králové
Fakulta informatiky a managementu

Komfortní grafické uživatelské rozhraní pro řízení inteligentních domů

Diplomová práce

2017 Miroslav Bartoš

Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informačních technologií

Komfortní grafické uživatelské rozhraní pro řízení inteligentních domů
Diplomová práce

Autor: Miroslav Bartoš
Studijní obor: Aplikovaná informatika

Vedoucí práce: Ing. Jan Matyska

Hradec Králové duben 2017

Prohlášení:

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne: Miroslav Bartoš

Poděkování:

Rád bych poděkoval své rodině, akademickým pracovníkům i přátelům. Konkrétně děkuji svým rodičům za bezbřehou podporu během celého studia, své přítelkyni také za podporu a motivaci, V. Sobotkovi a P. Pscheidlovi za odborné debaty a posouvání vpřed, kolegům z firmy Apso s.r.o. za přátelské prostředí a tím i prostor ke studiu a Janu Matyskovi za již druhé vedení směru závěrečné práce. Ti všichni mají podíl na této práci.

Anotace

Cílem diplomové práce je výběr postupů, technologií a nástrojů pro nastolení standardu vývoje specificky zaměřených webových aplikací. Na jeho základě je vypracován koncept a ukázková implementace komfortního grafického uživatelského rozhraní pro ovládání inteligentních domů. Tato práce se zabývá návrhem optimálního uživatelského rozhraní pro intuitivní navádění k potřebnému vybavení domů, analyzuje principy responzivního designu a vyvozuje efektivní nástroj pro implementaci. Na základě nejčastějších chyb konkurenčních řešení je část teorie věnována grafickému designu včetně analýzy barev a písma.

Výsledné uživatelské rozhraní splňuje nároky moderní webové aplikace. Vybrané technologie přesouvají část výpočtů na klientskou aplikaci a při změně dat aktualizují minimální část obsahu. Na základě zmapování trhu s inteligentní elektroinstalací aplikace umožňuje ovládání určitého počtu akčních prvků. Hotová aplikace je schopna rychlého přizpůsobení a poté nasazení na infrastrukturu systému pro ovládání inteligentních domů. To dává uživateli možnost spínat, regulovat nebo číst informace z elektricky napájeného vybavení domů a bytů.

Annotation

Title: Comfortable graphical user interface for intelligent home control

The aim of the diploma thesis is the selection of procedures, technologies and tools for the implementation of the standard of development of specific web applications. Based on this, a concept and demonstration of a comfortable graphical user interface for intelligent home control is developed. This thesis deals with the design of the optimal user interface for intuitive orientation to the necessary home equipment, analyzes the principles of the responsive design and derives an effective implementation tool. Based on the most common mistakes of competing solutions, part of the theory of graphic design, including color and font analysis.

The resulting user interface meets the demands of a modern web application. Selected technologies move part of the calculations to the client application and update the minimum content when they change the data. Based on smart market mapping, they allow you to control several action elements. A ready-made application is capable of quickly adapting and then deploying intelligent home management infrastructure.

Obsah

Úvod

Vývoj veškerých typů budov směřuje ke stále silnější integraci různých systémů. Instalace zabezpečovací, osvětlovací a další techniky do nových nebo stávajících staveb je již několik let perspektivní záležitostí. Tímto postupem se zrodil obor zvaný inteligentní elektroinstalace, kterému se věnuje první kapitola. Na rozvoj tohoto stále nového oboru poukazují čísla prodejů, úroveň instalace i nové strategie podniků, které se doposud zabývaly pouze jednoduchým vybavením domů bez známek komplexnějšího propojení.

Tím vzniká poptávka po pohodlném ovládání. To musí být schopné obsluhovat veškeré vybavení a příkazy všech členů rodiny. Se vzrůstající popularitou mobilních zařízení vzniká požadavek na takové řešení, které bude dostupné nejen z desktopových počítačů, ale i na všech mobilních platformách. To samotné dnes již k úspěchu nestačí, a proto se analýza věnuje také UX designu. Moderní aplikace musí komunikovat s uživatelem prostřednictvím barev, rozdělovat obsah pomocí použitých fontů a zanechávat v uživateli pozitivní emoce.

V dnešním stále zrychlujícím tempu je však nutné zvolit takové postupy a nástroje, které umožní efektivní a rychlý vývoj. Klíčový je návrh znovupoužitelných komponent a použití hotových řešení, které usnadňují práci designéra a vývojáře. Právě díky těmto vlastnostem se dostávají do popředí nové technologie popsané v aplikačních rámcích, a především v analýze technologií pro webové aplikace. Jejich častým jmenovatelem je jednoduchý inovativní princip, výběr z hotových prvků namísto zdlouhavého vytváření vlastních a široká komunita věnující se opravě chyb a posouvání technologie vpřed.

Tento trend je možné vidět nejen u sestavování uživatelského rozhraní ale i dalších částí aplikace. Objektově orientované programování je založeno na vytváření celistvých entit s veřejnými metodami pro opakované výpočty i přenositelnost mezi projekty. Objekty pro přístup k databázi nabízejí abstrakci, díky které je práce s databází přehlednější a odstiňuje od sebe logickou vrstvu od datové. Kdykoliv je tedy možné změnit jednu část aplikace bez zásahu do jiné. A tato analogie pokračuje až k aplikačním serverům. Vybrané aplikační servery jsou opět složené z modulů a jejich rozšíření obnáší nalezení nového modulu, integraci do systému a nastavení. Správně vybraná technologie a nástroje dokáží zefektivnit práci vývojáře a snížit cenu výsledného řešení.

Cíle

Práce se zaměřuje na navržení a realizaci moderní aplikace se specializací na řízení inteligentních domů. Ta by měla být platformě nezávislá a neměla by být omezena velikostí zobrazovacího zařízení.

Analýzou UX designu bude poznamenán grafický návrh aplikace s ohledem na specifický účel uživatelského rozhraní. Tento výběr bude podpořen technologií, která umožňuje nejpohodlnější užívání aplikace a zároveň vyhovuje kritériím znovupoužitelnosti, aktivní komunitě a strmé učící křivce.

Tato práce by měla nabízet inovativní princip řízení přístupu uživatelů k akčním prvkům i na samotné obrazovky uživatelského rozhraní. Řízení přístupu musí korespondovat se zabezpečením samotné aplikace a požadavkem na automatizované sestavení výsledného rozhraní s minimálními úpravami při budoucím rozšiřování.

Na základě vyjmenovaných požadavků bude sestaven koncept, který se stane předobrazem výsledného řešení. Tím by měla být aplikace schopná rychlého přizpůsobení a nasazení na hotovou infrastrukturu pro ovládání inteligentních domů. Mezi její přednosti by mělo patřit snadné přizpůsobení pro konkrétního zákazníka. Výhodou by bylo případné rozšíření o definování uživatelských pravidel nebo instalace pluginů.

1 Smart Home

Na základě specifického zaměření této práce bude analyzován vývoj různých systémů, které se integrují do budov a bytů. Jelikož se vybavení zabezpečovací, osvětlovací a další technikou stává trendem, zrodilo se odvětví zvané inteligentní elektroinstalace, resp. inteligentní dům. Podle publikace Inteligentní dům [1] od Mgr. Valeše vystihuje inteligentní elektroinstalace vybavení počítačovou a komunikační technikou s cílem zvýšit komfort, pohodlí, snížit spotřebu energií, poskytnout zábavu a bezpečí pomocí řízení všech technologií v domě a jejich interakcí s vnějším světem. Díky tomu je stavba schopná předvídat a reagovat na potřeby obyvatel.

1.1 Vývoj elektroinstalace

Myšlenka automatizovaného bydlení vznikla již v 50. letech. Počátkem veškerého technického pokroku byla mechanizace. Ta přinesla stroje a zařízení a převzala část práce člověka. Druhým milníkem byla automatizace, která zjednodušila řízení strojů pomocí výpočetního vybavení.

S rostoucími nároky na pohodlí, bezpečnost a hospodárnost ve stavebnictví narůstá podíl automatizace jak v bytové výstavbě, tak v účelových stavbách. Automatizace budov se podle knihy Automatizované systémy budov [2] vyvinula v samostatný obor automatizační techniky, který poskytuje zákaznicky orientovaná řešení jak provozovatelům, tak uživatelům všech druhů budov. Automatizační prvky, jakými jsou snímače, akční členy, regulátory a rovněž vizualizace procesů, zpravidla fungují decentralizovaně.

Aby tyto přístroje mohly provádět složité úkoly, musí být vybaveny komunikačním systémem pro vzájemnou výměnu dat. K tomuto účelu se používají provozní sběrnice a sítě. Obor inteligentní dům využívá dosavadních řešení. Kombinací těchto řešení poskytují stavby nový pohled, a především nové funkcionality. Samozřejmostí již bývá promyšlená příprava nových domů pro budoucí instalaci, aby bylo možné případné změny provádět rychle, jednoduše a bez stavebních prací.

1.2 Rozdělení dle míry inteligence

Inteligentní dům přináší nový způsob komunikace člověka s domem. Doposud bylo vše řízeno obyvateli domu. Technika inteligentního domu je však založena na opačném přístupu. Publikace Inteligentní dům [1] přináší rozdělení do pěti stupňů, pro lepší porozumění možným mírám „inteligence“ domu.

1.2.1 Obsahující inteligentní zařízení a systémy

Tento typ domu obsahuje pouze samostatná zařízení a systémy pracující nezávisle na ostatních. Příkladem je snímač pohybu a snímač úrovně osvětlení. Tento jednoduchý systém reaguje na vstup člověka do místnosti pouze v případě, že není dostatek venkovního osvětlení.

1.2.2 Obsahující inteligentní komunikující zařízení a systémy

Druhý typ využívá opět inteligentně fungující zařízení a systémy, které si z důvodu zdokonalení činnosti vyměňují informace mezi sebou. Typické využití je možné nalézt v propojení zámku vchodových dveří a bezpečnostního vybavení. Po zamčení dveří se automaticky zapne alarm a další prvky bezpečnosti a dojde ke zhasnutí světel, stažení rolet v přízemí, vypnutí některých spotřebičů a snížení nastavené teploty.

1.2.3 Propojený dům (tzv. connected home)

Základem této míry inteligence je vnitřní a vnější komunikační síť. Největším přínosem je možnost interaktivního vzdáleného ovládání systémů. Vzniká tím přístup k informacím o domě a jeho službám. Opět se nabízí použití s bezpečnostním systémem, který v případě vniknutí do domu rozsvítí světla, zavolá bezpečnostní službu a umožní vzdálený přístup k bezpečnostním kamerám.

1.2.4 Učící se dům

Učící se dům zaznamenává aktivity aktuálně probíhající v domě a nashromážděné údaje používá k předvídání potřeb jeho obyvatel. Vyhřívání je první týden monitorováno a systém vyhřívání si vytvoří vlastní plán spouštění a nastavení topení. Poté již není nutné zasahovat do ovládání.

1.2.5 Pozorný dům

Charakteristickou vlastností tohoto typu domu je neustálé monitorování polohy lidí a předmětů. Dům je poté schopný přesně a okamžitě reagovat na potřeby obyvatel. Technologie jsou samočinně ovládány. Na rozdíl od učícího se domu zde vše probíhá v reálném čase. Příkladem mohou být speciální podlahy pro snímání polohy obyvatel.

1.2.6 Shrnutí

Mezi jednotlivými typy inteligence je možné nalézt jistou souvislost. Každá další míra inteligence zahrnuje tu předchozí a přidává několik vlastních benefitů. Zatímco „Pozorný dům“ je stále ještě konceptem, zbývající stupně jsou běžně dostupné. V tuto chvíli je nejpokročilejším stupněm „Učící se dům“.

1.3 Rozdělení dle prostředí

Druhým a stejně důležitým hlediskem při volbě systému pro inteligentní ovládání domu je prostředí, do kterého budou komponenty instalovány. To, kam budou komponenty zasazeny, silně souvisí s účelem automatizace budovy. Tímto členěním se zabývá odborná literatura Automatizované systémy budov [2].

1.3.1 Automatizace budov v soukromé bytové výstavbě

Podle tendencí v soukromé bytové výstavbě se několik automatizovaných funkcí stává určitým standardem. Za samozřejmost bývá považována regulace spotřeby energie, kdy jsou optimalizované regulační funkce integrovány do systému vytápění. Do komponent regulace teploty výrobci integrují programy pro časování a sepnutí režimu noční spotřeby. Tuto funkci lze shrnout jako energetickou úspornost.

Dalším frekventovaně používaným příkladem automatizace je řízení osvětlení. Osvětlení obytných domů se mnohdy samočinně spíná instalovaným hlásičem vstupu. Zachycení pohyblivého objektu provádí jeden snímač a v kombinaci se signály snímačů intenzity osvitu se osvětlení sepne jen v tom případě, že je již dostatečné šero. Jedná se o primitivní případ jednoduché automatizační funkce. Řešení představuje spojení řízení události s logickou operací. Zde je aspektem pohodlí.

Daleko komplikovanějším se stává osvětlení celého obytného domu. Především zapínání a vypínání osvětlení z jednoho místa. Konvenční řešení by znamenalo zavedení rozsáhlých kabelových rozvodů. Doslova revoluční postup zde nabízí použití sběrnicových systémů a s nimi související komunikace mezi všemi komponentami. Poté lze pohodlně ovládat spínačem např. i alarm v případě vloupání. Mimo jiné jde v tomto směru o zajištění bezpečnosti a centralizaci ovládání.

Podle výše zmíněného popisu získaly v soukromé výstavbě zájem především hospodárnost a úspora energií, komfort a bezpečnost.

1.3.2 Automatizace budov v účelových stavbách

Mezi účelové stavby se řadí například kancelářské budovy, nákupní střediska, nemocnice atd. Obsahují přístroje na výrobu tepla, instalace chladicích přístrojů a vzduchotechniky. Tyto, většinou podnikatelsky provozované objekty, se vybavují zpravidla náročnými řídicími a regulačními systémy. Ty zajišťují ve spojení s dispečerským řízením bezvadný provoz. Výše zmíněné je možné popsat jako optimalizaci spotřeby energií a úspory ve snížení počtu pracovníků obsluhy.

Průzkumy výkonnosti zaměstnanců jasně ukazují, že se zhoršováním pracovního prostředí klesá i výkonnost pracovníků. Z tohoto důvodu jsou již standardně účelové stavby vybaveny vzduchotechnikou. Tato opatření přispívají k růstu komfortu.

Častým a pro automatizaci náročným požadavkem bylo pravidelné přeuspořádání vnitřních prostor těchto budov. Může nastat situace, kdy je např. z těsných kanceláří urychleně nutné vytvořit konferenční místnosti. Rychlá a bezproblémová restrukturalizace firmy je možná díky přeprogramování inteligentních komponent namísto přetahování celé kabeláže. Tato jednoduchá operace zvyšuje flexibilitu automatizace budov.

Na první místo je posuzována opět hospodárnost a úspora. Oproti automatizaci v soukromé výstavbě je zde zásadní komunikace pomocí sběrnicových sítí a zajištění flexibility díky inteligentním komponentám.

2 Design webové aplikace

Úkolem designéra je řešit rozpor mezi záměry provozovatele aplikace a potřebami jejích uživatelů. Hledat styčný bod mezi zájmy a cíli obou stran a vybudovat k němu pro oba příjemnou a pohodlnou cestu. Před postupem návrhu webové aplikace, o kterém tato práce pojednává, je nutné popsat úzce související pojem. Tím je UX design.

2.1 UX design

UX je zkratkou pojmu *User eXperience*. UX má široký význam zahrnující v sobě zkušenosti, zážitky, emoce, postoje, přístup, styl, používání, proces učení, vzorce chování a jejich změny a vývoj. Jednoduše jde o to, co uživatel nějakého produktu dělá, co přitom prožívá, jaké v něm produkt zanechává pocity a také co si jeho použitím odnáší. Tento složitý komplex nelze výstižně přeložit, a proto je vhodné zůstat u anglického „user experience“. UX designu se obvykle věnuje designér. Snaží se navrhnout a vytvořit produkt s příjemným a užitečným používáním, budícím a zanechávajícím pozitivní emoce. Tato práce je zaměřena především na design webových a mobilních aplikací. Zmíněné postupy a zásady jsou však použitelné i v jiných oborech. Pro potřeby této práce bude využíván UX design interaktivních aplikací.

Slovo aplikace se ustálilo ve významu software (softwarová aplikace). Produkty označené pojmem interaktivní aplikace lze rozdělit do několika kategorií:

1. Webové stránky – pomocí odkazů vzájemně propojené dokumenty
2. Webová aplikace – dynamické webové stránky s interaktivními službami
3. Počítačové aplikace – aplikace spustitelné v operačních systémech (včetně jejich GUI)
4. Mobilní aplikace – rozhraní programů cílených na smartphone, tablety a další přenosná zařízení
5. Nepočítačové aplikace
6. Rozhraní přístrojů a automatů
7. Další interaktivní uživatelská rozhraní, tzn. jakýkoli přístroj či nástroj, který s uživatelem komunikuje (k akcím a příkazům uživatele poskytuje odezvu a očekává reakce)

Obecné postupy jsou použitelné pro většinu z popisovaných grafických, uživatelských rozhraní. Tato diplomová práce je cílena na první čtyři skupiny aplikací z výše vyjmenovaných a detailněji rozebraných v knize Dobrý designér to všechno ví! [3]. Dále se kniha věnuje rozdílu mezi pojmy interaktivní a interakční v kontextu designu.

Design ve standardním pojetí je pouze jednosměrnou komunikací. Úkolem designéra je navrhnout způsob, jakým bude čtenáři sdělena informace. Příkladem může být obraz nebo plakát a mezi druhy komunikace jej lze zařadit jako monolog. Oproti tomu při interaktivním způsobu komunikace uživatelé pouze nekonzumují obsah sdělované informace, nýbrž poskytují zpětnou vazbu a svým chováním ovlivňují další chod aplikace. Tím byl popsán význam slova interaktivní, a právě zde přichází na řadu tzv. interakční design. Interakční designér poté mimo stylu a podoby informace definuje i dorozumívací jazyk mezi uživatelem a rozhraním počítače. Dále formuluje styl, jakým se bude dialog vést, navrhuje snadno srozumitelné a co nejméně matoucí příkazy k nejčastějším cílům uživatele.

Zobrazení kroků, slov a vět dialogu odpovídají běžnému zadání klasického designu. Stále je nutné navrhnout konkrétní datovou tabulku, navrhnout chybové hlášení, design úvodní prezentace firmy i tlačítka přihlašovacího formuláře. Všechny tyto dílčí prvky jsou však zastřešeny právě interakčním designem do jednoho sourodého celku. Během postupu je kladen důraz na konzistenci, logiku, intuitivnost, srozumitelnost a samozřejmě na jednoduchou a přímočarou cestu k cílům. Zjednodušeně je prioritou interakčního designu sjednocení všech dílčích částí interaktivní aplikace do jediného sourodého, logického a intuitivního systému.

