VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

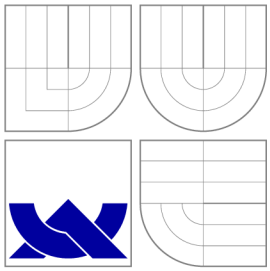
DATABÁZE PRO INTELIGENTNÍ DOMÁCNOST

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

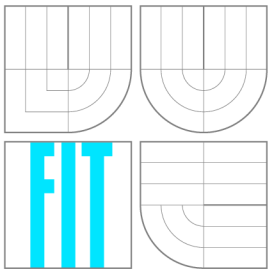
AUTOR PRÁCE
AUTHOR

Bc. PAVEL VAMPOLA

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

DATABÁZE PRO INTELIGENTNÍ DOMÁCNOST

DATABASE FOR THE INTELLIGENT HOME

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PAVEL VAMPOLA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JAN KOŘENEK, Ph.D.

BRNO 2015

Abstrakt

Tato práce se zabývá databází pro systém inteligentní domácnosti, který je vyvíjen na FIT VUT v Brně. Práce popisuje návrh a implementaci databáze inteligentní domácnosti v systému PostgreSQL a zaměřuje se na urychlení dotazů a vkládání do databáze. Navržená databáze je schopna, na procesoru Intel Xeon E5410 s 12GB RAM a jedním diskem, obsloužit přibližně 16000 domácností. Dále práce popisuje serverovou aplikaci, která komunikuje s uživatelskými zařízeními a poskytuje jim data uložená v databázi. Celý systém inteligentní domácnosti byl důkladně otestován reálnými uživateli. Bylo provedeno několika měsíční testování alfa a beta verze. Serverová aplikace je zároveň zanesena do systému kontinuální integrace Jenkins.

Abstract

This thesis deal with database for smart homes, that is developing at FIT VUT in Brno. Thesis describes design and implementation of database for smart home in PostgreSQL and focus on speeding up queries and inserting to database. Designed database, on processor Intel Xeon E5410 with 12GB RAM and one hard drive, is capable to serve about 16000 homes. The thesis describes server application, which comunicate with user devices and provides data stored in database. Whole smart home system was thoroughly tested by real users. Several months of testing was made on alpha and beta versions. Server application is also integrated to system of continual integration Jenkins.

Klíčová slova

internet věcí, inteligentní domácnost, domácí automatizace, databáze

Keywords

internet of things, inteligent home, home automation, database

Citace

Pavel Vampola: Databáze pro inteligentní domácnost, diplomová práce, Brno, FIT VUT v Brně, 2015

Databáze pro inteligentní domácnost

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Jana Kořenka, Ph.D. Uvedl jsem všechny literární prameny publikace, ze kterých jsem čerpal.

.....
Pavel Vampola
26. května 2015

Poděkování

Děkuji vedoucímu práce Ing. Janu Kořenkovi, Ph.D. za odbornou pomoc a konzultace. Dále bych chtěl poděkovat celému týmu projektu IoT na FIT VUT za skvělý kolektiv. Poděkování také patří mé rodině a přátelům, kteří mě při psaní diplomové práce podporovali.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Lidé se odjakživa snaží zvětšovat své pohodlí. Díky této vlastnosti se vyvíjí stále nové vynálezy, které mají za úkol zvyšovat komfort jejich uživatelům. Často se jedná o činnosti, které kdysi člověk musel dělat sám, ale nyní je za nás dělají stroje. Jedná se například o pračky, myčky nádobí, kávovary, apod. V poslední době tento trend zachází ještě o něco dál a vznikají systémy takzvané inteligentní domácnosti, které se snaží pohodlí svých uživatelů ještě o něco zvýšit.

Vize inteligentních domácností je řídit všechny spotřebiče domácnosti v souladu se zájmem uživatele a to nejlépe bez jeho zásahu. Například při příjezdu ke svému domu vaše inteligentní domácnosti automaticky otevře garáž, rozsvítí v ní a v momentě, kdy zaparkujete, vám odemkne a otevře dveře do domácnosti, pustí hudbu, vyhřeje saunu, připraví váš oblíbený čaj nebo kávu, pustí televizi, nebo cokoliv, co si dokážete představit.

Tyto automatizované systémy již nějakou dobu existují, ale jejich inteligence je značně omezená. Uživateli umožňují ovládat své spotřebiče přes mobilní telefon nebo počítač, případně navíc umožňují definovat některá pravidla, kterými se domácnost řídí. Může se jednat o automatické zavírání oken při dešti, zabezpečení objektu při odchodu a podobně. K opravdovým inteligentním domácnostem jim však ještě chybí umění rozhodovat o objektu bez uživatelova zásahu, či definovaných pravidel.

Díky miniaturizaci, snížení spotřeby a ceny čipů i bezdrátových technologií se v posledních několika letech na trhu otevírají nové možnosti pro řešení inteligentních domácností a počet komerčních firem na této části trhu se zvyšuje. Zájem však nezůstal pouze v komerční sféře a i na fakultě informačních technologií v Brně vznikl projekt IoT, který se začal věnovat systému inteligentní domácnosti a začal ho vyvíjet a dále zkoumat. Jednou částí je databáze, které se se tato práce věnuje.