Nejen při větším počtu designérů je potřeba organizovat postup práce s cílem maximalizovat kvalitu produktu a rychlost vývoje. Tyto procesy se dají začlenit do dvou známých kategorií. Podobně jako při běžném vývoji aplikací se jedná o tzv. vodopád nebo agilní přístup. Za účelem vytváření designu, ve kterém se budou orientovat vývojové týmy, je nápomocné, aby UX designéři rozuměli vývojovým procesům. Jedním z klíčových důvodů, proč věnovat čas a text navíc je tým obsahující externí pracovníky navrhující design. Ti musí být schopni plánovat a koordinovat práci souběžně se softwarovými vývojáři. Tuto problematiku popisují Jesmond Allen a James Chudley ve své publikaci Smashing UX design [4].

Při organizačním postupu nazvaném vodopád přechází činnost z jedné fáze k další. Od sestavení požadavků zákazníka, přes návrh, implementaci až po validaci. Důsledkem je absence zpětné vazby. Design je zde umístěn před dalším postupem. Navíc je kladen důraz na zdokumentování a ukončení prací předtím, než vůbec započne další fáze. Postup odpovídá činnostem UX uskutečňovaným ještě před započetím samotného vývoje. Výstup práce UX designéra, kterým může být mimo jiné softwarový drátěný model, může ovlivnit část specifikace produktu.

Agilní přístup je novější metodou, která reaguje na nedostatky vodopádu. Mezi tyto nedostatky patří špatně shromážděné informace nebo změna požadavků projektu. Obojí má za následek zdržení vývoje. Pokud neexistuje vedení UX, jsou změny zadání téměř jisté. Prosazovatelé agilního přístupu vývoje argumentují tím, že čas potřebný pro návrh by měl být vždy obětován v zájmu vývoje provozuschopného produktu, který bude testován, sestaven a případně změněn tak, jak je nezbytné.

V této podkapitole byl popsán rozdíl mezi klasickým designem a UX designem. Jsou jimi jednoduché kroky a zásady, kterých se designér drží a více či méně se všechny snaží poskytnout uživateli maximální komfort a snadnou obsluhu. Většina publikací však opomíná jeden faktor, který stojí nad vším dosud popsaným. Tím faktorem je klient. Osoba s iniciativou zřídit daný projekt, spustit jej a především financovat.

2.2 Grafický design

Použité technologie mají nepřímý vliv i na samotného uživatele. Rychlost vykreslování, odezva na akce atd. jsou měřítka, která vytváří kladný dojem a dlouhodobě udržují uživatele právě u onoho uživatelského rozhraní. Nicméně člověk si tvoří dojem již z prvního pohledu. Bez jakékoliv vzájemné interakce musí uživatelské prostředí působit na uživatele přívětivě. Výzkum ideálního grafického pojetí uživatelského rozhraní lze rozdělit do tří bodů. Právě těm náleží obsah následujících podkapitol.

2.2.1 Barvy

Barvy fascinují vědce po celá staletí. Gage v roce 1993 [5] popsal první teorie barev od významných filosofů Democritus a Aristoteles). Autor Goethe [6] však již v roce 1810 popsal teorii, podle které barvy vytvářejí emocionální reakce (teplo, vzrušení). Kurt Goldstein [7] roku 1942 rozšířil Goetheho intuice a provedl experimentální pozorování ohledně působení barev na funkci organismu. Zjistil, že některé barvy produkují systematické fyziologické reakce projevující se jako emocionální prožitek (tím může být například nepříjemné vzrušení), změnu při orientaci v prostoru (člověk se potřebuje z uzavřených prostor pohybovat směrem ven), ale i jasně patrné reakce (energické chování).

Na základě této práce bylo zjištěno, že barvy s delší vlnovou délkou působí teple a pocity rozrušení. Naopak barvy s kratší vlnovou délkou vyvolávají pocit chladu nebo relaxace. Nakashi [8] popsal problém vztahu mezi barvou a chováním jako extrémně bohatý a komplexní. Z jeho článku nazvaného The Effects of Red and Green Surroundings on Behavior vyšel závěr, že červená a v menší míře i ostatní teplé barvy, oranžová a žlutá, mají budící až rozrušující vliv na chování, positivní efekt na rychlé motorické reakce, působí jako povzbuzující pohon a vyvolává světlou náladu. Mezi negativními účinky lze zmínit zhoršení vykonávání činností, které vyžadují rozhodnost, přesnost, koordinaci nebo jemnou motoriku. Kontrastní barvy modrá nebo zelená mají uklidňující účinek.

Článek Color and psychological functioning: a review of theoretical and empirical work [9] od Andrew J. Elliota si dal za cíl sjednotit dosud zjištěné závěry teoretického i empirického výzkumu zaměřeného na vztah barev a lidské chování. V oblasti výzkumu barev a pozornosti se ukázala jasná výhoda červené barvy. Výzkum barev a bdělosti odhalil, že modré světlo zvyšuje bdělost a výkonnost u úkolů zaměřených na pozornost. Duševní výkonnost je narušována zobrazením červené barvy. V oblasti obchodu bylo zjištěno, že modrá barva (např. barva loga) zvyšuje dojem kvality a důvěryhodnosti.

Volba barev tvoří důležitou část při tvorbě uživatelského rozhraní. Aplikace pro řízení inteligentního vybavení je používána v pohodlí domova a svým vzhledem dokáže ovlivnit chování jejích uživatelů. Pro tento specifický účel je na základě rešerše doporučena modrá nebo zelená barva. Ta by měla zabírat větší plochu uživatelského rozhraní a podprahově působit uklidňujícím dojmem. Zažité barvy pro rozhodování zelená a červená by měly podtrhnout tlačítka pro výběr možností. V jakémkoliv jiném, než varovném kontextu jsou barvy oranžová a červená pro tento účel nežádoucí. Vzhledem k tomu, že ovládací prvky tvoří část vybavení domu, měly by zůstat především konzervativní a splynout s jakýmkoliv okolím. Zbylá kombinace barev musí dostatečně zvýraznit text a přitom zůstat neutrální.

2.2.2 Písmo, typografie

Písmo je jedním z nejdůležitějších vynálezů lidstva. Umožnilo neopakovat chyby předků a posouvat se tak vpřed. Existují různé druhy písma pro různé účely. Pro správný výběr je nutné je nejprve umět rozlišit a klasifikovat. Z důvodu častého zaměňování budou však nejprve přiblíženy vybrané pojmy. Autorka Kristin Cullen ve své knize Design Elements, Typography Fundamentals [10] popsala, jaký má typografie dopad na celkový design. Typografii označuje jako technický obor zabývající se písmem. V tomto oboru se pod pojmem font označuje kompletní sada znaků abecedy jedné velikosti a jednotného stylu. Řez písma (angl. typeface) opticky rozlišuje významem stejné znaky. Rozlišujeme základní řez, kurzívu a tučné písmo. Další řezy jako polotučné písmo rozšiřují nabídku v tučnosti nebo šířce znaků. A nakonec rodina písma (angl. font family) sdružuje řezy písma. Značí množinu jednoho nebo více fontů, které sdílejí stejný zřetelný design.

Podle publikace The fundamentals of typography [11] se v typografii rozlišují absolutní a relativní měřítka. Absolutní měřítka jsou fixní, a proto jsou nám bližší a jednoduchá na pochopení. Mezi jejich jednotky patří například milimetr, bod nebo palec. V typografii je mnoho měr, kupříkladu řádkování znaku, spojeno s velikostí typu fontu. To znamená, že jejich hodnota je definována relativně vzhledem k jiné míře. Jednotky *em* a *en* nemají žádnou předepsanou absolutní velikost. Jejich velikost je relativní k nastavené velikosti typu písma.

Rozestup mezi jednotlivými řádky (angl. leading) je dalším příkladem použití relativních jednotek. Mnoho aplikací pro leading automaticky přiřazuje procentuální hodnotu. Pokud má písmo velikosti 10 bodů a leading nastavený na 120 %, skutečný rozestup je poté 12 bodů. Pokud by se tak nestalo a leading by zůstalo konstantní, jednotlivé znaky by při zvětšování začaly zasahovat do řádku nad nimi. Na podobném principu je postaven responzivní design, který upravuje šířku a rozložení objektů na stránce relativně k použitému zařízení (šířce obrazovky tohoto zařízení). Tolik k použitým jednotkám. Mezi vybrané faktory ovlivňující dobrou čitelnost a tím použitelnost písma patří také střední výška písma (angl. x-height). Ta udává velikost malého x ve vybraném řezu písma. Písma s větší střední výškou písma jsou čitelnější. Spacing, leading a x-height napomáhají při výběru správného fontu.

Kniha Responzivní webdesign [12] se věnuje již praktické volbě stylů. Kapitola nazvaná *Fluidní typografie* podporuje předchozí tvrzení a přibližuje aspekty typografie v relativních jednotkách. Díky jim mohou typografické prvky reagovat na změny zařízení, která aplikaci zobrazují. Vyzdvihuje zajímavý jev, kterým je shodná velikost fontu (16 px) napříč prohlížeči. Nastavení font-size na 100 % tak bude mít v prohlížečích jednotný výsledek. Toto fixní řešení však podle knihy zrazuje uživatele a ten tím přichází o výhody fluidní typografie.

Vzhledem k interpretaci výchozích velikostí písma nadpisů, bloků, sekcí a tagů článek u prohlížečů a snadným násobkům je doporučována velikost 62,5 %. Ta odpovídá ideálním 10 pixelům. Právě zde přicházejí na řadu relativní jednotky. Hodnota 1.2 em odpovídá velikosti 12 px pro běžný obsah a nastavených 1.6 em zvětší text na 16 pixelů. Tím je popsán základní přístup ke stylování textu, který zabrání zhušťování a překrývání znaků. Podobně je potřeba přistupovat k samotnému rozvržení obsahu, čemuž se věnuje kapitola Responzivní design.

Standardně spadají písma do jedné ze dvou kategorií, a to serif nebo sans serif. Písma označovaná jako serif mají na konci příčné linky. Díky těmto sotva znatelným linkám je písmo lépe čitelné a rozpoznatelné. Písmo serif pomáhá k lepší orientaci na stránce. Naopak písma sans serif jsou jednodušší, za to však hůře čitelná. Jednoduchá písma sans serif jsou označována jako moderní, kdežto písma serif jako tradiční. Při sestavování grafického uživatelského rozhraní lze pro jejich estetické kvality a čitelnost využít různé typografické elementy.

Před rozvržením dispozic musí být zváženo, jakým způsobem bude uživatel získávat informace. Uživatel potřebuje vstupní bod designu. Myslí se tím jedna část z celého obsahu, která bude zvýrazněna a (nejen) uživatele, který vidí obrazovku poprvé, dokáže navést k tomu podstatnému a zároveň mu pomůže zorientovat se ve zdánlivě chaotickém a komplexním množství informací. K nechtěnému opačnému efektu (ztráta orientace uživatele) může dojít při použití většího množství různých fontů. Maximální doporučené množství fontů v jednom dokumentu jsou tři fonty.

Pro potřeby této práce bylo rozhodnuto o výběru dvou fontů. Jeden pro zvýraznění popisků formulářů, konkrétní části úvodní obrazovky, která má za úkol informovat o podstatných hodnotách získaných ze senzorů, případně vybraného nastavení. Ten by měl být rozhodně patkový pro jeho lepší orientaci v textu oproti bezpatkovým písmům. Druhý font by neměl být tolik kontrastní jako první. Úkolem tohoto fontu je zobrazit zbylý text, a především zůstat výborně čitelný i na menších zařízeních. V důsledku toho budou vybírány pouze fonty s dostatečnou hodnotou x-height. Oba fonty musí podporovat znaky latinské abecedy a musí být dostupné zdarma.

Pro tento případ byla vybrána knihovna fontů Google Web Fonts [13], která nabízí velké množství fontů s vyhledáváním podle vlastních kritérií, výčtem vlastností, které byly popsány v této práci a editorem pro okamžité otestování použitelnosti v prohlížeči. Nápomocné je i třídění výsledů podle popularity, které pomohlo k jednoduššímu nalezení kvalitního fontu splňujícího všechny požadavky.

Jako zástupce patkového písma byl vybrán font Gentium [14]. Podle Google Web Fonts je to druhý nejpoužívanější font mezi patkovými písmy (pokud vynecháme monospace a ručně psané fonty) s výskytem na 2,9 miliónech webových stránek. Vyznačuje se několika použitelnými tloušťkami, běžným stylem i kurzívou. V lednu 2016 byl upraven pro zlepšení renderování.

Roboto [15] byl vybrán z bezpatkových fontů. Splňuje požadovaná kritéria a může se pochlubit 9,8 milionu použití převážně pro operační systém Android. Je charakterizován mechanickou kostrou a geometrickými tvary, díky čemuž je dobře kombinovatelný s většinou patkových fontů.

2.2.3 Grafické komponenty

Během zpracování kapitoly 3 Konkurenční řešení bylo zjištěno několik znepokojujících podobností. Většina dostupných grafických uživatelských rozhraní je stále založena na jednoduchém seznamu. Jde o nejjednodušší výčet možností řízení. Uživatel se posouvá směrem dolů, přičemž u některých produktů má možnost filtrovat výsledky pomocí druhého seznamu, který vytváří logické skupiny podle místností nebo účelu prvku. Důvodem pro tento způsob zobrazení je snadná implementace. Největší problém tohoto přístupu je zobrazení na všech zařízeních, a to nikoliv kvůli čitelnosti nebo zhoršenému ovládání na malých/velkých zařízeních ale kvůli využití prostoru. Jde o zobrazení, které v uživateli nevytváří dobrý dojem. Kvůli popsanému problému vznikla dodatečně tato podkapitola, která má za cíl najít řešení opět responzivní a jednoduché ale zároveň lépe hospodařící s prostorem.

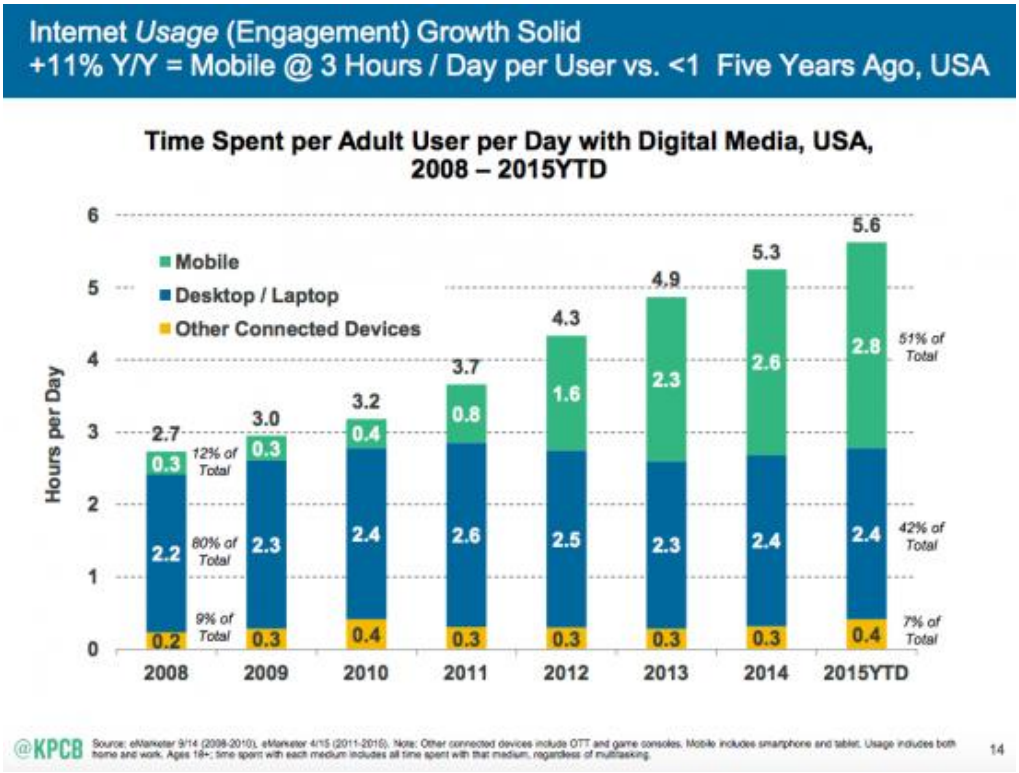
Článek 6 Reasons for Employing Component-based UI Development [16] se zabývá aplikací celistvých grafických prvků při vývoji uživatelských rozhraní. Komponenty jsou atomické jednotky a vytváření rozhraní z komponent umožňuje jejich budoucí znovupoužití. Primárním důvodem k jejich výběru je právě znovupoužitelnost. Vzhledem k tomu, že technologie přicházejí i odcházejí, je znovupoužitelnost alespoň části systému neocenitelná. Na tuto problematiku navazují autoři článku A component-based user interface approach for Smart TV [17]. V něm tvrdí, že přístup k návrhu designu pomocí stejných komponent urychluje vývoj. Podporuje totiž iterativní přístup vývoje. Během procesu návrhu komponenty, místo vytváření zcela nové, se designér zaměřuje na využití dosud investovaného času a pouze rozšiřuje hotové komponenty.

Problémem při návrhu designu může být udržet jednotný vzhled a chování napříč celou aplikací nebo dokonce celým portfoliem. Použití atomických, již nastylovaných, samostatných komponent zajišťuje UX konzistenci při rozšiřování produktu. K významným výhodám komponentového přístupu se dále přidává rychlejší přesun od designu k vývoji. Autoři obhajují komponenty jako cestu, jak se vypořádat s neustálou potřebou zrychlit vývoj nového systému nebo provedení změn. Stále zůstává potřeba průzkumu chování uživatele a kvalitní sběr požadavků od zákazníka. Znatelné urychlení lze pozorovat při vytváření wireframů, visuálního designu, vývoji uživatelského rozhraní a tím k celkovému zrychlení práce na front-endu.

2.3 Responzivní design

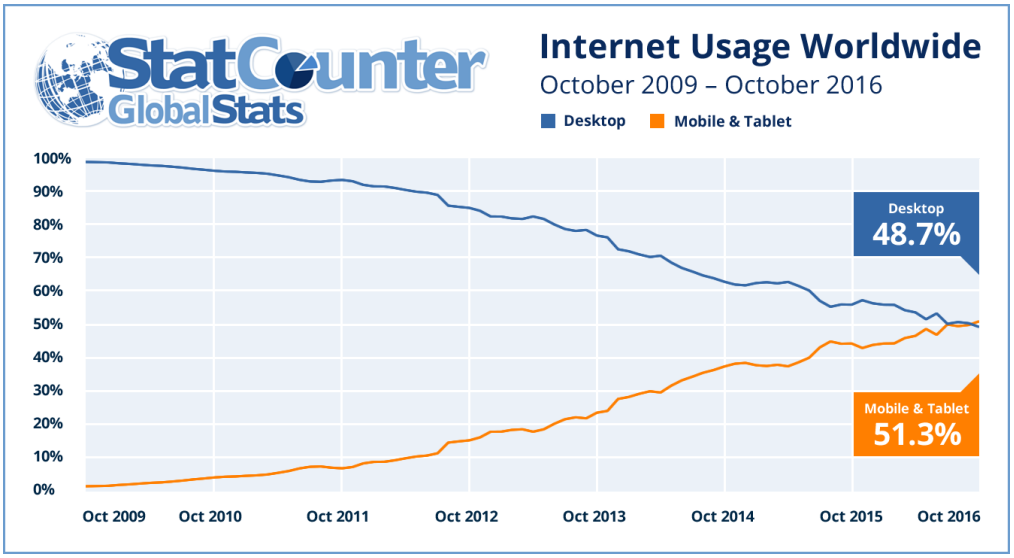
Pro co nejpřesnější analýzu je důležité poznat nejen samotného uživatele ale i jeho zařízení. Produkt cílený na desktop bude stěží konkurovat aplikacím odladěným pro mobilní platformy. Portál KBCP [18] se každý rok věnuje trendům v oblasti internetu, vyhodnocuje je a vytváří reporty. Právě jeden z nich ukazuje počet hodin strávených uživateli na jejich zařízeních od roku 2008 do 2015.

Z následujícího grafiky Obrázek 1 - Popularita mobilních zařízení je možné vyčíst, že v roce 2013 se stolním počítačům a notebookům vyrovnaly mobilní zařízení. Od naprosté nuly až k tomuto výsledku jim stačilo pouhých 6 let. V roce 2015 už přesvědčivě dominují, jelikož spotřebitelé tráví 51 % z času vymezeného pro média právě na smartphonech a tabletech.



Obrázek 1 - Popularita mobilních zařízení

Trend vzrůstající popularity mobilních zařízení potvrzuje zpracovaná statistika nezávislé analytické společnosti StatCounter [19]. Ta uvádí, že v říjnu 2016 měla desktopová zařízení podíl využití 48,7 %, zatímco mobilní zařízení dosáhly již 51,3 %. Jednotlivé země se v užití mobilních zařízení liší, avšak celosvětový trend z obrázku 2 je nevyvratitelný.



Obrázek 2 – Porovnání počtu uživatelů desktopových a mobilních zařízení

Předchozí grafy ukazují, že existuje hned několik zařízení, na které je potřeba se zaměřit. Publikace Moderní web [20] v kapitole Adaptivní vs. responzivní web design popisuje moderní metodiky webového vývoje. Obě zavádějí koncepci zlomových bodů. Rozdíl mezi adaptivní a responzivní metodikou spočívá v tom, jak se mění stránky mezi zlomovými body. Adaptivní web design používá sadu rozvržení s pevnou šířkou, kdežto responzivní web design spoléhá na flexibilní rozměry, takže stránky se přizpůsobují i mezi zlomovými body.

Responzivní design je sada technik, jejichž účelem je zpřístupnit aplikaci co nejvíce uživatelům na nejrůznějších zařízeních. Právě díky responzivnímu designu je možné se spolehnout, že se aplikace přizpůsobí zařízení a zájmům uživatele.

Se slovním spojením responzivní web design prorazil Ethan Marcotte. Jako průkopník tohoto oboru vydal v květnu 2010 článek a o rok později knihu s názvem Responsive Web Design [21]. Autor se zaměřil na fluidní mřížky, flexibilní obrázky a dotazy na zobrazovací médium. Tyto 3 oblasti jsou nosnými pilíři responzivního webdesignu. Mimo jiné, jak popisuje Ethan Marcotte ve své knize, je vhodné zaměřit se na změnu obsahu. Ta pilíře podporuje, a ještě více umocňuje funkčnost responzivity.

2.3.1 Fluidní mřížky

Prvním pilířem responzivního designu jsou fluidní mřížky. Jak název napovídá, jedná se o mřížku složenou ze sloupců a oddělovacích mezer. Šířky sloupců bývají relativní k šířce obalujícího elementu. Obalující element může mít pevnou nebo proměnlivou šířku. Právě přizpůsobivost vnitřních sloupců k obalujícímu elementu tvoří mřížku tzv. fluidní.

Fluidní mřížky vynikají v tom, že samy rozvrhnou obsah na stránce, což určí výsledný rozměr. Při správně práci je možné zaměřit se více na obsah a uživatelský prožitek než na strukturu aplikace. Výsledným produktem je aplikace, která se přizpůsobí jakémukoliv rozlišení. Není potom nutné vytvářet hned několik návrhu pro větší či menší zobrazovací zařízení. Další výhodou je, že designér nemusí řešit desetinná místa použitých hodnot a složitě testovat napříč prohlížeči.

V současné době existuje několik hotových řešení, která se jeví jako jeden z nutných nástrojů při vývoji webového designu. Jedno z nich bude použito i v této práci. Fluidní mřížkou to však nekončí. Fluidní může být i např. typografie. Ta se vyznačuje použitím relativních jednotek písma, mezer a odsazení. Tím je přesvědčivě zajištěno bezproblémové zobrazení.

2.3.2 Adaptivní obrázky

Druhý pilíř responzivního designu umožňuje poskytovat obrázky bez ohledu na omezení rozměrů obrazovky. Adaptivní obrázky reagují na různé rozměry a rozlišení obrazovek. Tento pilíř se snaží překonat hned dva problémy v mobilní sféře. Pokud má mít stránka fluidní strukturu, je potřeba, aby se strukturou byl kompatibilní i obsah. V opačném místě vznikne prázdné místo vedle fixního obsahu.

Již zmíněné fluidity lze snadno dosáhnout u textu. Přizpůsobení obrázků je však složitější. Druhým problémem je reakce na kvalitu obrazovek moderních zařízení. Především moderní AMOLED displeje zobrazují až 2x více pixelů, než je standard. Dvojnásobný počet pixelů má za následek dvojnásobné rozlišení a tím mechanismus pro detekci vyšší pixelové hustoty.

2.3.3 Dotaz na médium

Dotazy na médium tvoří třetí pilíř responsivního designu. Jedná se o rozšíření specifikace HTML5. Toto rozšíření určuje, jaké má prohlížeč aplikovat kaskádové styly. Již zmíněná koncepce zlomových bodů, které vznikají prostřednictvím dotazů na médium, mění rozvržení stránky. Prohlížeč je schopný rozpoznat mobilní zařízení včetně natočení, rozlišit ho od desktopového počítače a uplatnit vybrané kaskádové styly. Na základě zlomových bodů jsou postaveny aplikační rámce pro stylování webových stránek. Právě těmto rámcům se věnuje následující kapitola.

2.3.4 Aplikační rámce

Pro vývoj responsivních webových dokumentů existuje hned několik technik. Nejčastěji jsou však využívány tzv. boilerplate. Jedná se o předpřipravený znovupoužitelný kód, jenž může vývojář použít jako základ pro vlastní projekt. Dobrý boilerplate využívá podle publikace Responsivní webdesign okamžitě [12] zkušeností webové komunity, je optimalizovaný a testovaný. Tyto nástroje značně urychlují vývoj. Vzhledem k maximálnímu rozsahu těchto knihoven se zvyšuje jejich velikost a tím i velikost souborů, které musí prohlížeč při načítání obsahu stahovat. Je tedy žádoucí vygenerovat minimalizovanou knihovnu – pokud to daný nástroj podporuje nebo použít pluginy buildovacích nástrojů pro redukci velikosti kaskádových stylů. Podle zmíněných kritérií se zohledněním zdroje byly vybrány 2 hotová řešení.

2.3.4.1 HTML5 Boilerplate

HTML5 Boilerplate [23] je široce používaný boilerplate (tzn. často se opakující kód s minimálními nebo žádnými změnami) a podle serveru Github také aktivně využívaný. To dokazuje 37 tisíc hvězd. Tato šablona se zaměřuje výhradně na moderní prohlížeče. Pro uživatele starších prohlížečů dokonce zobrazuje upozornění na potřebnou aktualizaci. Pro mobilní boilerplate generuje H5BP tzv. mobile boilerplate. Ten se zaměřuje na mobilní webové stránky.

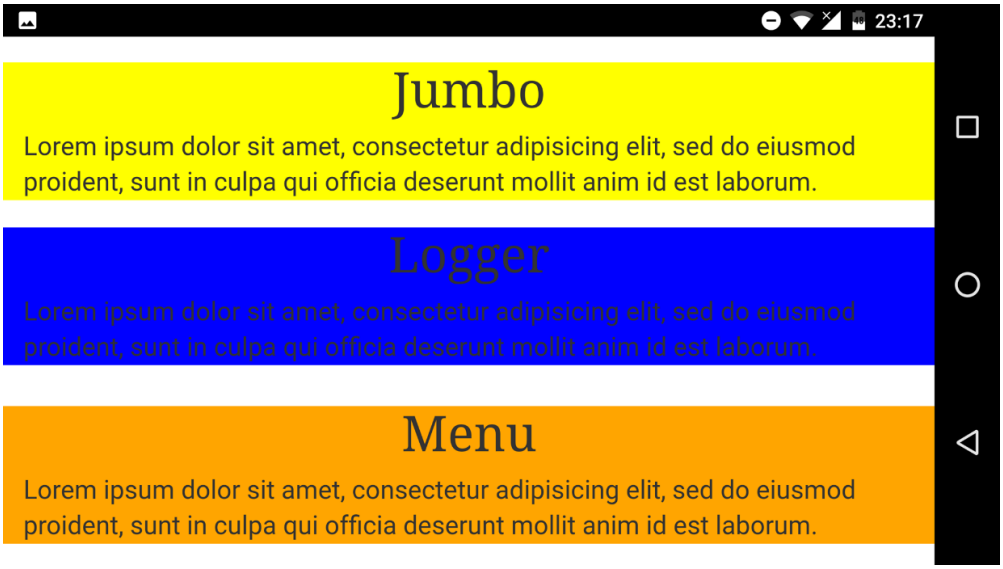
2.3.4.2 Twitter Bootstrap

Framework Bootstrap [24] nabízí mnohem více funkčnosti než boilerplate HTML5 Boilerplate. H5BP je základní boilerplate pro vývojáře, kteří vyžadují maximální míru kontroly. Bootstrap přináší k jeho výhodám nastavení velikosti písma, mřížkový systém a responzivitu. Na oficiálních stránkách je možné si ze standardní složení frameworku vybrat pouze to, co je opravdu nutné zahrnout.

Bootstrap daleko převyšuje konkurenci ve velikosti komunity. To dokazuje 109 tisíc hvězd na portálu Github. Bootstrap je určený pro vývoj responsivních mobile-first webových stránek. Obsahuje podporu pro čtyři typy zařízení podle šířky obrazovky. Podporuje dvanácti sloupcové rozložení webové stránky. Díky tomuto principu nabízí intuitivní práci a tím i strmou učící křivku. Limitující by mohlo být pokrytí všech dostupných zařízení pouze čtyřmi skupinami podle rozšíření. Pro testování použitelnosti byla navržena jednoduchá šablona. Na následujícím obrázku je zobrazena na desktopu s rozlišením 1920x1080.

Obrázek 3 - Testovací šablona Bootstrap

Problémová by mohla být také mobilní zařízení s vysokým rozlišením. Následující obrázek představuje tu samou šablonu zobrazenou na zařízení Huawei Nexus 6P s rozlišením 2560x1440 pixelů, kde rámec obstál na výbornou.



Obrázek 4 - Testovací šablona Bootstrap na mobilním zařízení

Pro výjimečné případy mnohopalcových displejů s nestandardním rozlišení poslouží rozšíření Bootstrap Big Grid, který široká rozlišení rozdělí na další podtřídy, které vývojář dokáže lépe uchopit.

2.3.5 Dynamický obsah

Dynamický obsah je schopný přizpůsobit se kontextu uživatele. Obsahem se myslí myšlenky, navigace, ovládací prvky, formuláře, slova i video. Pro menší zobrazovací zařízení je možné skrýt ovládací prvky, zobrazit pouze podstatný obsah apod.

2.3.6 Aplikace responzivního designu na hotový projekt

Jednoduchá změna aplikace je možná použitím elementu *meta* s názvem *viewport*. Tím lze přizpůsobit stránky uživatelům s menším nebo proměnlivým rozlišením. Týká se to především uživatelů mobilních telefonů nebo tabletů. Responzivní webdesign je návrhem několika dílčích logických reakcí. Vývojářská komunita se brání masivním změnám na trhu s moderními zařízeními. Komunita se snaží zavést postupy, které přetrvají do budoucna. Responzivita se stává standardem.

Záleží na vývojáři a aplikaci nebo webových stránkách, zda postačí úprava stávajícího produktu nebo bude potřeba začít do nuly. Na webové stránky s pevnou strukturou je možné použít fluidní mřížku. Stálý obsah a dostatečně flexibilní redakční systém by vyžadoval pouze vložení dotazů na médium. V robustním kódu by bylo cestou vkládání dynamických obrázků. Aplikování responzivního designu nevyžaduje splnění všech pilířů. Velikost současné aplikace není překážkou pro zavedení responzivního designu. Lze tedy říci, že není nutné zanevřít na doteď odvedenou práci pro vytváření responzivní aplikace. Responzivní design souvisí i s čistým HTML kódem a jednoduchými CSS styly. Jednodušší struktura dokumentu s sebou přináší mimo jiné zeštíhlení kaskádových stylů. Redukce kaskádových stylů může mít pozitivní vliv na výkon. A to i odstraněním nepoužitých stylů. Čistější kód CSS stylů zahrnuje i menší řetězení tříd. Řetězení tříd vytváří ty nejméně efektivní selektory a opět zpomaluje načítání webových stránek. Zjednodušení CSS je dokonce více v souladu s responsivním designem, pokud se minimalizuje množství svérázných řešení (tzv. hacků). Jelikož prohlížeč Internet Explorer 6 je minulostí, není důvod k přehnanému řetězení selektorů.

Při úpravě stávajícího projektu směrem k jednotnému zobrazení na všech typech zařízení jde ruku v ruce se změnou kódu i změna obsahu – zamyšlení se nad sémantikou. Typickým příkladem je semknutí a označení bloků textu jednoznačným HTML tagem nebo CSS identifikátorem. Ty jasně oddělují význam jednoho bloku od druhého. Při zobrazení výsledné stránky se však z jasné struktury stává obsah, jehož význam musí uživatel hledat v textu. Velice nápomocné je viditelné separování textu změnou velikosti písma u nadpisů, použití rozdílného fontu nebo uzamčení souvisejícího textu jednotným podbarvením.

2.3.7 Zhodnocení

Kniha Responzivní web design [12] argumentuje tím, že je nezbytná strategie, která by stavěla uživatele na první místo. Zatím se však vývojáři neshodli na jednotném přístupu. Také je údajně nepravděpodobné, že by se to někdy podařilo. Tady si zmíněná kniha silně protiřečí s publikací Dobrý designér to všechno ví [3]. V [12] je údajně cílem stavět uživatele na první místo. Kniha [3] naopak upozorňuje, že důležitějším faktorem pro návrh designu a vývoj je klient. Ten totiž zastřešuje celý projekt, financuje jej a především, pokud se výsledný produkt mine cílům a zájmům klienta, vyrobí se zmetek. V tu chvíli designer fatálně selhal.

2.4 Klient vs uživatel

V předchozí podkapitole byl zmíněn klient jako opomíjená osoba v publikacích o webdesignu. Většina literatury se totiž zaměřuje na stoprocentní spokojenost uživatele. To, jak bude vysvětleno v následujících odstavcích, je však jen polovina úspěchu produktu.

Petr Staníček ve své knize [3] zastává radikální postoj při popisu práce s klientem. Zkušenost se zjišťováním základních požadavků od klienta popisuje slovy: „Klient neví, co chce.“ Za poučeným klientem, který skutečně ví, co chce, podle něj stojí značná zkušenost a praktická znalost v oboru někoho z jeho členů týmu. Klient samozřejmě má požadavky. Také je schopný zacházet do podrobných detailů. Klientovy priority a cíle se však mnohdy liší od priorit interaktivní aplikace. Ve výsledku tak může být výsledným produktem lednice s nešťastně vyřešeným ovládáním. Tento problém se však může stát markantním ve chvíli, kdy celým produktem je sama aplikace.

Klientem může být dlouhodobě prosperující firma s dlouhodobým obchodním plánem. Má zpracované marketingové strategie, pravidelně realizuje SWOT analýzy a přemýšlí nad svými vizemi a cíli. Pořád jde ale jen o společnost, která rozumí pouze svému vlastnímu byznysu. Nastává zde jev, který by se dal nazvat projektová mezera. Projektová mezera je chybějící propojení znalostních bází designéra a jeho klienta. Jde o chybějící článek mezi odborností zadavatele a odborností jeho dodavatele, kterému zadává zakázku zcela mimo svůj obor činnosti. Jedna strana nerozumí byznysu strany druhé.

Klient má obvykle jakousi nejasnou představu o několika nedůležitých dílčích aspektech výsledné podoby. V takové situaci se ukazuje nejasný a neurčitý prostor mezi dvěma projekty – mezi klientovým zadáním nových webových stran a projektem softwarové společnosti na vyhotovení produktu klientovi. Prostor byl nazván projektová mezera. V tuto chvíli je možné přesně určit postavení webdesignéra. On je osobou, která má vyznačit onu zmíněnou projektovou mezeru a propojit oba projekty do jednoho funkčního celku. Jedním z hlavních úkolů designéra je částečně suplovat klientova chybějícího projektového manažera.

Klíčovým úkolem je získat od klienta všechny chybějící informace a sestavit je do zadání, které bude dávat smysl a zároveň s ním bude klient souhlasit. Jádrem problému je fakt, že klient vše podstatné ví. Klient má jasno v tom, co od svého podnikání očekává a kam jej chce směrovat. Jen netuší, jak tyto znalosti promítnout do zadání zakázky na webové stránky nebo online aplikaci. Jednoduše tvorba aplikací není jeho obor. Designér musí fungovat jako tlumočník z jazyka klienta do jazyka návrhářů a vývojářů (nebo naopak). Nejfatálnější jsou projektové mezery skryté. Jsou k nalezení v projektech, kde strana klienta a strana dodavatele není jasně daná a je nutné je vymezit uměle. Největším rizikem tedy může být paradoxně spolupráce s IT firmou, mediální agenturou apod.

Zájmy klienta a zájmy dodavatele jsou obvykle zcela rozdílné, ne-li protichůdné.

Zájmy klienta jsou klíčové, určují úspěch celého projektu a je zcela nezbytné jim vycházet vstříc. Zájmy dodavatele jsou až sekundární a limitují pouze jeho pohodlí a snadnost realizace. První zájmy určují profit a smysluplnost počínání strany, ve které figuruje designér. Otázkou však je, zda je realizace projektu rentabilní. Jedná se o jediný stupeň volnosti ze strany dodavatele. Zájmy a cíle klienta jsou prvořadé a lze z nich slevovat pouze v rámci jednání o ceně a parametrech zakázky. Pokud bude z jakéhokoli důvodu, ať už kvůli vlastnímu profitu, nebo třeba i v dobré víře projekt pozměněn a výsledek se mine klientovým cílům, došlo k nejfatálnějšímu selhání. Byl vyroben zmetek. Produkt nenaplňuje cíle zadavatele a tím neplní svůj účel.