Cílem této práce je navrhnout a vytvořit databázi, která slouží inteligentním domácnostem jako úložiště dat. Databáze je umístěna na serveru a tím pádem jsou data dostupná všem uživatelům, ať se nachází kdekoliv, jediné co musí mít, je připojení k internetu.

Text práce je rozdělen do vícero logických celků. V druhé kapitole je představen koncept internetu věcí a jeho minulost. Navazující pojem inteligentní domácnosti je probírán v kapitole třetí, ve které jsou zároveň uvedeny současná existující řešení inteligentních domácností a jsou popsány jejich vlastnosti. Dále se práce zabývá architekturou a podrobněji popisuje řešení inteligentní domácností vyvíjené na fakultě informačních technologií v rámci projektu IoT. Současným databázových technologiím se věnuje čtvrtá kapitola. Požadavky na databázi inteligentní domácnosti a její návrh jsou popsány v páté kapitole. Funkcionalita serveru a jeho aplikace jsou probírány v šesté kapitole. V této kapitole je blíže popsána serverová aplikace sloužící ke komunikaci s uživatelskými zařízeními, která byla implementována

v rámci této práce. V sedmé kapitole jsou popsány možnosti optimalizace produkčního systému, které mohou vést ke zvýšení výkonu a jsou podrobněji popsány možnosti optimalizace na úrovni databáze. Poslední osmá kapitola obsahuje testování a vyhodnocení dosažených výsledků.

Kapitola 2

Internet věcí

Internet věcí (Internet of things) může být vnímán jako koncept vzájemného propojení velkého množství objektů (věcí) pomocí sítě internet. To zahrnuje všechno od mobilních telefonů, přes ledničky, teploměry, kávovary, žárovky až po téměř cokoli co si představíte.

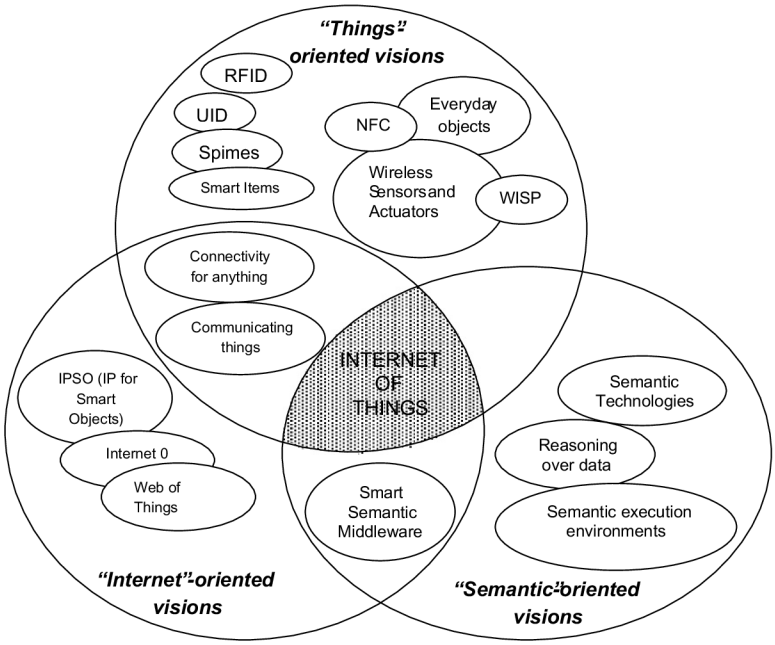
Přesto však internet věcí nemá žádnou jednotnou definici. Liší podle toho, jaká výzkumná komunita ji vytváří a na jaké cílové čtenáře cílí. Tímto vzniká nejednoznačnost, co to vlastně internet věcí je. Jako vhodná cesta pro porozumění celému internetu věcí a různých pohledů na něj se jeví náhled na problematiku ze tří pohledů, které jsou zobrazeny na obrázku 2.1 [19]. Na základě článku The Internet of Things: A survey[19] se dá internet věcí rozdělit na:

Věcně orientovaný pohled se zaměřuje na jednotlivé věci, jejich identifikaci, nízkoúrovňovou komunikaci atd.

Internetově orientovaný pohled se zaměřuje na komunikaci a propojení všech prvků internetu věcí.

Sémanticky orientovaný pohled se zaměřuje na technologii pro uchování, vyhledání, třídění a interpretování dat generovaných všemi připojenými zařízeními.

Je zřejmé, že ani jeden pohled nepostačí pro komplexní definici internetu věcí, ale musí se tyto pohledy propojit. Jedny z prvních definic byly věcně orientované a vycházely z průmyslu, kde bylo potřeba monitorovat a sledovat objekty zájmu, proto se zejména zaměřovaly na jednoduché senzory, které odesílaly informace o své pozici. Tento pohled je rozhodně klíčový pro internet věcí, protože bez koncových zařízení by nemohl IoT existovat. Avšak není jediný a mnoho aspektů internet věcí nezahrnuje. Na reprezentaci, uchovávání, vyhledávání, organizaci a zpracovávání informací získaných z internetu věcí se zaměřuje sémantický pohled. Protože se očekává, že počet objektů v internetu věcí bude nesmírně narůstat, tak sémantické technologie v tomto ohledu budou hrát klíčovou roli. Propojení a komunikaci mezi objekty se věnuje internetově orientovaný pohled. V rámci tohoto směru se zrodila například IPSO alliance (IP for smart objects), která se snaží prosazovat IP protokol pro chytré objekty. Dalšími technologiemi v tomto směru jsou 6LoWPAN, Internet 0 a další.[19]



Obrázek 2.1: Pohledy na Internet věcí

2.1 Minulost internetu věcí

Internet věcí je považován za novinku posledních desetiletí. Je ale vhodné si připomenout, že nad konceptem podobným internetu věcí přemýšlel třeba myslitel a vynálezce Nikola Tesla již v roce 1926, kdy v rozhovoru pro časopis Colliers řekl: „Když bude bezdrátová technologie dokonale rozvedena, celý svět se změní v jeden velký mozek, kde všechny věci budou součástí reálného a rytmického celku ... a nástroje, skrze které bychom toho byli schopni, budou v porovnání se současným telefonem úžasně jednoduché. Člověk jej bude moct nosit ve své kapse.“ [21] Na svou dobu se tato myšlenka velice blíží popisu stavu, kdy je internet věcí všudypřítomný, dokáže sledovat a vyhodnocovat data, která sbírá po celé Zemi.

O 24 let později, v roce 1950, se Alan Turing, zakladatel moderní informatiky, ve svém článku Výpočetní mechanismy a inteligence v Oxford Mind Journal vyjádřil: „... že může být dobré poskytnout stroj s nejlepšími smyslovými orgány, jaké se dají za peníze koupit a pak je učit rozumět a mluvit anglicky. Tento proces může být podobný jako běžná výchova dětí.“ [24] Turing se v této myšlence přibližuje k představě, že stroje dokáží myslet, lépe řečeno, že stroje jsou schopny učit se na základě složitých algoritmů ze svého okolí a pak tyto znalosti aplikovat. Tento přístup se velmi blíží budoucím představám o internetu věcí, kdy stroje spolu komunikují přímo, bez zásahů člověka.

V roce 1966 Karl Steinbuch řekl: „V pár desetiletích budou počítače integrovány do skoro veškerého průmyslového produktu.“, což je jedna z dalších myšlenek internetu věcí, kdy mohou být téměř všechny věci vybaveny výpočetní jednotkou, která je miniaturní a dokáže komunikovat se svým okolím. V internetu věcí jsou tyto výpočetní jednotky reprezentovány svoji adresou a jsou schopny komunikovat přes internet.

V roce 1969 byla spuštěna počítačová síť ARPANET, kterou vnímáme jako předchůdce dnešního internetu. Tato síť umožnila připojení spotřebičů a senzorů. Jeden z prvních pro-

jektů, který tuto možnost využil byl nápojový automat na univerzitě Carnegie Melon přibližně v roce 1982. Tento automat umožňoval lidem kontrolovat inventář automatu a zdali jsou nově vložené nápoje již vychlazeny. Tyto informace bylo možné(a stále je) získat vzdáleně pomocí Finger protokolu.

Pojem síť „internet“, který vychází z ARPANETu vzniká až v roce 1987 a za další 3 roky se do této sítě připojí první spotřebič jiný než počítač. Jedná se o toustovač, který připojili John Romkey a Simon Hackett a umožňují jej skrz internet zapínat a vypínat.

I přesto, že se myšlenka internetu věcí vyvíjí a formuje již pár desítek let, jeho jméno vzniklo až v roce 1999 Kevinem Ashtonem jako titulek prezentace vytvořené pro společnost Procter & Gamble.

V dalších letech se vývoj značně urychlil a na svět přicházelo čím dál více produktů, vynálezů a událostí, které měnili tvář internetu věcí. Některé z nich jsou:

- LG představila plány na první inteligentní ledničku. (2000)
- Založena IPSO alliance. (2008)
- Google představil samořídící auto. (2010)
- Vznikl Bluetooth low energy. (2010)
- NEST koupeno společností Google, představili inteligentní termostat. (2011)
- Spuštění IPV6. (2011)
- Apple představuje produkty: ibeacon, homekit, healthkit. (2014)

Kapitola 3

Inteligentní domácnost

Pojmem inteligentní domácnost, chytrý dům nebo smart home se rozumí prostředí domu, které zvyšuje komfort svým uživatelům a reaguje na jejich podněty a potřeby. Má za úkol umožnit uživateli kontrolovat jeho domov, zabezpečit objekt, zpříjemňovat zábavu, snížit náklady na provoz a další. Aby toto mohl inteligentní dům splňovat, musí mít k dispozici data, která určují stav objektu a přístup k řízení jednotlivých částí domova. Pro sběr dat se používají různé typy senzorů, které dokáží zaznamenat různé veličiny a digitalizovat je. Pro ovládání jednotlivých částí domácnosti bývají instalovány aktuátory, které dokáží ovládat zařízení.