Pro vytvoření kvalitního produktu a vyhnutí se chybám je nutné vytyčit 2 cíle:

- Zjistit a co nejlépe definovat skutečné zájmy a cíle klienta
- Vytvořit produkt, který je bude co nejlépe plnit

3 Konkurenční řešení

V roce 2001 uvedl britský mobilní operátor projekt „Orange-at-Home“. Spoluautor knihy Inside the Smart Home [30] pan Richard Harper jej popisuje jako hotový dům, který byl vybaven potřebnou infrastrukturou, připojením k síti a napojen na centrální server. Ten měl za úkol obsluhovat všechny funkce domu. Mimo jiné ovládal osvětlení, vytápění, zabezpečení, audio-vizuální techniku, záclony, vanu atd. Přístup k ovládání domu byl pomocí WAP, SMS nebo vytáčeného tónu. Účelem operátora bylo poskytnout příležitost k prozkoumání toho, co by uživatelé mohli využít, pokud jim poskytnete všechny aktuálně dostupné možnosti. Již z počátku se počítalo s tím, že ne všechny možnosti mohou být přijaty kladně. Zajímavé bylo, že toto smart řešení mělo značný úspěch v pracovním prostředí klientů. Naopak v domech klientely nebylo jednoduché se prosadit. Především nebylo jednoduché neinvazivně zakomponovat smart vybavení do interiérů a také se zdálo být příliš komplexní. Lidé se proto po zkušenosti začali vracet zpět k jednoduššímu způsobu ovládání. Dnes se však stává komplexní vybavení domů s prvky učení stále běžnějším vybavením domů. Následující společnosti se zabývají zmiňovanou problematikou a vystupují na trhu s vlastním řešením.

3.1 Loxone

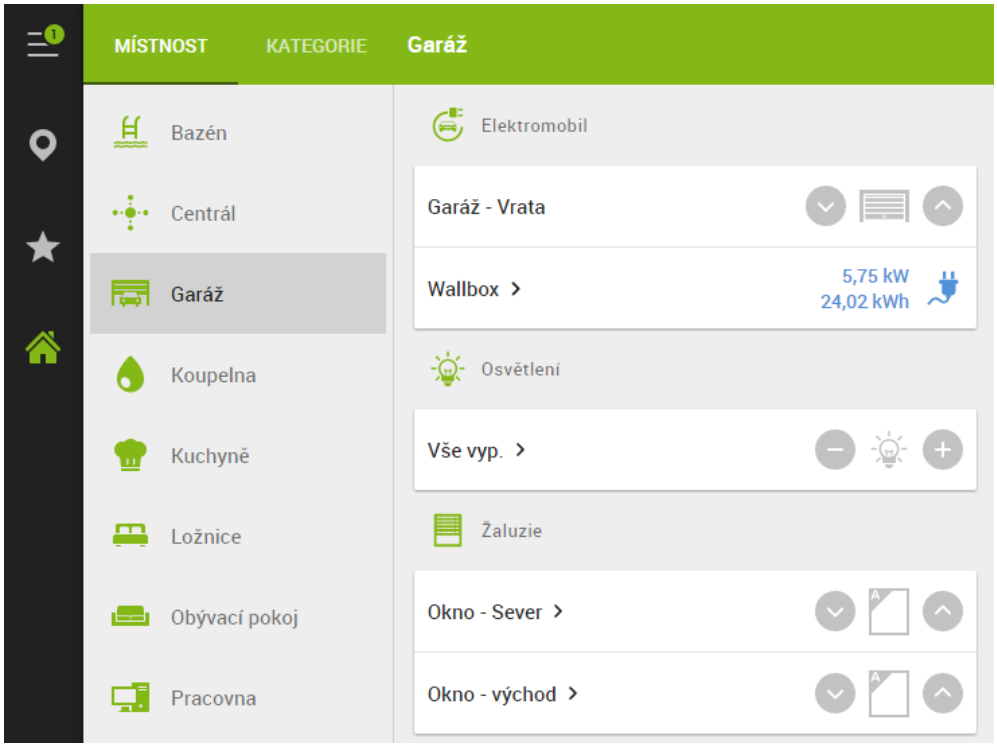
Loxone [31] je rakouská firma zabývající se modulárním a všestranným systémem domácí automatizace. Hlavním produktem je miniserver doplněný o rozšiřovací moduly. Ty si může zákazník libovolně dokupovat a jednoduše včlenit do své instalace. V tuzemsku nemá významnou historii, ale po technické stránce se jedná o jeden z nejvyspělejších systémů domácí automatizace na našem trhu. To dosvědčuje dobrá technická podpora, která s ukázkovou vstřícností zaslala materiály pro tuto práci.

Aktivní komunita uživatelů může být důležitým parametrem při volbě řešení pro vlastní byt či dům. Komponenty od firmy Loxone pokrývají všechny důležité potřeby ovládání. Samozřejmostí je řízení osvětlení, regulace teploty, žaluzií a rolet nebo zabezpečení. Nabízí však i ovládání bazénu, sauny nebo dobíjení elektromobilu. Ze softwarového vybavení je nutné vyzdvihnout Loxone Config. V něm si může klient snadným drag-and-drop způsobem propojit nově koupený akční prvek se stávajícím způsobem nebo jednoduše přeprogramovat ovládání tlačítek. Pro technicky zdatné typy je to perfektní způsob, jak si přizpůsobit systém na míru a hlavně sám. Není potřeba zásahu Loxone. Výborná je možnost otestování aplikace pro řízení inteligentních prvků domácnosti přímo v prohlížeči.

Následuje hodnocení prezentačního nástroje pro veřejnost. Konkrétně velké množství informací, které vytvářejí velmi dlouhé webové stránky a absenci pomocného tlačítka pro okamžitý přesun nahoru k menu. To může souviset i s celkem pomalým načítáním obsahu, který byl zobrazován na výkonných zařízeních s garantovaným připojením 20+ Mb/s. Ihned viditelná je fotografie zastupitele pro Česko a Slovensko s nedostatečným rozlišením a bez zaostření. Fatální je však použití fontu, jenž nezvládá diakritiku českého jazyka. Loxone nabízí překlad do 17 světových jazyků, což vytváří dojem velkoleposti. Je použit jeden font nejspíše z počátků rakouské organizace. Minimálně českému a slovenskému jazyku nebyla věnována žádná úprava fontu.

Loxone je určitě silným hráčem na trhu mezi společnostmi zabývajícími se inteligentní elektroinstalací. Její technologické možnosti a recenze klientů přidávají na důvěře. Špatné zobrazování znaků s diakritikou a další podobné designové přestupky dávají najevo neosobní přístup takto velké společnosti k českému trhu. Na obrázku Obrázek 5 - Aplikace firmy Loxone je vyobrazena aplikace pro řízení inteligentního domu. Levé menu obsahuje panel pro přístup k ovládání, kde je možné vyhledávat podle místností nebo kategorií prvků. Dále lze nalézt panel Oblíbené a úvodní obrazovku s polohou a aktuálním počasím.

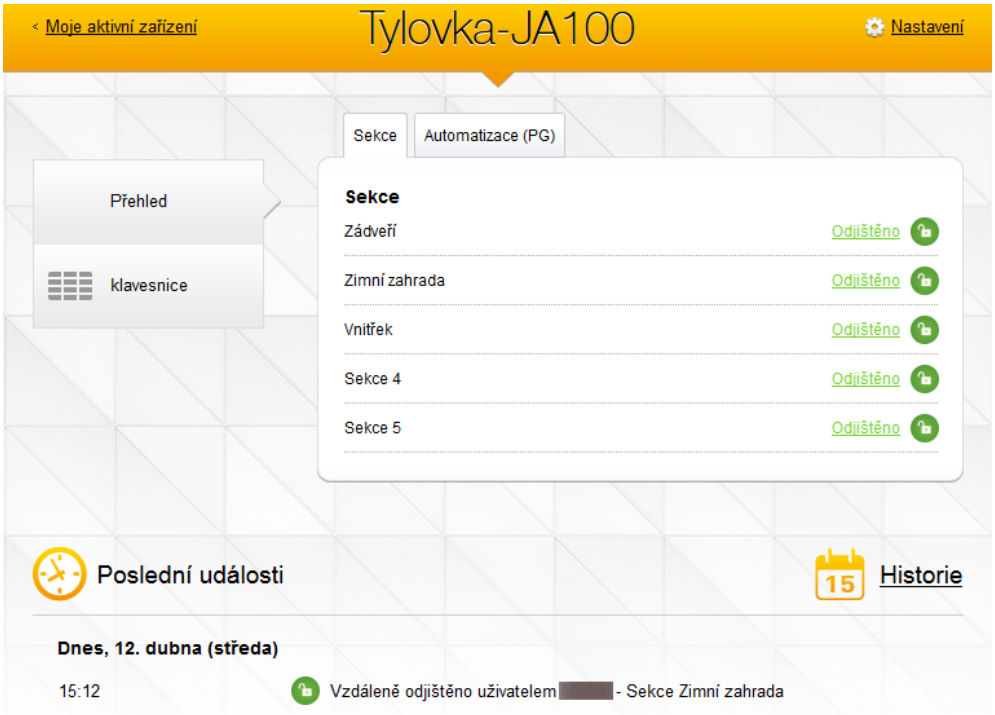
Vše je intuitivní a dá se najít tam, kde to uživatel očekává. Víceméně jde o kvalitní práci programátora, která se neminula požadavkům uživatele. Aplikace neurazí, ale nic navíc nenabídne. Podstatnou obrazovku pro řízení vybavení představují 2 seznamy. Jeden pro lepší orientaci mezi desítkami prvků a druhý detailní s akčními tlačítky jedno na druhém. Je to běžný typ ovládání dostupný i v open-source projektech jako openHAB [32]. Zelená barva společnosti je výborně použitelná pro webovou aplikaci. Písmu se nedostala větší pozornost. Oddělení menu od běžného textu se děje zvýrazněním do velkých písmen. Z širšího pohledu má aplikace perfektní čitelnost a neporazitelnou jednoduchost. Je vidět, že míří na opravdu širokou klientelu, a tak zde není prostor na jediný experiment, zajímavost nebo příjemné překvapení. Je těžké určit, z čeho design vychází, ale jednoduché, k čemu míří. Nabízí se srovnání s bezplatně dostupnými responzivními šablonami.



Obrázek 5 - Aplikace firmy Loxone

3.2 Jablotron

Jablonecká společnost Jablotron [33] má na českém trhu výsadní místo. Právě proto byla vybrána k porovnání konkurence jako zajímavá varianta k zahraničnímu a „nadnárodnímu“ Loxone. Opět je hlavním produktem centrální prvek – tentokrát s názvem Jablotron 100. Firma se zabývá především zabezpečením domů a bytů a tomu odpovídá i její nabídka. Mezi dostupnými produkty je možné najít kompletní vybavení pro zabezpečení domácnosti, a to včetně kamerového systému, vzdáleného ovládání pomocí internetového připojení či komunikace s centrálou prostřednictvím GSM brány. Dále lze chytrý dům vybavit ovládáním osvětlení, regulací topení nebo zkombinovat dostupná relé k vlastnímu řešení. Nabídka čítá drátové i bezdrátové řešení.

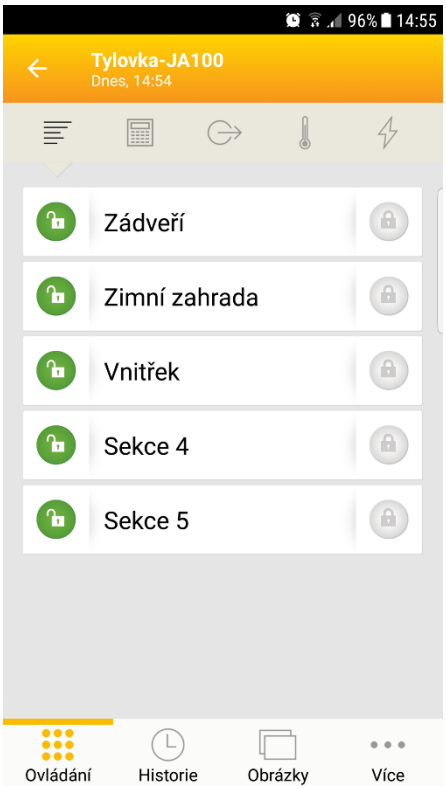


Obrázek 6 - Webové rozhraní společnosti Jablotron

Všechny funkce je možné ovládat z webového rozhraní MyJablotron (Obrázek 6 - Webové rozhraní společnosti Jablotron), ke kterému se dá připojit odkudkoliv z webového prohlížeče. Webová aplikace je hlavním produktem pro pohodlné ovládání. Je plně responzivní. Bílá barva je neutrální. Oranžová barva má rušivý efekt. Podle kapitoly 2.2.1 Grafický design silněji působí už jen červená. Jasně však ctí grafickou prezentaci společnosti.

Uživatelská rozhraní Jablotron se vyznačují klasickým menu pro přechod mezi obrazovkami. V rámci nich se uživatel pohybuje přes menu druhé úrovně. Jde o složitější, a ne úplně intuitivní způsob. Oproti webovému rozhraní Loxone působí zajímavě. Polovinu akčních prvků (zejména v horním menu) představují běžné odkazy. Je zde prostor ke zvýraznění a obohacení rozhraní. Mnohem výraznější nedostatek představuje písmo. Tam, kde by bylo žádoucí jasně oddělit části obsahu, patří jiný font. Nadpis sekce je ukázkové místo aplikace, kde chybí patkové písmo upoutávající pozornost a vizuálně oddělující prostor nastavení od historie. Menu obsahuje nekonzistenci malých a velkých písmen. Prakticky jediný zájem o písmo značí použití tučného písma. Jde o dobrou práci vývojáře bez působení designéra.

Alternativou je aplikace pro mobilní platformu Android se stejným názvem. Pro hodnocení grafické stránky platí stejné výhody i nedostatky jako pro webovou aplikaci. Nabízí se použití stejných grafických komponent jako u webové aplikace. Ovládání zabezpečení je na mobilní aplikaci povedené. Špatným dojmem působí zobrazení bílého textu na žlutém pozadí horního panelu.

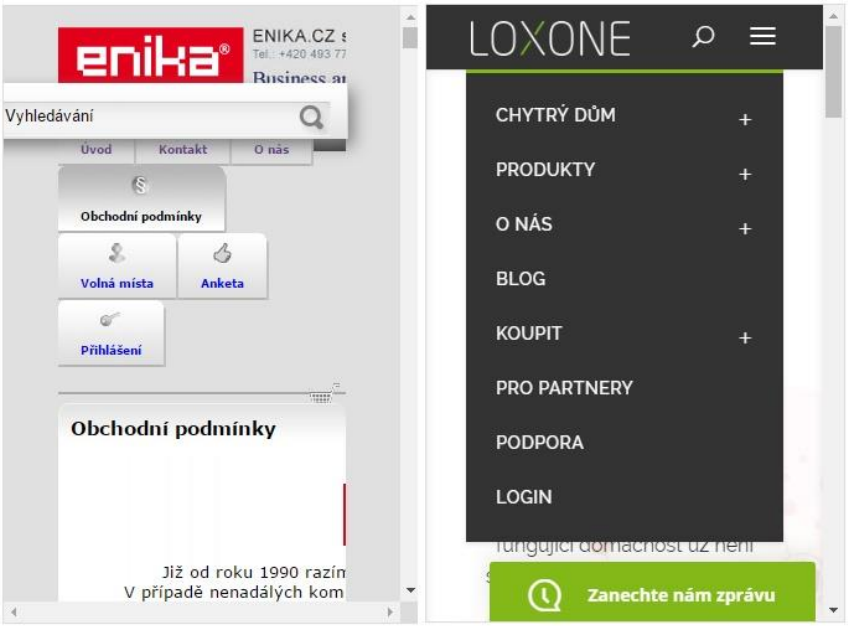


Obrázek 7 - Aplikace MyJablotron pro mobilní paltformu Android

3.3 Enika

Třetí společnost byla vybrána zejména kvůli odlišnosti od zbylých dvou. Společnost ENIKA.CZ se prezentuje jako ryze česká firma založená roku 1990. Její pole působnosti je omezeno na Českou a Slovenskou republiku, kde nabízí komponenty pro osvětlení a automatizaci. V posledních letech se snaží prosadit mezi konkurencí s vlastním propojením hotových komponent. Starší systém BOSys funguje na frekvenci 433 MHz. I přes benefit bezdrátového ovládání zůstává v nabídce nejspíše proto, že jeho nástupce není zpětně kompatibilní s novými komponentami. Ty tvoří údajně nová generace systému s názvem Poseidon. Liší se v použité pracovní frekvenci 868 MHz. Firma zmiňuje směr moderních domů, které upřednostňují pohodlí, komfort, úsporu a flexibilitu. Flexibilita je samozřejmě zajištěna již samotným bezdrátovým řešením. To je nasazený standard, bez kterého není možné obstojně působit na trhu. Mimo vyobrazení bezdrátového vysílače a přijímače neexistuje žádná zmínka inteligentního řešení. Během posledních 4 let firma neprojevila posun vývoje vpřed.

Obrázek 8 - Ukázka responzivní webové aplikace porovnává webové portály společností Loxone a Enika. Při rozlišení 320x480 zobrazovacích bodů jsou patrné jasné výhody webových dokumentů splňujících standardy responzivity. První z portálů je v této situaci nepoužitelný pro ovládání.



Obrázek 8 - Ukázka responzivní webové aplikace

Webová stránka firmy ENIKA.CZ nedokáže fungovat na malém rozlišení – konkrétně rozlišení 320x480 zobrazovacích bodů. Vyhledávací tlačítko překryje první 3 volby menu, čímž znepřístupní obsah návštěvníkům a potencionálním klientům. Dalším selháním je obsah položky Ceník. Ten k datu 30. 1. 2017 obsahuje zastaralý ceník z roku 2016 a nefunkční zobrazení obrázku. Otázkou zůstává, zda je tato společnost konkurenceschopná bez nabídky jakékoliv aplikace pro řízení akčních členů ať z desktopu, či přenosného zařízení.

Tabulka 1 - Srovnání konkurečních řešení

	Webová prezentace	**Responzivita**	**Barvy**	**Písmo**	**Grafické kompo-nenty**
Loxone	loxone.com	Ano	Vhodné	Nedostačuje	Seznam
Jablotron	jablotron.com	Ano	Nevhodná žlutá	Nedostačuje	Dvě úrovně menu
ENIKA.CZ	enika.cz	Ne	X	X	X

4 Postup návrhu webové aplikace

Před prvním náčrtem a prvním řádkem kódu je nutné rozhodnout o postupu vývoje. Metodik vývoje je velké množství. Mezi hlavní proudy v metodických přístupech lze zařadit rigorózní metodiky a agilní přístup. Jejich popisu a porovnání vybraných přístupů se věnují následující dvě kapitoly.

4.1 Rigorózní metodiky

Rigorózní metodiky vycházejí z přesvědčení, že budování informačních systémů lze popsat, plánovat, řídit a měřit. Snaží se podrobně a přesně definovat procesy, činnosti a vytvářené produkty. Při tomto způsobu vývoje probíhají jednotlivé fáze (plánování, analýza, návrh, implementace, zavedení) sekvenčně za sebou. Existují ale také rigorózní metodiky založené na iterativním a inkrementálním vývoji. Příkladem takové metodiky je Rational Unified Process (dále jen RUP).