Současná řešení těchto systému se rozdělují především na dva přístupy:

- Systémy s centrální řídící jednotkou
- Systémy s decentralizovaným inteligentním řídícím systémem

Oba přístupy se však potýkají se stejným úkolem, a to jakou formou bude vytvořeno spojení mezi senzorem, aktuátorem a případně řídící jednotkou. Řešení této komunikace se provádí buď drátovou nebo bezdrátovou formou. Často se přistupuje i k řešením hybridním, kdy se mohou používat obě formy, aby bylo zaručena interoperabilita systému.[20]

Pro drátové technologie se může použít již existující telefonní kabeláž, kde většinou komunikují zařízení inteligentní domácnosti v pásmu, které neruší případné hovory či DSL komunikaci. Může být využita elektrická síť, kde se jednotlivé prvky inteligentní domácnosti vloží do elektrických zástrček a dokáží spolu komunikovat pomocí elektrické sítě. Jiným řešením drátové komunikace může být použitím ethernetového nebo jiného kabelu. Jejich použití však většinou znamená zásah do současné elektroinstalace při montáži.

U bezdrátových technologií je nutné dbát na dostatečnou propustnost sítě, její zabezpečení, odolnost vůči rušení, nízkou spotřebu a adekvátní dosah. Bezdrátové technologie umožňují dynamickou rekonfiguraci sítě, což znamená jednoduché přidání, odebrání nebo přemístění zařízení. Umožňují použití baterií napájených modulů, které nevyžadují žádnou kabeláž. Mezi používané bezdrátové technologie patří například WLAN, Bluetooth, Z-Wave nebo ZigBee.

3.1 Existující řešení

Řešení v různých podobách existují již poměrně dlouho. Firmy většinou tíhnou spíše k individuálním projektům pro každého zákazníka a instalaci provedou jejich zaměstnanci nebo odborníci. Téměř vždy mají centrální jednotku s možnostmi připojení zařízení pomocí různých sběrnic. Projektové řešení však nebývá jediným obchodním modelem těchto firem, ale z jejích prezentací vyplývá snaha jít spíše tímto směrem. Zároveň se však snaží být i modulární a umožnit klientům si zařízení přikupovat a vkládat do jejich řešení inteligentní domácnosti.

Řešení posledních let spíše používají bezdrátové technologie a snaží se zákazníky nalákat na vyšší modularitu a podporu zařízení jiných firem. Zařízení tohoto typu jsou navrhnuty takovým způsobem, aby i pro zákazníka-amatéra nebyl problém daný výrobek nainstalovat do své domácnosti a začít jej používat. Oslovují zákazníka možnosti zakoupit si minimální počet zařízení, vyzkoušet je a v případě spokojenosti dokoupit další. Zařízení modulárních řešení jsou většinou navrženy tak, aby je nebylo složité kdykoliv přemístit a nainstalovat na jiná místa.

mFi

Řešení mFi od společnosti Ubiquiti Networks je ovládáno softwarem mFi Controller. Tento software se instaluje do uživatelova počítače a pracuje se s ním přes internetový prohlížeč. Umožňuje ovládat domácnost a nastavovat pravidla, podle kterých se inteligentní objekt bude řídit. Ubiquiti Networks bohužel neposkytují žádnou aplikaci na mobil, kterou by uživatel mohl domácnost ovládat.[8]

Ubiquiti Networks rozděluje svá mFi zařízení do několika skupin:

- In-Wall Devices zahrnuje dvě zařízení s Wi-Fi pojmenované Mfi Outlet, jedná se o zásuvku ve zdi, která dokáže monitorovat spotřebovanou energii a mFi Switch, který funguje jako spínač/stmívač umožňující ovládat různé druhy světelných zdrojů.
- mPower jsou vzdáleně ovládatelné zásuvky přes Wi-Fi, podobné mFi Outlet, ale nejsou ve zdi a zároveň poskytují větší počet zdířek.
- mPort je centrální jednotka, která vytváří síť inteligentní domácnosti. K jednotce se můžou zařízení připojit bezdrátově přes Wi-Fi, kabelem RJ45 nebo přes mFi port.

Haidy Home

Řešení HAIDY Home je modulární sběrnicový systém. Sběrnice HAIDY Home je typu CIB a může být jak drátová, tak bezdrátová. Možná je i kombinace. Bezdrátové řešení využívá frekvenci 868 MHz a je dostupné přibližně od poloviny roku 2014. Prvky jsou připojeny buď přímo v rozvaděči, nebo v podomítkových krabicích v místnostech, podle toho, co je výhodnější z hlediska instalace. Na sběrnici v jednotlivých místnostech tak mohou být připojeny jak ovládací a snímací prvky (např. vypínače, senzory teploty nebo CO2), tak i akční členy ovládající příslušný spotřebič (např. servohlavice topení, spínače osvětlení a zásuvek, stmívače). Haidy Home se dá ovládat prostřednictvím počítače nebo mobilu skrz android nebo iOS aplikaci.[7]

iNELS

Elko EP nabízí dvě řešení, jedno je drátové, iNELS Bus systém, které je vhodné pro novostavby nebo rozsáhlé rekonstrukce budov. Bezdrátové řešení se jmenuje iNELS RF control, které nevyžaduje žádný zásah do stávajícího objektu. Tyto systémy jsou řízeny řídící jednotkou, ke které jsou další zařízení připojena skrz sběrnici CIB nebo bezdrátově. Toto řešení inteligentní domácnosti lze ovládat pomocí mobilu přes Android nebo iOS aplikace, přes chytrou televizi, ovládací jednotky RF Touch, které jsou vybaveny dotykovým displejem a dále je možnost ovládat domácnost pomocí bezdrátových klíčenek, kterým jde nastavit funkce.