4.1.1 Rational Unified Process

RUP je metodika založená na tzv. nejlepších praktikách softwarového vývoje. Kategorizaci a popisu hlavních proudů metodik se věnuje publikace Metodiky vývoje a údržby informačních systémů [22] od autorky Buchalcevové. Mimo jiné popisuje i vybranou metodiku. Metodika RUP je založena na tzv. nejlepších praktikách softwarového vývoje:

- Iterativní vývoj
    - Ideální je znát všechny požadavky v předstihu; to je však pouze ideální případ a právě s problémem vývoje ve fázích počítá iterativní vývoj
- Řízení požadavků
    - Vždy mít na paměti požadavky uživatelů
- Použití komponentové architektury
    - Umožňuje opětovné použití kódu
    - Umožňuje jednodušší testování po částech
- Vizuální modelování
    - Použití diagramů a grafů
    - Specifikace UML slouží k standardizované a zároveň intuitivní reprezentaci komponent, uživatelů a jejich vzájemné interakci
- Kontrola kvality softwaru
    - Kladení důrazu na testování

- Řízení změn
    - Dílčí části projektu se mohou řešit v různých týmech, lokacích nebo alespoň na jiných platformách
    - Řízení změn upozorňuje na nutnou synchronizaci jednotlivých částí

Životní cyklus softwaru je rozdělen na podcykly. Předmětem každého z nich je nový produkt. Jeden vývojový cyklus je v Rational Unified Process rozdělen do 4 po sobě jdoucích fázích:

1. Počáteční fáze
2. Fáze rozpracování
3. Konstrukční fáze
4. Fáze nasazení

Zahrnutí RUP mezi rigorózní metodiky lze obhájit podrobným definováním procesů a činností při vývoji softwaru. Cílem je výhradně zaměření na vývoj nového řešení. Obecným a podstatným nedostatkem této metodiky je zaměření pouze na úroveň projektu. Výsledkem je poměrně malý rozsah, neboť se zaměřuje pouze na vývoj řešení a nezahrnuje provoz ani údržbu.

4.2 Agilní metodiky

Rigorózní metodiky mají několik nedostatků. Vycházejí z předpokladu, že požadavky je možné specifikovat předem. Snaží se zabránit změnám. Jako celek jsou velmi náročné. Během jednotlivých fází vzniká několik meziproduktů. Výsledkem je vychýlení z přímé cesty k fungujícímu software, který odpovídá požadavkům uživatelů.

Agilní metodiky mají několik společných principů. Jsou charakteristické iterativním vývojem s velmi krátkými iteracemi. Zaměřují se na fungující systém s maximální hodnotou pro zákazníka. Toho je docíleno pouze důrazem na spolupráci a komunikaci se členy týmu i zákazníkem. Agilní metodiky jsou tolerantní ke změnám a automatizovanému testování.

Pojem *Agilní* byl zaveden v roce 2001 v Utahu. Právě zde se na jednom místě setkalo 17 vývojářů a představitelů tzv. *lightweight software development methods*. Společně se shodli na základních principech agilního přístupu, podepsali *Manifest agilního vývoje softwaru* a vytvořili *Alianci pro agilní vývoj softwaru*. Jeden z účastníků, Martin Fowler, tvrdí [25], že manifest deklaruje čtyři hodnoty, přičemž prvky na levé straně mají větší relativní význam než prvky na pravé straně . V manifestu se uvádí:

„Odhalili jsme lepší způsob vývoje software, sami jej používáme a chceme pomoci ostatním, aby jej používali. Z toho pohledu dáváme přednost:

- Individualitám a komunikaci před procesy a nástroji
- Provozuschopnému software před obsažnou dokumentací
- Spolupráci se zákazníkem před sjednáváním kontraktu
- Reakci na změnu před plněním plánu“

4.2.1 Scrum

Jednou z nejčastěji používaných agilních metodik je Scrum. Vývoj software není definovaný proces, jak rigorózní metodiky předpokládají, ale empirický proces. Na tomto přesvědčení je Scrum založen. Scrum je adaptivní, rychlý a samoorganizující. Vývoj probíhá v rámci několikatýdenních iterací nazývaných sprint. Na konci sprintu je předvedena a nasazena nová užitná vlastnost.

Klíčovou praktikou metodiky je absolvování každodenních 15minutových porad (tzv. Scrum Meetings), které slouží pro koordinaci a integraci prací. Denní porady se konají vždy ve stejný čas a na stejném místě a v praxi je mohou navštěvovat i manažeři (bez zásahu do porady). Celou poradu řídí jeden Scrum Master. Denní porady umožňují týmu sdílet znalosti. Každý ze zúčastněných mustí zodpovědět tři otázky:

- Které položky dokončil od minulé poradě?
- Které nové úkoly má řešit?
- Jaká vidí omezení a překážky pro řešení úkolů?

Metodika Scrum je založena především na řízení projektu. Podle článku Limitations of Agile Software Processes [26] je Scrum jednou z přístupnějších metod řízení projektu pomocí agilního vývoje. Dokáže ustoupit nátlaku na snižování nákladů a současně zůstat plně použitelný. Obecně však metodiky agilního vývoje nemusí být vždy nejlepším řešením, a to především při aplikaci na dlouhotrvající a komplexní systémy. Následují fáze metodiky Scrum:

- Plánovací fáze
    - specifikace funkčních požadavků, plán dodávek, architektonická a byznys vize
- Vynášecí fáze
    - do souboru požadavků (backlog) jsou připojeny nefunkční požadavky jako bezpečnostní, výkonnostní apod.

- Fáze vývoje (sprinty)
    - týmy dodávají funkcionalitu s nejvyšší prioritou každých 30 dní
    - na konci každé iterace tým předvede výsledek
- Fáze dodávky
    - produkt je předán uživatelům

Rational Unified Process je zdokumentovaný, podložený modely a grafy. Jeho těžkopádné první fáze vytvářejí několik redundantních meziproduktů, zpomalují celý proces vývoje a znemožňují dodatečné zásahy do hotového řešení. Scrum je naopak agilní metodikou. Dokáže pružně reagovat na změny v požadavcích. Nevýhodou je absence dokumentace a podložených postupů. Je nejlépe použitelný pro tým 2 až 10 lidí nebo více takových tymů. Tento přístup však lépe vyhovuje cíli této práce.

4.3 Specifikace potřeb

Specifikace potřeb je jedna z prvních činností při vývoji softwaru. Úzce na ní spolupracují řešitel i zadavatel. Zadavatel definuje a vysvětluje svá očekávání od systémů. Řešitel formalizuje a zaznamenává požadavky. Agilní přístupy se ve většině případů distancují od case nástrojů. Stále je však potřeba prvně pochopit vybranou problematiku, a právě k tomu je ideální alespoň zjednodušený use case diagram dostupný v přílohách [1]. Jeden obecný diagram, který nebude nutné upravovat při každé změně požadavků, dokáže rychleji přiblížit problematiku dalším členům týmu a zároveň je spojením mezi zadavatelem a řešitelem.

Specifikace požadavků je jedním ze základních dokumentů využívaných ve všech etapách vývoje informačního systému. Podle publikace Analýza a návrh informačních systémů [35] vzniká právě během zjišťování a analýzy nejvíce závažných chyb. Jejich odstranění v době již realizovaného systému je velmi nákladné. Podle zmíněné publikace je odstranění chyb po skončení projektu až 200x nákladnější, než kdyby chyba byla identifikována a odstraněna již v zárodku.

Pro vyvarování se chyb lze zavést několik opatření. Samozřejmostí je dobrá a intenzivní spolupráce obou stran. Předejití chyb zajistí důsledné omezení nereálných požadavků. Během systematické práce vývojářů je vhodné zavést priority a pracnost dílčích částí. Požadavky na systém musí splňovat alespoň základní kritéria, která zajistí přesnější implementaci. Základem je srozumitelnost požadavku. Analytik by měl vytvářet takové požadavky, které budou čitelné pro něj, vývojáře i zákazníka. Se srozumitelností souvisí jednoznačnost požadavku. Tomu napomůže dodržování terminologie a jednoduchý styl. Kritickou vlastností požadavku je jeho proveditelnost. To znamená, že požadavek musí být implementovatelný vzhledem ke všem známým omezením systému, omezením daným rozsahem projektu a jiným technickým omezením.

Posledním zásadním projevem správného požadavku je jeho ověřitelnost. Ověřitelnost znamená schopnost potvrdit, že byl daný požadavek opravdu splněn. Příkladem nesplnění této vlastnosti je neúplný požadavek na rychlost. Vybraná vlastnost systému má být rychlá. Takový požadavek nelze navrhnout a ani řádně otestovat. Rychlost lze vnímat subjektivně, a pokud zákazník nebude s řešením spokojený, zadavatel jej nemůže považovat za hotový. Pokud má být podle zadání vyřízen požadavek do 3 sekund, je možné vlastnost otestovat, jednoznačně říci, zda byla splněna a teprve poté teprve s právem požadovat finanční odměnu. Vybraná metodika Scrum přidává povinnost shrnutí funkčních i nefunkčních požadavků, které jsou tématem následující podrubriky.

4.4 Funkční požadavky

Požadavky lze dělit do kategorií. Nejčastěji jsou uváděny funkční a nefunkční požadavky. To souvisí s vymezením požadavku jako jedné vlastnosti systému. Kategorie požadavků spolu úzce souvisí a platí mezi nimi vzájemné vztahy. Požadavky jednotlivých druhů se nesmí vzájemně popírat či dokonce vylučovat. Stejně problematické by mohly být v budoucnu duplicity. Zde bude přiblížen první z nejběžnějších typů požadavků, a to funkční požadavky.

Funkční požadavky přesně zastupují pojem softwarový požadavek. Podle knihy UML 2 a unifikovaný proces vývoje [27] jde o popis toho, co by měl systém umět. To jinými slovy znamená, že funkční požadavky popisují požadovanou funkci systému. Právě proto jsou nazývány též behaviorální požadavky. Funkční požadavek je možností či postupem, který má uživatel dostat. A právě podle těchto kritérií byly seskupeny požadavky pro tuto práci dostupné v přílohách [2]. Popisují nástroje a postupy pro dosažení podnikatelských cílů. Musí být tedy v souladu s těmito cíli. Souvislost požadavků se záměry zadavatele je již zřejmá. Na druhé straně je specifikace požadavků manuálem pro vytváření nového systému. U rozsáhlejších zadání je vhodné seskupovat požadavky do skupin i v druhé vrstvě a vytvářet tím určitou hierarchii. Nabízí se rozdělení na podkategorie uživatelské požadavky, které popisují interakci mezi uživatelem a systémem. Druhou podkategorii mohou tvořit systémové požadavky. Ty popisují chování ve vztahu systém – systém.

4.5 Nefunkční požadavek

Druhou kategorií pro zatřídění jsou tzv. nefunkční požadavky. Do ní se řadí ty, které formulují omezení kladená na systém nebo proces vývoje. Pro vytvářený projekt specifikují omezující podmínku. Funkční požadavky mohou mít více vrstev a seskupovat se do různých kategorií např. podle názvosloví oboru, kterému se zákazník věnuje. Nefunkční požadavky se však napříč různými typy systémů příliš neliší a vyplatí se použít některé z následujících podkategorií. Jsou jimi požadavky na výkon nebo kapacitu systému. Zpracované nefunkční požadavky jsou k nalezení v přílohách [3]. Častý je požadavek na dostupnost, při jejímž nesplnění jsou zároveň stanoveny sankce. Mezi nefunkční požadavky spadají i důležité nároky na zabezpečení aplikací.

Výše zmíněná kniha vysvětluje popsané pojmy. Kromě toho je označuje za základní, a přitom silné a postačující. Dále doporučuje postup, při kterém jsou nejdříve sestaveny funkční požadavky. Po nich následuje druhá kategorie nefunkčních. Z praxe se osvědčuje třetí, neméně důležitý krok přiřazení atributů ke všem požadavkům. Takovým atributem může být jednoduché označení priority od 1 do 3, čímž okamžitě vznikne seznam vytříbených úkolů, na nichž je potřeba začít pracovat jako s předností, ne-li urgentně.

4.6 Řízení přístupu

Tato podkapitola popisuje vztah mezi uživatelem a systémem. Řízení přístupu uživatelů k obsahu webové aplikace je již standardem. Cílová skupina, rodina využívající systém ovládání chytrých domů, se skládá ze dvou dospělých lidí, dvou dětí a případných návštěv. Je žádoucí zamezit dětem přístup ke spínání vytápění a ovládání poplašného zařízení. Ještě nebezpečnější situace může nastat při potřebě nabídnout ovládání domu (např. osvětlení obývacího pokoje) cizí osobě.

Mechanismů kontrol je hned několik. Tím nejčastěji používaným je systém založený na rolích. Článek s názvem A Unified Administrative Model for Role-Based Access Control [28] se věnuje právě řízení přístupu založeném na rolích. RBAC je tolik oblíbený díky jeho flexibilní správě. Jsou jasně definované role s více či méně omezeným přístupem. Uživatel spadá do jedné ze skupin. Kdykoliv je možné přidávat skupiny a řadit do nich uživatele nebo naopak přidávat uživatele s přiřazenou rolí.

V předchozích kapitolách bylo uvedeno, že vzhledem k rychlosti a cenně vývoje je ideálním způsobem navrhování atomických prvků. Ty se mohou libovolně nořit do ostatních nebo ležet hierarchicky na stejné úrovni. I zde se dá samozřejmě použít řízení přístupu na bázi rolí. Článek Component-Based Access Control: Secure Software Composition through Static Analysis [29] se věnuje rozšiřitelným platformám složených z komponent. Středem zájmu je management komponent za běhu systému. Nejde přímo o oblast této práce, ale zjištěných poznatků bylo využito. Analogicky k vybranému článku byl sestaven systém řízení přístupu založeném na komponentách. Pokud by grafické uživatelské rozhraní bylo generováno ze znovupoužitelných prvků, je možné ještě před renderováním každého prvku jednoduše rozhodnout, zda jej uživatel vůbec může vidět. Společně s atributem priority uživatele byl sestaven uchopitelný obraz ověření přístupu.

5 Technologie IS

Informační systém je robustní webovou aplikací. Již od počátku je navrhován pro větší množství uživatelů s frekventovanějším, datovým tokem, propracovanějším, bezpečnostním hlediskem apod. O to kritičtější musí být samotná analýza a výběr vhodných technologií. Veškerý výběr nástrojů a technologií je nutné zdůvodnit. Tzn. výsledek praktické části závisí na hledání kvalitních a aktuálních zdrojů, poté na výsledcích porovnání. Vývoj nových technologií je natolik rychlý, že i během dvouletého vývoje bylo potřeba přehodnotit tyto výsledky. V následující kapitole bude popsán důvod výběru právě oné varianty.

5.1 Koncept webové aplikace

Pro začátek je nutné popsat termín webový framework a rozdělit jej na dvě části, a to webová aplikace a framework. V následujících odstavcích budou tyto pojmy vysvětleny.

5.1.1 Webová aplikace

„Webová aplikace je softwarový systém založený na technologiích a standardech World Wide Web konsorcia (W3C), které poskytuje zdroje zaměřené na web jako obsah a služby skrz uživatelské rozhraní, webový prohlížeč.“ Publikace Web engineering [36]

5.1.2 Framework

„Framework je model konkrétní domény nebo jejím důležitým aspektem. Framework může modelovat jakoukoliv oblast, ať už se jedná o technickou oblast jako je distribuce nebo odvoz odpadu, nebo aplikační domény, kterými mohou být bankovnictví nebo pojišťovnictví. Framework poskytuje opakovaně použitelný design a opakovaně použitelné implementace ke klientům.“

„Softwarový framework je sada kódu nebo knihoven, které poskytují funkčnost společnou pro celou třídu aplikací. Zatímco jedna knihovna obvykle poskytuje jeden specifický kus funkčnosti, framework poskytuje širší škálu používanou jedním typem aplikace." Disertační práce Framework Design [37]

5.1.3 MVC

Informační systém v podobě webové aplikace je možné separovat do tří základních bloků. Prvním z nich je prezentační vrstva zodpovědná za funkční, přehledné a uživatelsky přívětivé grafické rozhraní. Druhá vrstva je klíčová z pohledu transformace dat mezi prezentační a datovou vrstvou. Business vrstva by měla odstínit prezentační vrstvu od datové a jasně definovat a zdokumentovat rozhraní – typicky ve formě webových služeb. Třetí z primárních vrstev je vrstva datová. Ta zajišťuje cílené uložení informací a jejich další zpracování. Jedná se často o heterogenní složku systému. Zahrnuje systémy s primárními daty, specializovaná interní datová úložiště a reportovací nástroje. Pro tuto implementaci existuje návrhový vzor Model – View – Controller [38], který pokrývá všechny tři zmíněné vrstvy.

V analýze je potřeba vybrat nástroj, který je schopen pokrýt veškeré požadavky. Nástroj by měl být schopný nasazení návrhového vzoru MVC. Výhodou by bylo kompletní řešení od návrhu grafického uživatelského rozhraní až po integraci databáze. Podle webového portálu BuiltWith [39], který se od roku 2007 zabývá vytvářením statistik použitých webových technologií, mají majoritní zastoupení právě tři skupiny frameworků.

První skupinou jsou frameworky programovacího jazyka PHP. Jejich podíl tvoří celých 28 % ze všech použitých technologií. Jazyk PHP tím potvrzuje pozici snadno aplikovatelného a komunitně oblíbeného jazyka. Na druhém místě se s 18 % vyskytuje ASP .NET. Nástroj .NET je soubor technologií od společnosti Microsoft určených pro vývoj pro Web, Windows i mobilní zařízení. Active Server Pages jsou specializovány přímo pro vývoj webových portálů. Významnou trojici uzavírá Java EE. Jedná se o enterprise edici platformy Java. Java EE od firmy Oracle je zaměřena na vývoj komplexních informačních systémů a webových portálů. Podobně jako u PHP je možné vybrat si z několika frameworků.

5.1.4 PHP

PHP frameworky jsou podle zjištěných výsledků nejpoužívanější. Pro účel řízení inteligentního domu je jedním z nejdůležitějších aspektů modulovatelnost a znovupoužitelnost kódu. Z tohoto důvodu vychází požadavek na silně objektový jazyk, jenž jazyk PHP nesplňuje.

5.1.5 ASP .NET

Zbývá rozhodnout mezi ASP .NET a Java EE. Z tohoto porovnání vyšla pro tento případ lépe enterprise edice jazyka Java. Její klady v podobě lepší podpory objektově relačního mapování, např. JPA, Hibernate, EclipseLink, oproti Entity frameworku dostupného pro ASP .NET. Podobně je v Java EE propracovanější přístup Context Injection. Komunita vyvíjející pod platformou Java EE a výběr vývojových prostředí jsou posledními body přispívajícími frameworkům Javy.

5.1.6 Spring

Spring je open-source aplikační framework využívající vzor dependency injection a inversion of control kontejner pro platformu Java. V běžném objektově orientovaném přístupu se objektům vytváří závislost nebo se sami musí dotázat tzv. Object Factory, aby jim potřebnou závislost vytvořila. Místo toho je v přístupu dependency injection závislost předána v konstruktoru nebo pomocí settrů a o konkrétní přiřazení závislosti se vývojář nestará. Závislostí je zde myšlen druhý objekt, na který potřebuje mít první objekt referenci. Podle oficiální dokumentace pro framework Spring [40] je Spring MVC jedním z modulů celého Springu.

Návrhový vzor MVC napomáhá návrhu celé aplikace. V tomto případě určuje jeden hlavní Controller (tzv. front controller), který bude zachytávat veškeré požadavky z prezentační vrstvy a předávat je dále na specificky postavené controllery, které volají a spouští k nim vztaženou aplikační logiku. Tímto popisem byla jasně specifikována střední vrstva aplikace. Ta začíná front controllerem a končí aplikační logikou, která přistupuje k datům. Spring je velice rozšířený framework, jehož verze 1.0 byla představena v roce 2004. Byl to silný nástroj pro budování náročných webových aplikací na platformě Java EE. Byl alternativou k tehdy nekonkurenceschopnému Enterprise Java Beans modelu. Díky jeho přednostem je oblíbený a rozšířený, ale díky jeho komplexnosti je náročný na pochopení, ještě více na bezproblémovou konfiguraci a s vlastním směrem vývoje není schopen implementovat všechny nové prvky Java Enterprise Edition.