WeMo

WeMo od společnosti Belkin je řešení, kde se téměř každé WeMo zařízení připojuje k internetu pomocí bezdrátové technologie Wi-Fi a tím je nezávislé na ostatních zařízeních. Pomocí Android nebo iOS aplikace si uživatel dané zařízení přidá do své sbírky a může ho začít nastavovat, spínat atd. Velká modularita a nezávislost jednotlivých zařízení je však vykoupená požadavkem na Wi-Fi přijímač/vysílač v každém WeMo zařízení a to má dopad na cenu zařízení.

Pro rozšíření funkčnosti vsadilo WeMo na podporu technologie IFTTT, která umožňuje propojit WeMo aplikaci s velkým množstvím dalších služeb. V IFTTT si uživatel může nastavit spouštěč a akci která se vykoná. Spouštěč může být například zmáčknutí tlačítka, aktuální teplota nebo hodina. Akce pak může být například změna stavu nějakého spotřebiče v domácnosti.

Nest

Firma Nest je známá především svým chytrým termostatem, který má sofistikovaný software, ten se učí podle toho jakým způsobem uživatel nastavuje teplotu v průběhu dne a na základě tohoto se učí požadavky uživatele a potom je aplikuje na automatické nastavování teploty v domácnosti. Svoji nabídku firma ještě rozšířila o detektor kouře a CO2. Termostat a detektor mohou být ovládány pomocí aplikace pro android nebo iOS. Termostat Nest podporuje široké zastoupení protokolů, které se využívají v tomto odvětví a proto většině uživatelů (podle Nestu více než 95%) stačí pouze vyměnit stávající termostat, případně pak lze využít zařízení Heat Link, které se připojí přímo ke kotli a bezdrátově komunikuje s termostatem. Firma Nest spolupracuje i s dalšími firmami, aby mohla nabídnout svým uživatelům větší pohodlí. Některé z spolupracujících firem jsou: Mercedes-Benz, Dropcam, Jawbone, Google, IFTTT, Logitech.

Insteon

Insteon přichází s řešením, které využívá svůj vlastní protokol jménem Insteon. Tento protokol pracuje s dvěma frekvenčními pásmy a kombinuje současnou domácí elektrickou síť a bezdrátovou technologii. Toto řešení firma zvolila pro zvýšení odolnosti systému proti rušení bezdrátového signálu. Síť je typu peer-to-peer a nevyžaduje žádný řídící prvek, má větší dosah než bluetooth a je levnější než Wi-Fi. Aby se tato Insteon síť dala ovládat i z mobilního telefonu přes iOS, Android nebo Windows Phone aplikaci, nabízí firma Insteon hub, který se stane bránou mezi internetem a Insteon sítí. Na obrázku 3.1 [6] je v popředí vidět Insteon hub a v pozadí různá zařízení tohoto řešení.

Obrázek 3.1: Zařízení Insteon

SmartThings

SmartThings podporuje ZigBee, Z-Wave, Wi-Fi zařízení a umožňuje i dalším firmám, aby se staly součástí tohoto řešení, v současnosti nabízejí zařízení od Honeywell, GE, Kwikset, Schlage, Trane a ve vývoji jsou zařízení od firem Philips hue, Sonos, Dropcam, Belkin, Withings, Ecobee. Zařízení ZigBee a Z-Wave se připojují pomocí brány SmartThings Hub. SmartThings se da ovládat pomocí aplikace na Android, iOS nebo Windows Phone a jde napojit na službu IFTTT, která výrazně rozšiřuje funkcionalitu řešení. Na obrázku 3.2 [5] jdou zobrazeny zařízení a zařízení vpravo je brána SmartThings Hub.

Obrázek 3.2: Zařízení SmartThings

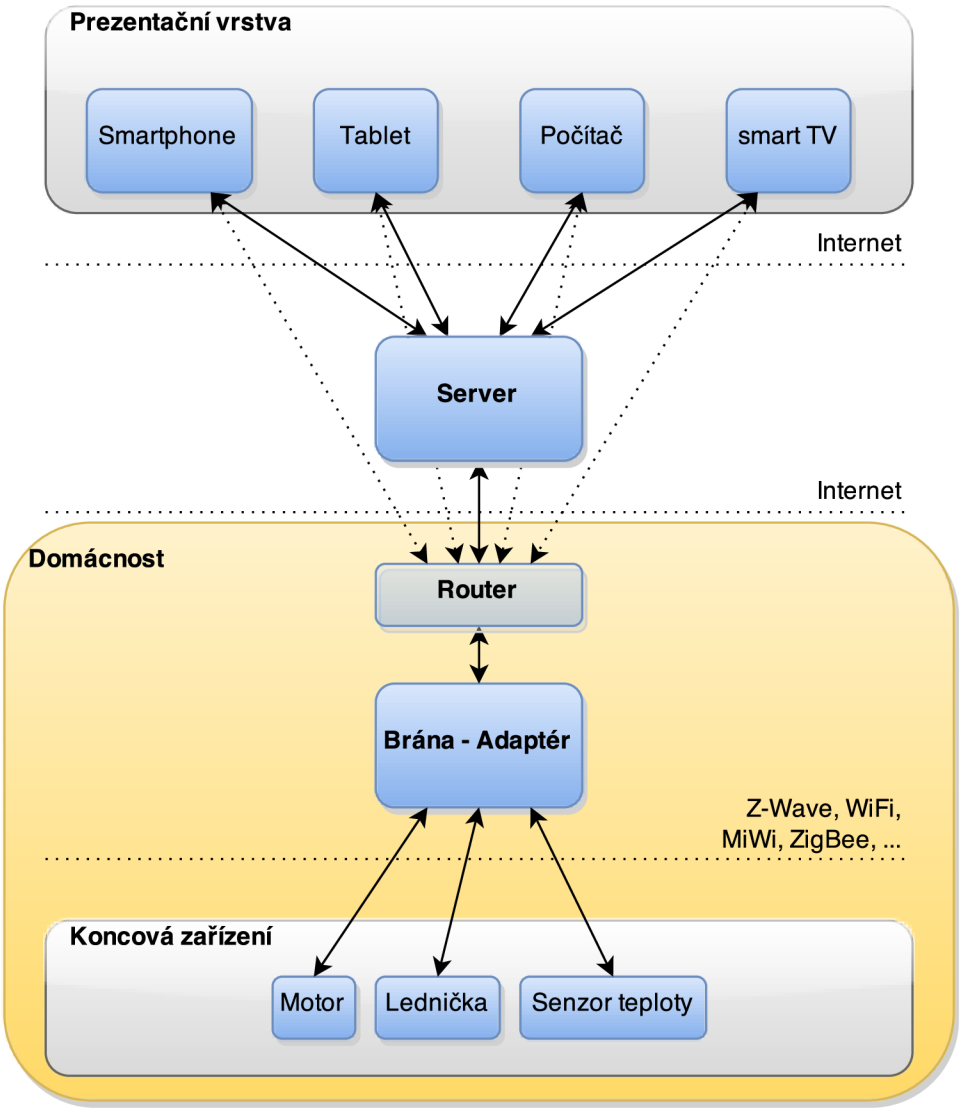
3.2 Inteligentní domácnost na FIT VUT

Tento projekt si klade za cíl vytvořit systém, který bude sloužit jako platforma pro výuku a výzkum v dané oblasti. Vyvíjené řešení musí být schopno podporovat širokou řadu senzorů a aktorů (koncových zařízení), a to nejen zařízení vytvořené v rámci tohoto projektu, ale i zařízení vytvořené třetími stranami, které se budou do tohoto řešení integrovat. Koncová zařízení budou připojena pomocí bezdrátové sítě k bráně inteligentní domácnosti. Tato brána bude přijímat a odesílat data koncových zařízení a zároveň komunikovat se serverem, který bude senzorická data uchovávat v databázi a zprostředkovávat je mobilním zařízením nebo jiným prezentačním platformám. Na serveru bude zároveň umožněn výpočet různých algoritmů z uživatelských a senzorických dat.

3.2.1 Architektura inteligentní domácnosti

Architektura inteligentní domácnosti vyvíjené na FIT má čtyři vrstvy, které jsou zobrazeny na obrázku 3.3. Nejnižší vrstva se skládá z koncových zařízeních, které obsahují čidla snímající veličiny ve svém okolí a aktory, které se dají ovládat a řídit pomocí informací poslaných z brány.

Další vrstvou je brána inteligentní domácnosti, v rámci tohoto projektu je brána také nazývána adaptér. K adaptéru jsou jednotlivá koncová zařízení připojena a podle potřeby mezi sebou komunikují. Adaptér je připojen na server, kterému posílá data o senzorech a případně přijímá informace o nastavení koncových zařízeních. Server data zpracovává a ukládá do databáze a v případě požadavku z prezentační vrstvy z ní data čte.



Obrázek 3.3: Architektura inteligentní domácnosti

Koncová zařízení

Zařízení inteligentní domácností se dají rozdělit do dvou skupin, jedná se o senzory, které monitorují své okolí pomocí interních a externích čidel a aktivní prvky, které své okolí ovlivňují.

Tyto prvky jsou napojeny na adaptér, se kterým komunikují a jsou reprezentovány svým jedinečným identifikátorem. Koncová zařízení vždy komunikují pouze s jedním adaptérem, se kterým jsou spárována. Pro spárování musí uživatel uvést adaptér i dané koncové zařízení do párovacího režimu, což se může provádět například zmáčknutím tlačítka. Adaptér se do párovacího režimu uvede aktivací v prezentační vrstvě.

Komunikace na této vrstvě probíhá bezdrátově a je kladen důraz na odolnost vůči rušení, plochu pokrytí signálu a odolnost vůči odposlechu posílaných informací.