5.1.7 JavaServer Faces

JavaServer Faces je Java specifikací pro vytváření komponentově orientovaných uživatelských rozhraních pro webové aplikace. Servlet (FacesServlet) zpracovává požadavky, načítá odpovídající view šablonu a renderuje odpověď pro klienta – nejčastěji v podobě HTML. Každé zobrazené view má v pozadí tzv. Manage Bean, která udržuje stav šablony a zprostředkovává operaci aplikační logiky. Velkou předností JSF je její učící křivka. Pokud má vývojář zkušenost s Enterprise verzí platformy Java, tak neznámou oblastí je pouze práce s uživatelským rozhraním a jeho stavem (udržovaném v Manage Bean), který si právě vyrenderovaná šablona drží při komunikaci s uživatelem. Cagatay Civici (vedoucí vývoje PrimeFaces) se v roce 2014 postavil čelem ke kritice JSF od Thoughtworks Technology Radar [41]:

„JSF is a stateful framework by nature and state makes web applications easy to develop with. With improved state management techniques introduced in JSF 2.0+ (e.g. stateless mode, partial state saving), JSF can scale as well.“

„JSF has been around for 10 years, if it was a bad framework, it will easily get lost in the jungle of web frameworks of Java. Various frameworks has reached their technology peek during past 10 years and started to fade away already. In JSF case, First Facelets saved it, JSF 2.0 fixed it along with JSF 2.2 and then PrimeFaces has given it a popularity boost. Clearly it solves a problem better than others which is creating web applications with an easy, efficient and productive way to meet deadlines. With JSF 2.2, using HTML friendly markup and pass through attributes it caught up with latest trends as well.“

Spring byl v době vytvoření výbornou konkurencí k soudobým Enterprise Beans. Jak výše popsal Cagatay Civici, Java EE se specifikací JavaServer Faces prošla během posledních 10 let radikálními změnami a pomocí nových přístupů dohnala a v použitelnosti dokonce i předčila Spring. Jednoduchý princip JSF v podobě Manage Bean, moderní knihovna komponent PrimeFaces a další technologie např. pro stylování prezentační vrstvy jako např. BootsFaces [42] rozhodují pro výběr specifikace JavaServer Faces.

5.2 Java EE a JavaServer Faces

Pokud je potřeba kolekce objektů, Java vývojář nezačne navrhovat vlastní řešení hash tabulky. Využije hotový balíček kolekcí. Stejně tak pokud vývojář potřebuje jednoduchou webovou aplikaci nebo zabezpečenou a distribuovanou aplikaci využívající transakce, nebude navrhovat vlastní low-level API. Využije Enterprise Edici platformy Java. Java EE je množina specifikací vyvinutá nejen pro komplexní podnikové systémy. Tato podkapitola má přiblížit základní pojmy, se kterými je potřeba se při vývoji webových aplikací na platformě Java seznámit.

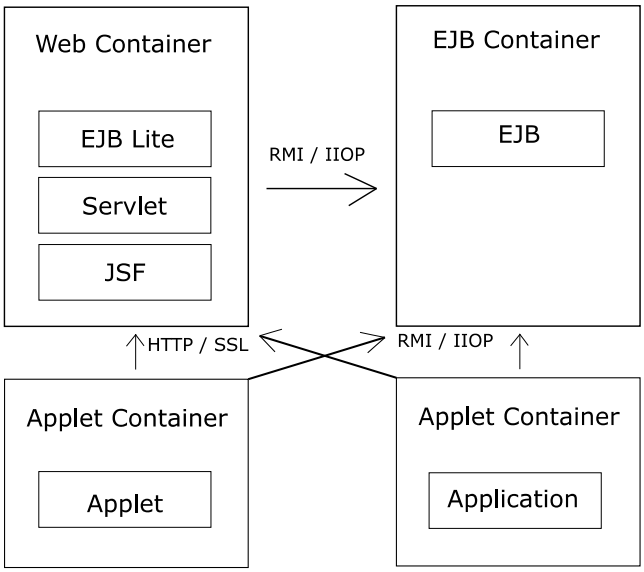
5.2.1 Architektura

Specifikace jsou implementovány uvnitř tzv. kontejnerů. Kniha Beginning in Java EE 7 [43] se zabývá historií Java EE, popisuje nové prvky 7. verze, a především vysvětluje celou její architekturu. Podle této publikace je kontejner běhové prostředí (runtime environment) pro Java EE, které nabízí komponentám služby jako správa životního cyklu (life-cycle management), dependency injection a další.

Komponenty mají přesně definovaný způsob komunikace uvnitř Java EE infrastruktury (mezi jednotlivými komponentami). Také musí být před nasazením zkompilovány do standardizovaného archivu. Java EE je nadřazena standardní edici Java (Java SE), což znamená, že jakákoliv Java EE komponenta může využívat Java SE API.

Níže je Obrázek 9 - Architektura Java EE, který zobrazuje vztah mezi jednotlivými kontejnery. Obdélník představuje kontejner a šipky představují protokoly, pomocí kterých se mezi sebou kontejnery dorozumívají. Jednou z komponent jsou webové aplikace složené ze servletů, servletových filtrů, web-event listenery, JSP a JSF. Jsou spouštěny ve webovém kontejneru a zprostředkovávají odpověď na HTTP požadavky z webového klienta.

K tomu, aby webová aplikace byla schopná zachytit a zpracovat zprávu, potřebuje již zmíněný servlet. Jednoduše řečeno je servlet pouze třídou, která odpovídá na určité typy požadavků ze sítě. Těmi bývají nejčastěji HTTP požadavky. Servlety běží v kontejneru, který zpracovává síťovou komunikaci (např. parsuje HTTP požadavky). Servlety také zajišťují SOAP a RESTful webové služby, které jsou dnes velice oblíbené pro komunikaci mezi jednotlivými aplikacemi využitím k přenosu informací právě HTTP protokolu.



Obrázek 9 - Architektura Java EE

Již několikrát zde bylo zmíněno spojení Inversion of Control (dále jen IoC). Tento návrhový vzor znamená, že kontejner přebírá kontrolu nad aplikační logikou a nabízí služby v podobě řešení transakcí a zabezpečení. Přebírání kontroly znamená, že kontejner sám spravuje životní cyklus komponent a pokud potřebují referenci na jinou komponentu, je jim doslova injektována od kontejneru (odtud dependency injection). IoC zvyšuje modularitu celého systému a tím jej vytváří snadno rozšiřitelným. Jelikož řeší především reference mezi objekty a jejich vytváření, aplikuje se u objektově orientovaného programování. S Java EE verze 6 vznikla specifikace s názvem Context and Dependency Injection. Ta přináší přístup nazvaný „loose coupling, strong typing“, což si lze vyložit jako snahu o vyvarování se pevných vazeb, a naopak důraz na určení typů parametrů. Tento zdánlivě složitý mechanismus lze prezentovat na jednoduchém příkladu:

Komponenta TextEditor bude nabízet kontrolu pravopisu. Běžný kód by vypadal následovně.

```
public class TextEditor {
  private SpellChecker checker;
  public TextEditor(){
    this.checker = new SpellChecker();
  }
}
```

Tímto byla vytvořena vazba mezi TextEditorem a SpellCheckerem. Znamená však pevnou vazbu s třídou SpellChecker. Pomocí přístupu CDI je možné vytvořit abstrakci tím, že konstruktor bude závislý na třídě SpellChecker, ale nebude v něm probíhat inicializace. Následující příklad umožní vytvořit nejprve závislost, která je dále předána třídě TextEditor.

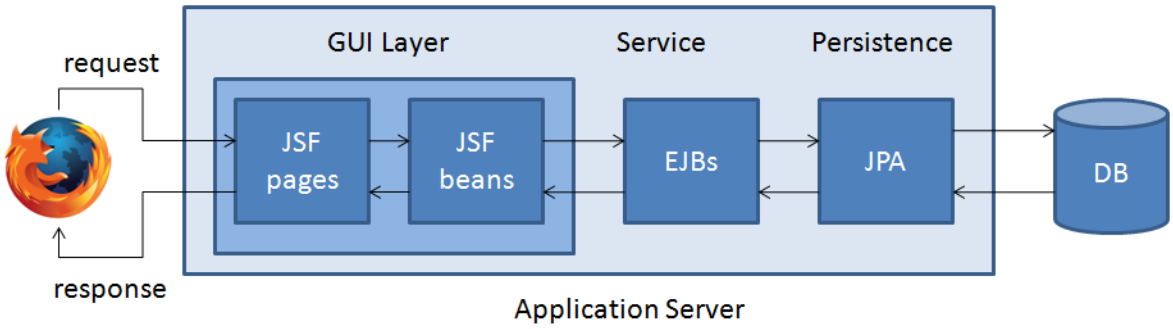
```
public class TextEditor {
    private ISpellChecker checker;
    public TextEditor(ISpellChecker checker){
        this.checker = checker;
    }
}
```

Zmíněný příklad je prvním krokem k tomu, aby se v mnohem komplexnějších případech staral o závislosti kontejner namísto vývojáře.

5.2.2 JavaServer Faces

Jak již bylo zmíněno při výběru vhodné technologie, JavaServer Faces není framework. JavaServer Faces je specifikací jazyka Java pro tvorbu komponentově orientovaných uživatelských rozhraní. Podobně jako organizace IETF [44] pomocí komunitních návrhů *Request for Comments* nebo W3C [45] prostřednictvím svých *Recommendations* zavádějí nové standardy, edice platformy Java se rozšiřují pomocí tzv. *Java Community Process* (dále jen JCP). Kdokoliv se tohoto mechanismu hodnocení a schvalování nových Java Specification Request (dále jen JSR) může zúčastnit. Výsledkem podnětu, který úspěšně prošel JCP je nová specifikace JSR s příslušným číselným označením. Pro tuto práci byla vybrána specifikace JavaServer Faces nesoucí označení JSR 344 [46].

Na grafice Obrázek 10 - Vrstvy aplikace JSF je vidět, že JavaServer Faces pokrývají prezentační a částečně i business vrstvu aplikace. Ta se poté skládá z databáze, persistentní vrstvy přístupu k databázi, služeb tvořící aplikační logiku, tříd se stavem vztažených ke konkrétní zobrazené šabloně a poté ze samotné renderované šablony.



Obrázek 10 - Vrstvy aplikace JSF

Při vzniku JSF bylo nutné vybrat šablonovací systém, pomocí kterého by se dynamicky tvořil obsah HTML stránky. Již v této době existoval jazyk JavaServer Pages (dále jen JSP). JSP byl vhodně vybrán jako defaultní šablonovací systém specifikace JSF. Mimo jiné umožňuje např. práci s kolekcemi, kde může generovat HTML obsah pro každý prvek kolekce.

5.2.3 Wildfly

Dále je potřeba vybrat prostředí, ve kterém aplikace poběží. Společnost Red Hat [47] nabízí moderní open-source software, který je spravován komunitou. Jeden z jejích produktů je nazván JBoss [48]. Je to silný nástroj pro běh aplikací založených na platformě Java. Jelikož jeho možnosti převyšují nároky této práce, byl jako aplikační server vybrán Wildfly [49]. Jde o jednoduchý aplikační server opět od společnosti Red Hat, který vyniká svou malou velikostí a složením z modulů. Většina potřebných knihoven je již ve Wildfly obsažena, ale kdykoliv je možné vytvořit vlastní moduly a možnosti serveru rozšířit.

5.3 Stateless webová aplikace s tenkým klientem

Jak již bylo na úvodu kapitoly řečeno, vývoj nových technologií postupuje rychlým tempem. Během vývoje uživatelského prostředí se dostal do popředí zájmu programovací jazyk Javascript nebo spíše knihovny a frameworky založené na Javascriptu.

5.3.1 React

Vývoj Reactu započal v roce 2014 – současně s touto prací. Jde o open-source knihovnu pro tvorbu uživatelských rozhraní v jazyce Javascript [50]. Jeho vývoj je spravován společností Facebook a individuálními vývojáři.

React dovoluje vytvářet jednoduché i náročné webové aplikace. Tou nejvyšší výhodou oproti jiným řešením je schopnost nepřekreslovat celý zobrazený dokument při změně dat v čase, ale vyřešit tento problém přívětivěji pro uživatele. Pokud vývojář použil atomické a uzavřené komponenty, bude při změně dat obnovena pouze komponenta spravující pozměněná data. To v praxi znamená překreslení pouze minimální části z celého obsahu webové stránky. Právě díky přístupu Reactu k promítání změn uživateli a maximální znovupoužitelnosti komponent se aktuálně těší velké oblibě. To dokazuje velikost komunity na serveru GitHub [51].

Kód Reactu je možné psát v běžném Javascriptu, ale má vlastní syntaxi nazývanou JSX. Při použití JSX je nutné přeložit jazyk pomocí dalších nástrojů jako Babel a tzv. *module bundler*. Ty se starají nejen o přeložení jazyka JSX, ale i o generování výsledného skriptu s koncovkou js, jeho minifikaci a případně tzv. uglifikaci – proces znečitelnění. Pro změnu obsahu uvnitř HTML dokumentu využívá React (resp. Javascript) Document Object Model (zkr. DOM). Ten slouží jako rozhraní umožňující skriptům dynamicky přistupovat a aktualizovat obsah, strukturu a styly dokumentu. React pracuje s tzv. virtuálním DOM. Izoluje změny mezi starým a novým virtuálním DOM a poté aktualizuje reálný DOM na konkrétním místě a pouze nezbytnými změnami. Proces aktualizace probíhá následovně:

- Signál notifikující změnu dat aplikace
- Aktualizace virtuálního DOM
- Výběr změn mezi starým a novým virtuálním DOM
- Aktualizace reálného DOM pouze nezbytnými změnami

Oproti předchozímu řešení s JavaServer Faces nabízí knihovna React přesunutí části aplikační logiky a dalších výpočtů před samotným renderováním na stranu klienta. O přípravu dat, sestavení šablony webového dokumentu a vykreslení uživatelského rozhraní se tedy nestará serverová část ale prohlížeč. Společně s částečným překreslováním pomocí virtuálního DOM nabízí lepší uživatelský prožitek než JSF.

Původní koncept webové aplikace vycházel z výběru nejlepších technologií. I nadále vyhovuje účelům práce platforma Java. Jazyk JSP pro prezentační vrstvu ani specifikace JavaServer Faces již nejsou potřeba. Je potřeba nalézt technologii pro vzájemnou komunikaci mezi klientskou a serverovou aplikací. Kniha Beginning in Java EE 7 [43] popisuje možnosti vzájemné komunikace mezi webovými aplikacemi.

Tou nejoblíbenější je Representational State Transfer [52] (zkr. REST). Aplikace založené na této architektuře jsou založené na velmi robustním, transportním protokolu HTTP a URI. To znamená, že každá URL adresa odkazující na rozhraní založeném na REST API je reprezentována objektem. S tímto objektem je možné komunikovat za pomocí GET, DELETE, POST nebo PUT požadavků. Pro tyto účely slouží specifikace nazvaná Java API for RESTful Web Services. Jelikož serverová část bude sloužit pouze předávání dat, stává se aplikace bezestavová. Tzv. *stateless aplikace* se vyznačují tím, že každý HTTP požadavek se vykonává v naprosté izolaci.

K vývoji javascriptové aplikace byl vybrán manažer balíčků NPM [53]. Ten je součástí běhového prostředí Node.js [54] založeném na enginu V8. Programy tvořící klientskou aplikaci musí být přeloženy, spojeny a zminifikovány. Mezi nejpoužívanější bundlery patří Grunt, Gulp a Webpack. Vzhledem k možnostem nastavení a početné komunitě na serveru GitHub byl vybrán Webpack [55].

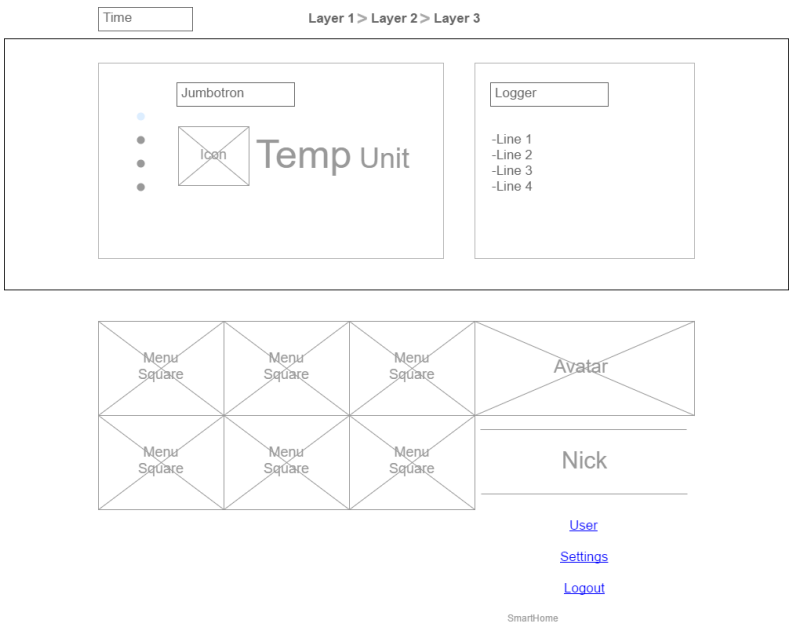
Hlavním cílem Webpacku je tzv. code splitting. Jedná se o proces neudržování jednoho velkého souboru, ale rozdělení kódu na několik menších částí s tím, že jsou volány na vyžádání. Navíc pokud některé dva z modulů Reactu obsahují stejnou závislost, je zahrnuta pouze jednou. Cílem je udržet počáteční čas načítání co nejnižší. Nastavení je zajištěno konfiguračním souborem webpack.config.js.

6 Implementace

Popis funkčního řešení obsahuje provedené kroky pro vytvoření grafického rozhraní. Na grafickou část navazuje vlastní řešení práv a omezení přístupu uživatele. To je příhodně zaměřené na obsluhu klientské aplikace; podporuje generování opakujících se částí uživatelského rozhraní, čímž minimalizuje úpravy pro řešení konkrétního zákazníka. Tomu jsou věnované zvláštní kapitoly. Implementaci ukončuje testování a integrace do akademického projektu Hausy. Ten představuje vlastní řešení akčních prvků, infrastrukturu jejich propojením a na nejvyšší vrstvě REST rozhraní pro komunikaci právě s produktem této práce. Ten je složen z klientské aplikace [4], bezestavové aplikace představující aplikační vrstvu [5] a konfigurace aplikačního serveru Wildfly [6]. To vše je přístupné na přiloženém CD.