Zařízení mohou být napájeny buď baterií, nebo napájením z elektrické sítě. Pro prodloužení životnosti zařízení s baterií se používá spánkový režim, při kterém se nevysílají data, ani nerealizuje nějaký užitečný výpočet. Ze spánkového režimu se v pravidelných intervalech probouzí, aby poslalo data adaptéru. Zařízení v elektrické síti není potřeba uvádět do spánkového režimu a díky tomu je možné je použít jako zařízení pro zvětšení dosahu sítě inteligentní domácnosti.

Brána (Adaptér)

Adaptér je bránou mezi zařízeními inteligentní domácnosti a internetem. Jeho základní funkce je jako mezivrstva mezi spárovanými koncovými zařízeními a serverem. Na server posílá naměřená data a zařízením posílá data, která přijal ze serveru.

Rozšířená funkcionalita spočívá v offline režimu, který umožní uživateli komunikovat s koncovými zařízeními i bez internetové konektivity. Adaptér komunikuje přímo s mobilním telefonem, tabletem, chytrou televizí nebo počítačem.

Oproti koncovým zařízením musí být adaptér spolehlivý a funkční po celou dobu, to vylučuje možnost napájení pomocí baterie, a proto se volí připojení do elektrické sítě.

Server

Server jednak komunikuje s adaptérem, který mu posílá data inteligentní domácnosti. Zároveň prostřednictvím adaptéru nastavuje zařízení v domácnosti podle podnětů od uživatele. Příchozí data jsou ukládány do databáze, aby mohly být na dotaz poslány na prezentační vrstvu.

Kromě komunikace typu dotaz-odpověď mezi serverem a prezentační vrstvou, má server možnost asynchronně komunikovat s uživatelem pomocí takzvaných push notifikací. Tato možnost je využívána pro události, o kterých chce uživatel okamžitě vědět, může se například jednat o neoprávněné vniknutí do objektu apod.

Zabezpečení komunikace je nezbytné, protože pokud by se útočníkovi povedlo přistoupit k nezašifrovaným datům, mohl by získat přístup k inteligentní domácnosti daného uživatele a získaná data zneužít, případně ovládat domácnost nebo vypnout bezpečnostní prvky a tím si objekt zpřístupnit.

Server podporuje výpočet různých algoritmů, které mají rozšířit možnosti inteligentní domácnosti a zvýšit komfort uživatele, jedná se o úlohy jako je předpověď počasí, automatické zavírání oken pokud prší nebo fouká silný vítr, šetření nákladů na vytápění objektu a podobně.

Prezentační vrstva

Slouží k ovládání inteligentní domácnosti uživatelem prostřednictvím různých aplikaci, například na mobilu, chytré televizi, počítači nebo na jiné platformě. Uživatel v těchto aplikacích vidí stav své inteligentní domácnosti a je mu umožněno ji nastavovat a ovládat služby, které server uživateli zpřístupní.

Kapitola 4

Databáze

Databáze inteligentní domácnosti slouží jako perzistentní úložiště dat, které musí být schopno uchovávat velké množství historických dat naměřených nainstalovanými senzory. Databáze musí být schopna provádět různé agregační dotazy, které zpřístupní velké množství dat. Získaná data slouží pro zisk informací v intervalu několika dní, týdnů nebo delších období.

V této kapitole jsou probírány některé současné databáze, které by mohly být vhodným kandidátem na databázi inteligentní domácnosti.

4.1 Relační databáze

Technologie relačních databází byla původně navržena E.F.Coddem a později ji implementovala firma IBM a další. Standard SQL je popsán ANSI a ISO normou, častěji se na ni ovšem odvoláváme jako na „SQL“ a číslo verze.

Relační systém řízení báze dat (RSŘBD) uchovává data v databázi skládající se z řádků a sloupců. Řádek odpovídá záznamu (record, tuple) a sloupce odpovídají atributům (polím v záznamu). Každý sloupec má určen datový typ. Datových typů je omezené množství a jsou závislé na konkrétním řešení databáze. Každý atribut záznamu může uchovávat jedinou hodnotu. Vztahy nejsou explicitní, ale spíše plynou z hodnot ve speciálních polích, tzv. cizí klíče (foreign keys) v jedné tabulce, který se rovná hodnotám v jiné tabulce. Pohled (view) je podmnožina databáze, která je výsledkem vyhodnocení dotazu.
V RSŘBD je pohled tabulka. RSŘBD využívá SQL pro definici a řízení dat, pro přístup k datům a získávání dat. Data jsou získávána na základě hodnoty v určitém poli záznamu.

Veškeré zpracováváni je založeno na hodnotách polí záznamů. Záznamy nemají jednotné identifikátory, které jsou neměnné po dobu existence záznamu. Neexistují žádné odkazy z jednoho záznamu na jiný. Vytvoření výsledku je prováděno pod kontrolou kurzoru, který umožňuje uživateli sekvenčně procházet výsledek po jednotlivých záznamech. Totéž platí pro update.[9]

4.2 Objektově-relační databáze

„Rozšířená relační“ a „objektově-relační“ jsou synonyma pro databázové systémy, které se snaží sjednotit rysy jak relačních, tak objektových databází. ORSŘBD je specifikován v rozšíření SQL standardu — SQL3.