6.1 Návrh grafického prostředí

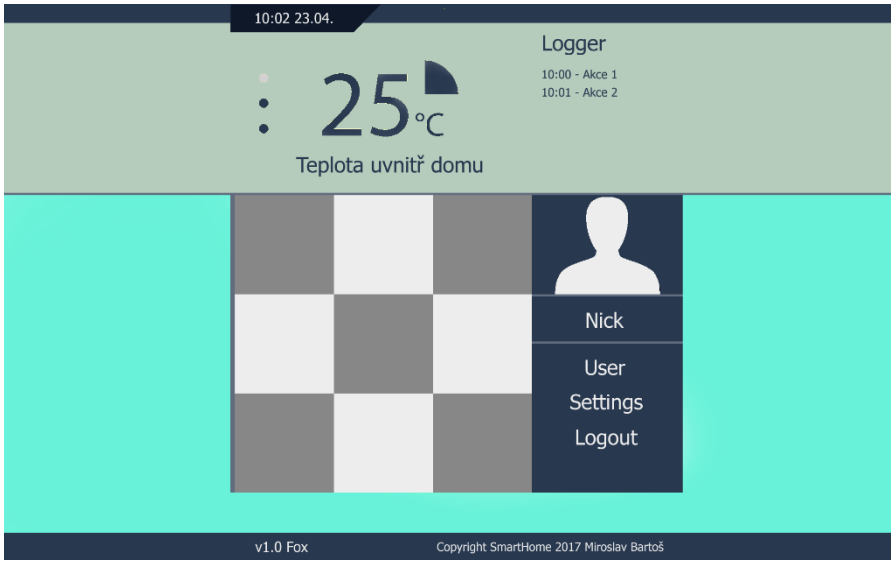
Návrh grafického rozhraní probíhal ve třech hlavních fázích podle kapitoly 4. Na začátku každé z nich byly stanoveny cíle a výsledek sloužil jako předloha pro pokračování v projektu. Během první fáze byl vytvořen drátový model, který měl přiblížit podobu uvítací obrazovky. Cílem bylo vytvořit předlohu takového grafického rozhraní, které bude zobrazitelné na nejmenších dostupných smartphonech i desktopových počítačích. Uživatel bude mít vše přehledně na jednom místě a již tento návrh ponese znaky znovupoužitelnosti.



Obrázek 11 - Drátový model uživatelského rozhraní

Na základě drátového modelu byl vytvořen grafický návrh s konkrétnější podobou aplikace. Tato fáze se opírala o výsledky výzkumu v kapitole 2. Barvy byly vybrány s ohledem na účel této aplikace. Cílem bylo navodit příjemný dojem bez agresivních, vyčnívajících prvků. Pro sladění několika použitých barev byla vybrána paleta ze serveru Colour lovers [56]. Dům vybavený inteligentními prvky má jasnou cílovou skupinu. Jsou to zajištění majitelé, kteří již používají chytrá zařízení. Jsou přípustní změnám, učení se nových věcí a ocení komfort nebo úspory s nimi spojené. Tato cílová skupina vrcholí mezi 30 a 35 roky [57].

Vybraná paleta barev obsahuje uklidňující modrozelenou barvu, která je kombinována s neutrální šedou až bílou. Pro kontrast proti barvě písma a zachování celistvosti byla vybrána tmavá modrá. Tato kombinace sklidila v této věkové skupině úspěch. Dále byl potvrzen návrh drátového modelu složený ze dvou horizontálně rozdělených prostorů. Ty se oba vertikálně dělí v poměru blížícím se zlatému řezu, čímž vytvářejí člověku nejpříjemnější poměr 4 oblastí. Ostré hrany na sebe dobře navazují i při přeuspořádání bloků pro mobilní zařízení.



Obrázek 12 - Grafický návrh uživatelského rozhraní

Finální grafické uživatelské rozhraní navazuje na grafickou předlohu. Nejvýraznější změnou je použitý font a obohacení textu o ikony. V kapitole 2.2.2 byly vybrány dva fonty. Jsou jimi patkové písmo Gentium pro zvýraznění např. nadpisů a oblíbený Roboto pro běžný text. Pro snadnější orientaci byly text a navigační dlaždice doplněny o ikony ze serveru FA [58].

Jak již bylo naznačeno při popisu grafického návrhu, uvítací obrazovka se dělí na 4 hlavní bloky. Přehledné menu a přechod na nastavení vycházejí z potřeb většiny grafických uživatelských rozhraní; musí být ihned dostupné a intuitivně ovladatelné. Majitelé inteligentní elektroinstalace mají k dispozici přehled posledních akcí. Ty mohou být vykonávány dospělými obyvateli domu i dětmi. Především kvůli dětem a autonomnímu chování domu uživatelé tento blok ocení. Konkurenční řešení společnosti Jablotron nabízejí takovou funkci pouze pro mobilní platformu Android, a to v podobě vyskakujících notifikací.

Tím zásadním blokem, který má upoutat pozornost, je volitelný přehled nejdůležitějších akcí. Uživatel má možnost vybrat si z jednoho druhu komponent (Info) pouze ty nejdůležitější a jejich hodnotu odečítat z panelu na dashboardu (úvodní obrazovce). Tento panel je ve skutečnosti jednoduchou komponentou přijímající vybrané hodnoty z REST rozhraní a sestavující komponenty ve formátu použitelném pro přehled. Ty jsou předány komponentě k vykreslení a doplnění o ovládací prvky přechodu včetně listování přetažením prstu. Maximální využití najde na nástěnném, centrálním displeji uvnitř domu. Volba čtyř prostorů přináší přehlednost a jednoduchost na desktopová zařízení. Dále zajišťuje pohodlné ovládání na tabletu. Ovládací tlačítka profilu mají stejnou šířku jako šedý oddělovač. Díky tomu není potřeba přehmatávat a uživatel dokáže uskutečnit volbu palcem pravé ruky. Podobně je tomu u menu na opačné straně. Jednotlivé obrazovky se generují samy, ale na přání je možné změnit jejich pořadí a tím nastavit nejpoužívanější obrazovky na dosah palce.

Obrázek 13 - Uživatelské rozhraní – uvítací obrazovka

Přehled důležitých informací musí být dobře viditelný i na nástěnné dotykové obrazovce. Výborného ovládání dosahuje na menších mobilních zařízení s obrazovkou do 7 palců. Celkové rozložení se změní, jednotlivé bloky se na smartphonech uspořádají pod sebe a jelikož odkazy zabírají celou plochu svého pozadí, jsou lehce stisknutelné i na rozměrnějších zařízeních, které uživatel drží v jedné ruce. Jelikož se počítá se zobrazením v režimu celé obrazovky, bylo uživatelské rozhraní doplněno o horní lištu s časem a navigační lištou zpět na úvodní obrazovku. Nevyužitý prostor by měl v budoucnu umožnit zobrazení nových akcí z panelu historie.

6.2 Řízení přístupu

React je knihovna pro psaní single-page aplikací. Pro přecházení mezi více obrazovkami byl využit balíček react-router (dále jen Router). Ten je schopný obsluhovat routování podle zadané URL adresy. Protože rozhoduje o obsahu stránky, umísťuje se do metody pro renderování uvnitř komponenty. Mimo routování je React schopný kontrolovat přístup, čehož bylo využito i v při vypracování praktické části.

```
render: function() {
  return (
    <div>
      <Router history={hashHistory}>
        <Route path="lighting" component={Lighting}
onEnter={this.checkPermission} />
      </Router>
    </div>
  )
}
```

Řízení přístupu je kontrolováno na nejvyšší úrovni aplikace. Ve chvíli, kdy je od uživatele zachycen přesun na jinou obrazovku, vyvolá Router metodu *checkPermission*. Ta vyvolá požadavek na server. Uvnitř něj odesílá unikátní token uživatele a část URL, která odpovídá požadované obrazovce uživatele.

Podle odpovědi reaguje Router přesměrováním uživatele na požadovaný obsah nebo zákazem přístupu a přesměrováním zpět.

```
let uuid = window.sessionStorage.getItem("token");
let request = {
  uuid: uuid,
  path: location.location.pathname
};

$.ajax({
  url: CONSTANTS.URL_HTTPS_AUTORIZATION,
  type: "POST",
  crossDomain: true,
  data: JSON.stringify(request),
  contentType: "application/json; charset=utf-8",
  dataType: "json",
  success: function(data){}
  error: function(data) {}
});
```

Server obslouží požadavek a na back-endové aplikaci je vyvolána metoda *post* uvnitř třídy *Authorize.java*. V ní jsou nejdříve injektovány závislosti na objekty pro přístup databázi – tzv. Data Access Object.

```
@Inject
UserDao userDao;
@Inject
ScreenDao screenDao;
@Inject
UserScreenDao userScreenDao;
```

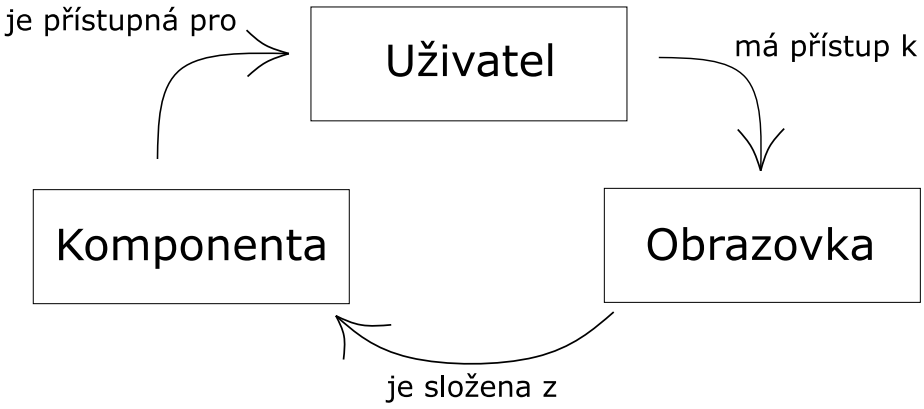
V první řadě je nutné zkontrolovat platný token. K tomu slouží DAO pro tabulku uživatelů. Na ní je možné volat metodu *findByToken* pro přečtení údajů z databáze. Parametrem metody je token přijatý požadavkem z klienta. Podobným přístupem je nutné ověřit, zda má daný uživatel přístup k požadované obrazovce pomocí metody findByUrl s parametrem obsahujícím název obrazovky.

```
user = userDao.findByToken(input.uuid);
screen = screenDao.findByUrl(input.path);
```

Pokud skončí proces ověření tokenu a přístupu k obrazovce úspěchem, je naplněn objekt Authorizator kladnou odpovědí a odeslán na klienta. Tam je odpověď vyhodnocena a uživatel je přesměrován. Pokud ověření na serveru selže, odesílá se na klienta chybová zpráva, která je zobrazena uživateli pomocí dialogového okna.

6.3 Dynamické načítání řídících obrazovek

Pro zajištění korektního zobrazení na jakékoliv obrazovce a rychlého vývoje bylo zapotřebí vymyslet jednoduchý model, který by umožnil spravovat obrazovky aplikace, akční prvky a zároveň by řešil přístup uživatele. Proto byl vyvinut přístup uživatel – obrazovka – komponenta zobrazený na obrázku Obrázek 14 - Řízení přístupu a skládání obsahu obrazovek. Ten tvoří cyklickou závislost mezi zmíněnými entitami. Každý uživatel je propojený se všemi obrazovkami. V rámci tohoto vztahu je možné řešit i kontrolu přístupu na tuto obrazovku zvlášť pro každého uživatele. Obrazovka je složena z akčních prvků, které vždy náleží právě jedné obrazovce. A nakonec existuje vztah mezi každým akčním prvkem a uživatelem, čímž je zároveň vyřešena kontrola přístupu na úrovni komponent z kapitoly 4.6.



Obrázek 14 - Řízení přístupu a skládání obsahu obrazovek

Takový přístup dává vývojáři možnost vybrat si, z jakých logických celků bude aplikace složena. Mezi typická rozdělení patří členění účelu (vytápění, osvětlení atd.) nebo vytváření logických celků simulujících místnosti.

Další výhodou je generování každé obrazovky z jediné komponenty Reactu nazvané Screen. Ta dokáže pomocí předaného parametru rozpoznat účel, vyslat požadavek o akční prvky spjaté pouze s touto obrazovkou a dynamicky vykreslit jakýkoliv typ akčního prvku. Vývojář má tedy minimum práce s případným rozšířením na míru zákazníka.

```
var Heating = React.createClass({
  render: function() {
    return (
      <Screen
        path="/heating"
        sName="Vytápění"
        icon="fa fa-thermometer-half" />
    )
  }
});
```

6.4 Generování komponent pro ovládání

Uživatelské rozhraní je vygenerováno podle uživateli přístupných komponent. Při inicializaci obrazovky se ukládá do stavu obrazovky její název a inicializuje se seznam komponent pro danou obrazovku.

```
getInitialState() {
  return {
    screenPath: '/heating',
    components: []
  };
}
```

Ještě před samotným připojením komponenty představující obrazovku do hierarchie načtené aplikace je vyvolán požadavek na vrácení všech akčních prvků. Tyto prvky souvisí s ovládáním na dané obrazovce.

```
$.ajax({
  url: CONSTANTS.URL_HTTPS_GET_COMPONENTS,
  type: "POST",
  crossDomain: true,
  data: JSON.stringify(request),
  contentType:"application/json; charset=utf-8",
  dataType: "json",
  success: function(data){
    this.setState({components: data});
  }.bind(this),
});
```

Po přijetí požadavku na seznam komponent je injektována reference na objekt přístupu k datům tabulky komponent uvnitř databáze. Podle přijatého parametru, který obsahuje název obrazovky jsou z databáze vyselektovány pouze akční prvky vztažené k obrazovce. Tento seznam je odeslán jako odpověď zpět na klienta.

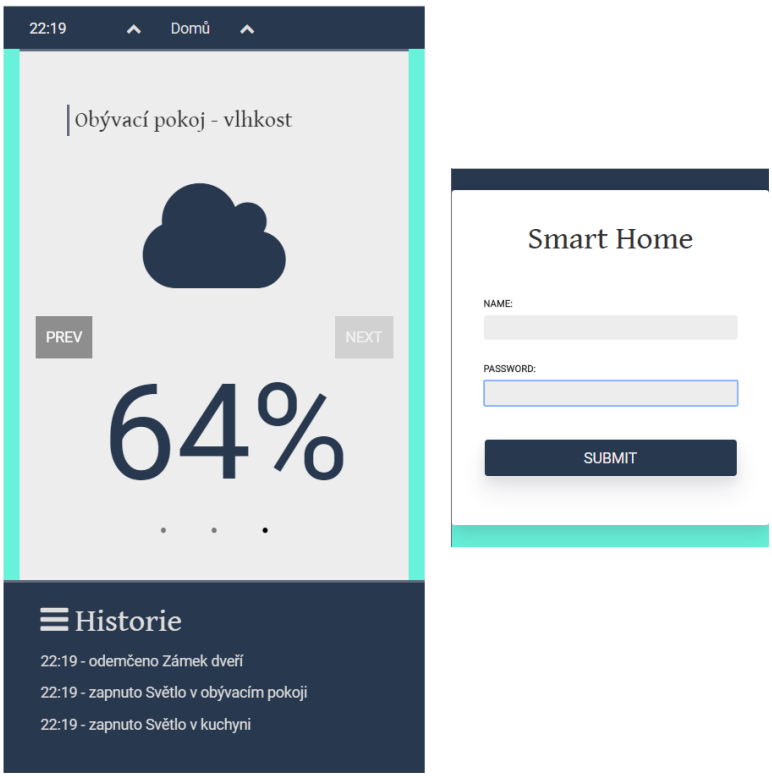
```
var squares = this.state.components.map(function(item, i){
  if (item.cType === "btn") {
    return (
      <BtnSquare
        key={i}
        id={item.id}
        cName={item.cName}
        value={item.value}
        description={item.description} />
    )
  } else if (item.cType === "sldr") {
    return (
      <SldrSquare
        key={i}
        id={item.id}
        cName={item.cName}
        value={item.value}
        description={item.description} />
    )
  }
);
```

Vrácený seznam je uložen do stavu komponenty, což automaticky vyvolá nové vykreslení metody render. Při této akci, ještě před samotným renderováním, je pomocí javascriptové funkce map uložena pro prvek z odpovědi ze serveru tzv. dlaždice. Dlaždice je atomická komponenta, která plní jediný účel – zobrazit popisek a akční prvek. Akce dlaždice se liší podle jejího typu. Pro potřeby uživatelského rozhraní inteligentních domů byly sestaveny 4 akční prvky. Jsou jimi tlačítko, posuvník, spínač a info panel. Následující ukázka kódu ukazuje metodu render komponenty představující tlačítko.

```
render() {
  return (
    <div className="col-sm-3 col-md-3 col-lg-3 btnSquare">
      <p>{this.props.description}</p>
      <div className="classicButton">
      <Button
        bsStyle="primary"
        onClick={this.handleClick}>Proveď</Button>
      </div>
    </div>
  )
}
```

6.5 Testování

Aplikace byla testována na zařízeních HP 4530s a Lenovo Flex 2 s dotykovou obrazovkou. Rozvržení prvků úvodní obrazovky bylo testováno na mobilním zařízení Huawei Nexus 6P. K testování byly vybrány nejpoužívanější prohlížeče. Jmenovitě jde o Internet Explorer verze 11, Mozilla Firefox 49 a Google Chrome 57 za pomocí pluginu Web Developer. Ve všech prohlížečích proběhlo testování na monitoru s rozlišením 1920x1080 pixelů. Aplikace obstála i při zmenšení šířky pod 300 pixelů, čímž vyhovuje většině smartphonů, všem tabletům a desktopovým počítačům. Mobilní zařízení Huawei Nexus 6P disponuje rozlišením WQHD s rozlišením 2560x1440 pixelů.



Obrázek 15 - Testování aplikace pro šířku 300 pixelů

Funkčnost rozhraní bezestavové aplikace a správné výsledky požadavků byly ověřeny nástrojem Postman. Ten umožňuje zasílání veškerých http požadavků včetně sestavení vlastní hlavičky ve formátu json a kontrolu přijatých požadavků v témže formátu.

6.6 Integrace do řešení Hausy

Fyzická i střední vrstva akademického projektu Hausy se neustále vyvíjí. Kvůli tomu nebylo možné mít v jeden okamžik všechny potřebné informace pro vytvoření kompletního systému. Jde o hlavní důvod, proč se nedržet zažitých a výborně popsaných rigorózních metodik. Ty mají pevné pilíře a jsou základním stavebním kamenem stále ještě většího podílu nových projektů. Vývoj jedné ze tří podstatných částí systému ve chvíli, kdy se o těch zbylých nedá říci, že se alespoň v základech nezmění, nutně potřebuje obsahovat prvky pružnosti. Jak už jejich název napovídá, právě díky své pružnosti a údernosti jsou agilní metodiky ideální pro tento projekt.

7 Závěr

V této práci bylo vytvořeno komfortní uživatelské rozhraní pro ovládání inteligentních domů. Rešerše se částečně týkala aktuálních trendů v oblasti inteligentní elektroinstalace a směru jejího vývoje. Zbylá část rešerše pokrývala kompletní náplň práce designéra. Bylo zjištěno, že nejde jen o grafickou část aplikace, ale i zaujetí z prvního pohledu, funkční a intuitivní ovládání a dobrý dojem po dokončení požadované akce. Volbou vhodné barvy, písma, rozvržení layoutu a použitím technologie aktualizující minimum obsahu při změně dat byl splněn požadavek na aplikaci UX. Nezbytnou vlastností se stává responzivita, díky níž se obsah přizpůsobí obrazovce uživatele.

Na základě zjištěných poznatků a ponaučení z dostupných řešení byla v rámci praktické části vypracována webová aplikace technologií JavaServer Faces. Během jejího vývoje se však prosadily řešení na základě Javascriptu. Původní koncept byl tedy přehodnocen a prezentační vrstvu nahradil tenký klient. Tím vznikl nový koncept zahrnující bezestavovou aplikaci na platformě Java komunikující s klientskou aplikací spouštěnou na straně klienta. Pro její vypracování byla vybrána nová a komunitně velmi oblíbená technologie – knihovna React. Tím bylo docíleno částečného přesunutí výpočtů na stranu klienta a splnění technologických cílů. Použitím frameworku Bootstrap bylo dosaženo responzivní aplikace. Jedno řešení je tak použitelné na smartphonu, tabletu, i desktopu.

Stěžejní část klientské aplikace tvoří úvodní obrazovka rozdělená na 4 bloky pro okamžitý přehled a přístup do dalších sekcí. Odtud má uživatel přístup na obrazovky pro ovládání inteligentního domu. Ty jsou vytvářeny z jediné komponenty. Každá obrazovka pro řízení obsahuje akční prvky. Ty se dělí na 4 typy, přičemž každý z nich se opět vytváří automatizovaně z jediné komponenty. Výsledkem jsou maximálně znovupoužitelné části aplikace a co nejjednodušší rozšíření. Přidání obrazovky zvládne kodér upravením routování. Nový typ akčního prvku znamená přidání atomické komponenty s popisem a ovládacím prvkem.

Tím nejpřínosnějším je však generování kompletního řešení pro zákazníka jedním SQL skriptem, který zapíše uživatele, obrazovky a těm zavede záznamy do tabulky komponent. Jakémukoliv návrhu inteligentního domu se přizpůsobí přihlašování, řízení přístupu i renderováním libovolného počtu komponent, čímž byl v praxi naplněn cíl pro vlastní řešení řízení přístupu s ohledem na snadné přizpůsobení požadavkům zákazníka.

Součástí implementace již není rozšíření o uživatelské definování pravidel. Uživatel by tak dostal možnost seskupovat vybrané akce do jediného tlačítka, čímž by se usnadnily rutinní činnosti. Na instalaci pluginu např. pro fuzzy vytápění a tím úsporu energií také nebyl během vypracování této práce prostor.

8 Seznam použitých zdrojů

[1] VALEŠ, Miroslav. *Inteligentní dům*. 2. vyd. Brno: ERA, 2008, viii, 123 s. 21. století. ISBN 978-80-7366-137-3.

[2] MERZ, Hermann, Thomas HANSEMANN a Christof HÜBNER. *Automatizované systémy budov: sdělovací systémy KNX/EIB, LON a BACnet*. 1. vyd. Praha: Grada, 2008, 261 s. ISBN 978-80-247-2367-9.

[3] STANÍČEK, Petr. *Dobrý designér to všechno ví!* [online]. I. vydání. Kamenné Žehrovice: vydáno vlastním nákladem autora, 2016 [cit. 2016-12-30]. ISBN 978-80-260-9427-2.

[4] ALLEN, Jesmond. a James. CHUDLEY. *Smashing UX design: foundations for designing online user experiences*. Chichester, West Sussex, UK: John Wiley, 2012. Smashing magazine book series. ISBN 04-706-6685-4.

[5] GAGE, John D. Color and Culture: PRACTICE AND MEANING FROM ANTIQUITY TO ABSTRACTION. Boston, 1993. ISBN 978-0-82122-043-6.

[6] WOLFGANG VON GOETHE, Johann. *Theory of Colours*. Germany, 1840. ISBN 0-262-57021-1.

[7] GOLDSTEIN, KURT. *SOME EXPERIMENTAL OBSERVATIONS CONCERNING THE INFLUENCE OF COLORS ON THE FUNCTION OF THE ORGANISM*. 1942. DOI: 10.1097/00002060-194206000-00002. ISBN 10.1097/00002060-194206000-00002. Dostupné také z: http://content.wkhealth.com/linkback/openurl?sid=WKPTLP:landingpage

[8] NAKSHIAN, Jacob S. *The Effects of Red and Green Surroundings on Behavior*. 2010, 1962. DOI: 10.1080/00221309.1964.9920584. ISBN 10.1080/00221309.1964.9920584. Dostupné také z: http://www.tandfonline.com/doi/abs/10.1080/00221309.1964.9920584

[9] ELLIOT, Andrew J. *Color and psychological functioning: a review of theoretical and empirical work* [online]. 2015 [cit. 2017-04-09]. DOI: 10.3389/fpsyg.2015.00368. ISBN 10.3389/fpsyg.2015.00368. Dostupné z: http://journal.frontiersin.org/article/10.3389/fpsyg.2015.00368/abstract

[10] CULLEN, Kristin. Design Elements, Typography Fundamentals: A Graphic Style Manual for Understanding How Typography Affects Design. Rockport Publishers, 2012. ISBN 1592537677.

[11] AMBROSE, Gavin a Paul HARRIS. *The Fundamentals of Typography: Second Edition*. AVA Publishing, 2011. ISBN 978-2940411764.

[12] SHARKIE, Craig a Andrew FISHER. *Responzivní webdesign: okamžitě*. Brno: Computer Press, 2015. ISBN 978-80-251-4384-1.

[13] Google Web Fonts library: Making the web more beautiful, fast, and open through great typography. [online]. 2017 [cit. 2017-04-09]. Dostupné z: https://fonts.google.com/

[14] Gentium: GWF. *Google Web Fonts library: Making the web more beautiful, fast, and open through great typography.* [online]. 2017 [cit. 2017-04-09]. Dostupné z: https:// https://fonts.google.com/specimen/Gentium+Basic

[15] Roboto: GWT. *Google Web Fonts library: Making the web more beautiful, fast, and open through great typography.* [online]. 2017 [cit. 2017-04-09]. Dostupné z: https://fonts.google.com/specimen/Roboto?selection.family=Roboto

[16] 6 Reasons for Employing Component-based UI Development. *Tandem Seven: Envision, Design & Build* [online]. Boston: Damien Scott, Mundi Morgado, 2017 [cit. 2017-04-09]. Dostupné z: http://www.tandemseven.com

[17] VALLECILLOS, Jesús, Javier CRIADO, Nicolás PADILLA a Luis IRIBARNE. A Component-based User Interface Approach for Smart TV. *Proceedings of the 9th International Conference on Software Engineering and Applications*. SCITEPRESS - Science and and Technology Publications, 2014-8-29, 455-463. DOI: 10.5220/0004999304550463. ISBN 978-989-758-036-9. Dostupné také z: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0004999304550463

[18] Internet Trends Report. *KBCP: Kleiner Perkins Caufield & Byers* [online]. 101 South Park Street San Francisco, 2016 [cit. 2017-01-10]. Dostupné z: http://www.kpcb.com/internet-trends

[19] Mobile and tablet internet usage exceeds desktop for first time worldwide. *StatCounter: Global Stats* [online]. [cit. 2017-04-26]. Dostupné z: http://gs.statcounter.com/press/mobile-and-tablet-internet-usage-exceeds-desktop-for-first-time-worldwide

[20] GASSTON, Peter. *Moderní web*. Brno: Computer Press, 2015. ISBN 978-80-251-4345-2.

[21] MARCOTTE, Ethan a [FOREWORD BY JEREMY KEITH]. *Responsive web design*. New York: A Book Apart, 2011. ISBN 978-098-4442-577.

[22] BUCHALCEVOVÁ, Alena. Metodiky vývoje a údržby informačních systémů: kategorizace, agilní metodiky, vzory pro návrh metodiky. Praha: Grada, 2005. ISBN 80-247-1075-7.

[23] *HTML 5 Boilerplate* [online]. [cit. 2017-04-24]. Dostupné z: https://html5boilerplate.com/

[24] *Twitter Bootstrap* [online]. 2017 [cit. 2017-04-24]. Dostupné z: http://getbootstrap.com/

[25] *The New Methodology: From Nothing, to Monumental, to Agile* [online]. 2005 [cit. 2017-04-13]. Dostupné z: https://www.martinfowler.com/articles/newMethodology.html

[26] TURK, D., R. FRANCE a B. RUMPE. Limitations of Agile Software Processes: Third International Conference on Extreme Programming and Flexible Processes in Software Engineering [online]. Alghero, Italy, 43 - 46 [cit. 2017-04-14]. DOI: 1409.6600. Dostupné z: https://arxiv.org/ftp/arxiv/papers/1409/1409.6600.pdf

[27] ARLOW, Jim a Ila NEUSTADT. *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky*. 2., aktualiz. a dopl. vyd. Brno: Computer Press, 2007. ISBN 978-80-251-1503-9.

[28] BISWAS, Prosunjit, Ravi SANDHU a Ram KRISHNAN. *Uni-ARBAC: A Unified Administrative Model for Role-Based Access Control* [online]. , 218 [cit. 2017-04-14]. DOI: 10.1007/978-3-319-45871-7_14. Dostupné z: http://link.springer.com/10.1007/978-3-319-45871-7_14

[29] PARREND, Pierre a Stéphane FRÉNOT. *Component-Based Access Control: Secure Software Composition through Static Analysis* [online]. [cit. 2017-04-14]. DOI: 10.1007/978-3-540-78789-1_5. ISBN 10.1007/978-3-540-78789-1_5. Dostupné z: http://link.springer.com/10.1007/978-3-540-78789-1_5

[30] RICHARD HARPER (ED.). *Inside the smart home*. London: Springer, 2003. ISBN 978-185-2338-541.

[31] *Loxone SMART HOME: Domácnost s autopilotem* [online]. Loxone Electronics, 2017 [cit. 2017-01-29]. Dostupné z: https://www.loxone.com/cscz/

[32] *OpenHAB: Empowering the smart home* [online]. Imprint, 2017 [cit. 2017-01-29]. Dostupné z: http://www.openhab.org/

[33] *Jablotron: Creating alarms* [online]. Sherwood, 2017 [cit. 2017-01-30]. Dostupné z: https://www.jablotron.com/cz/

[34] *ENIKA.CZ: Business and Technology* [online]. 2014 [cit. 2017-01-30]. Dostupné z: http://www.enika.cz/

[35] CHLAPEK, Dušan, Václav ŘEPA a Iva STANOVSKÁ. *Analýza a návrh informačních systémů*. Praha: Oeconomica, 2011. ISBN 978-80-245-1782-7.

[36] KAPPEL, Gerti. *Web engineering: the discipline of systematic development of web applications*. Hoboken, NJ: John Wiley, 2006, xvii, 366 p. ISBN 978-047-0015-544.

[37] RIEHLE, Dirk. *Framework Design: A Role Modeling Approach* [online]. Zürich, 2000 [cit. 2016-02-05]. Dostupné z: http://dirkriehle.com/computer-science/research/dissertation/. Disertační. ETH Zürich. Vedoucí práce Prof. Dr. Thomas R. Gross, ETH Zürich.

[38] SAEHWA, Kim a Choi KIBONG. *Comparing Model, View, and Controller Architecture and Logical User Interface Model* [online]. 2016 [cit. 2017-04-15]. DOI: 10.1166/asl.2016.7865. Dostupné z: https://doi.org/10.1166/asl.2016.7865

[39] Framework Usage Statistics: Statistics for websites using Framework technologies. *BuiltWith: Find out what websites are Built With* [online]. 2007 [cit. 2016-02-06]. Dostupné z: http://trends.builtwith.com/framework

[40] Web MVC framework: Introduction to Spring Web MVC framework. *Spring by Pivotal: Let's build a better Enterprise.* [online]. 2017 [cit. 2017-02-04]. Dostupné z: https://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html

[41] JSF is not what you've been told anymore: JSF and Primefaces. *Primefaces Blog* [online]. 2017 [cit. 2017-02-04]. Dostupné z: http://blog.primefaces.org/?p=3035

[42] BootsFaces: A powerful and lightweight JSF framework based on Bootstrap 3 and jQuery UI [online]. 2017 [cit. 2017-04-15]. Dostupné z: www.bootsfaces.net

[43] GONCALVES, Antonio. *Beginning Java EE 7*. Berkeley, CA: Apress, 2013. ISBN 978-143-0246-275.

[44] The Internet Engineering Task Force: Make the Internet work better [online]. [cit. 2017-04-15]. Dostupné z: www.ietf.org

[45] *W3C: The World Wide Web Consortium* [online]. 2017 [cit. 2017-04-15]. Dostupné z: www.w3.org

[46] JSR 344. JavaServer Faces 2.2: Update to the 2.1 version of the JavaServer Faces specification. https://www.jcp.org/en/jsr/detail?id=344, 2013.

[47] *Red Hat* [online]. 2017 [cit. 2017-04-15]. Dostupné z: www.redhat.com

[48] Red Hat JBoss Middleware: Tooling and middleware for developing enterprise software [online]. 2017 [cit. 2017-04-15]. Dostupné z: www.jboss.org

[49] *Wildfly: Flexible, lightweight, managed application runtime* [online]. 2017 [cit. 2017-04-15]. Dostupné z: www.wildfly.org

[50] *React: A Javascript library for building user interfaces* [online]. 2017 [cit. 2017-04-15]. Dostupné z: www.facebook.github.io/react/

[51] *GitHub: Development platform* [online]. 2017 [cit. 2017-04-15]. Dostupné z: www.github.com

[52] *JSR 339: JAX-RS 2.0:: The Java API for RESTful Web Services* [online]. 2017 [cit. 2017-04-15]. Dostupné z: www.jcp.org/en/jsr/detail?id=339

[53] *NPM: Package manager for JavaScript* [online]. 2017 [cit. 2017-04-15]. Dostupné z: www.npmjs.com

[54] Node.js: JavaScript runtime built on Chrome's V8 JavaScript engine [online]. 2017 [cit. 2017-04-15]. Dostupné z: www.nodejs.org

[55] *Webpack: Module bundler* [online]. 2017 [cit. 2017-04-15]. Dostupné z: www.webpack.github.io

[56] *Colour lovers: Website trends* [online]. 2017 [cit. 2017-04-23]. Dostupné z: http://www.colourlovers.com/web/trends/websites

[57] Smart Home Adoption Split by Age. *Tech Home Builder* [online]. 2015 [cit. 2017-04-27]. Dostupné z: http://techomebuilder.com/emagazine-articles-1/smart-home-adoption-spilt-age

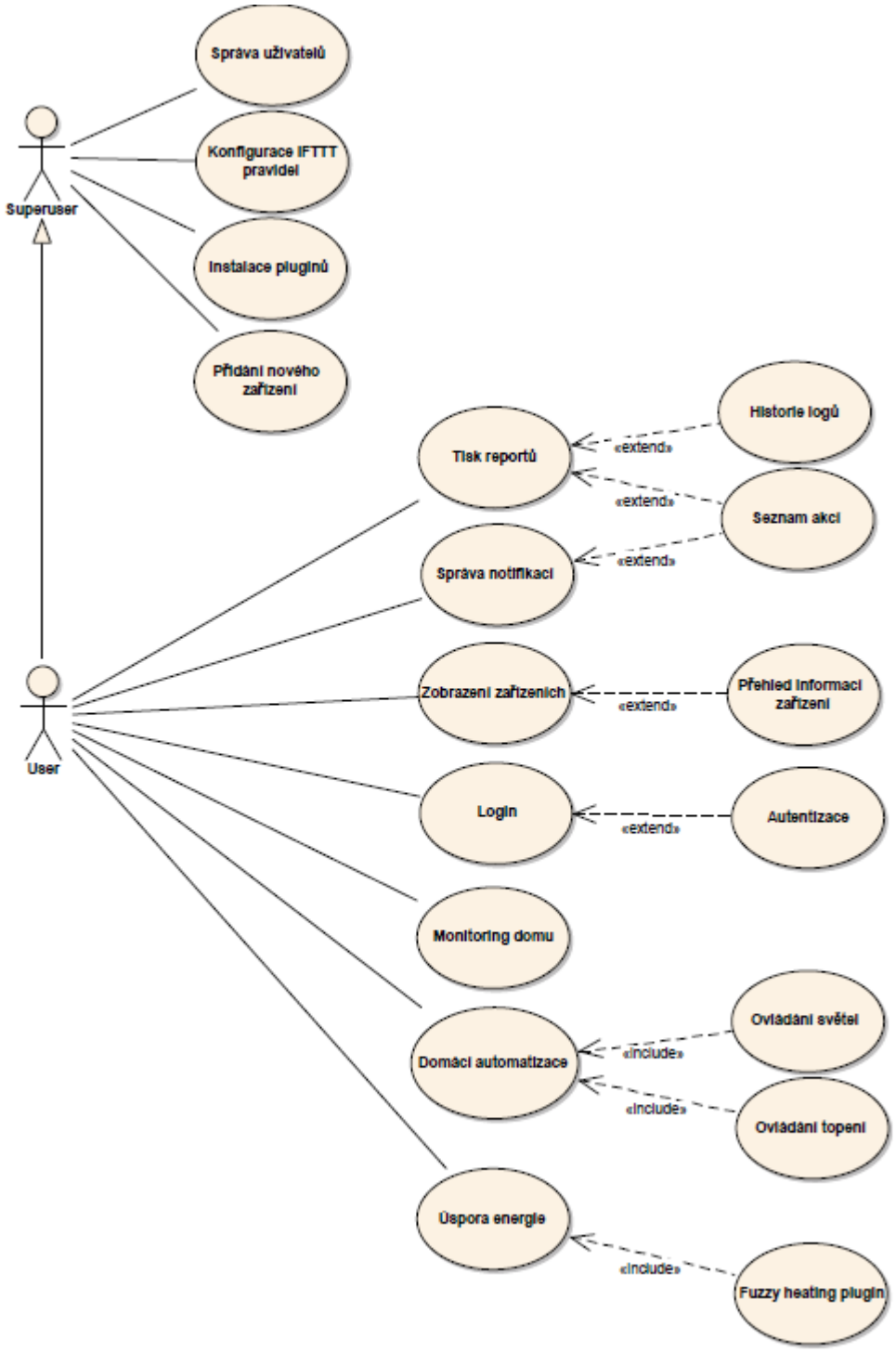
[58] *Font awesome icon* [online]. 2017 [cit. 2017-04-23]. Dostupné z: http://fontawesome.io/

9 Seznam ilustrací

10 Seznam tabulek

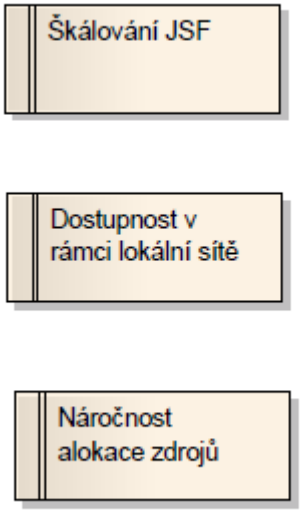
11 Přílohy

[1] Use case diagram



[2] Funkční požadavky

[3] Nefunkční požadavky



[4] CD – zabezpečená klientská aplikace v technologii React

[5] CD - bezestavová aplikace s klientskou i serverovou částí REST rozhraní specifikace JAX-RS

[6] CD - nakonfigurovaný aplikační server Wildfly včetně zabezpečení komunikace pomocí SSL a s vlastním modulem pro přístup do databáze