ORSŘBD využívá datový model tak, že „přidává objektovost do tabulek“. Všechny trvalé informace jsou stále v tabulkách, ale některé položky mohou mít bohatší datovou strukturu, nazývanou abstraktní datové typy (ADT). ADT je datový typ, který vznikne zkombinováním základních datových typů. Podpora ADT je atraktivní, protože operace a funkce asociované s novými datovými typy mohou být použity k indexování, ukládání a získávání záznamů na základě obsahu nového datového typu. ORSŘBD jsou nadmnožinou RSŘBD a pokud nevyužijeme žádné objektové rozšíření jsou ekvivalentní SQL2. Proto má omezenou podporu dědičnosti, polymorfismu, referencí a integrace s programovacím jazykem.

ORSŘBD podporuje rozšířenou verzi SQL – SQL3. Důvodem je podpora objektů (tj. dotazy obsahující atributy objektů). Typická rozšíření zahrnují dotazy obsahující vnořené objekty, atributy, abstraktní datové typy a použití metod. ORSŘBD je stále relační, protože data jsou uložena v řádcích a sloupcích tabulek a dotazy SQL, včetně zmíněných rozšíření, pracuje právě s nimi.

Jazyk SQL s rozšířením pro přístup k ADT je stále hlavním rozhraním pro práci s databází. Přímá podpora objektových jazyků stále chybí, což nutí programátory k překladu mezi objekty a tabulkami.[9]

4.3 Objektové databáze

V objektovém SŘBD již nejsou informace uchovávány v tabulkách jako tomu bylo u relačních a relačně objektových databází, ale ve formě objektů, které jsou snadněji přístupné v objektově orientovaném jazyce a pro programátora snadněji použitelné. Objektové databáze mají plnou podporu objektů, jejich zapouzdření, dědičnosti, polymorfismu, jednoznačné identifikace objektu a reference mezi objekty. Každý vložený objekt do databáze již není identifikován pomocí primárního klíče, ale pomocí svého OID, které na logické úrovni odpovídá ukazateli do virtuální paměti počítače.

Podobně jako v relačních databázích ukázal vývoj databázových systému nutnost standardizace. Pro objektové databáze vznikl standard ODMG. V rámci standardu ODMG byl navržen dotazovací jazyk OQL. Jedná se o čistě dotazovací jazyk, tedy OQL kromě vyhledávání a výběru neumožňuje žádné manipulace s objekty v databázi. Centrálním příkazem jazyka je příkaz select, který je zobecněnou variantou selectu z jazyka SQL. Jazyk podporuje kvantifikátory, agregační funkce, konverzní funkce, pojmenované dotazy, množinové operace a další. Součástí jazyka jsou i syntaktické konstrukce umožňující definice nových objektů a literálů, avšak vznikají pouze jako tranzientní objekty podílející se na vzniku návratové hodnoty výběrového příkazu, a nejsou tedy vkládány do databáze. OQL je silně typovaný jazyk, výsledkem každé operace je objekt některé třídy, typicky kolekce objektů.[1]

4.4 NoSQL databáze

NoSQL databáze přicházejí v době, kdy množství zpracovávaných a ukládaných dat je pro relační databáze neúnosně velké. NoSQL v sobě zahrnuje mnoho rozdílných systémů, které se hodí na různé problémy. Tyto systémy se vytvářely zejména kvůli řešení problémů, pro které jsou relační databáze nevhodné. Většina NoSQL databází používá ne-relační datový model a nemá definici databázového schéma. Každé NoSQL řešení má odlišný model, který používá.[23]

NoSQL databáze již nejsou vázány principem ACID(Atomicity, Consistency, Isolation,

Durability) jako to bylo u relačních databází. Na úkor konzistence tedy získávají větší rychlost a mohou zaručit lepší dostupnost a škálovatelnost. Škálování v tomto případě probíhá přidáváním dalších serverů, neboli škálování na horizontální úrovni.

NoSQL databáze nepodporují standard SQL jako relační databáze. Některá řešení jsou však inspirována jazykem SQL a byly vytvořeny podobné jazyky, které nemají plnou funkcionalitu jako SQL, například nenabízí vlastnosti jako jsou operace agregace, seřazení nebo spojení(join). Příkladem těchto jazyků může být GQL (Google Query Language) nebo HQL (Hypertext Query Language).

Jiný způsob dotazování může být pomocí NoSQL API, které typicky podporuje operace:

- Get(klíč, hodnota) – Získej hodnotu daného klíče
- Set(klíč, hodnota) – Ulož nebo aktualizuj hodnotu k danému klíči
- Delete(klíč) – Smaž daný klíč a k němu navázanou hodnotu
- Execute(klíč, operace, parametry) – Nad hodnotou danou klíčem vykonej zvolenou operaci
- Multi-get(key1, key2, ..., keyN) – Získej množinu hodnot danou klíči

NoSQL databáze se rozdělují do čtyř základních skupin:

Sloupcově orientované jsou vhodné pro operace, které pracují se sloupci, jedná se o agregace a úpravy dat v menším počtu sloupců.

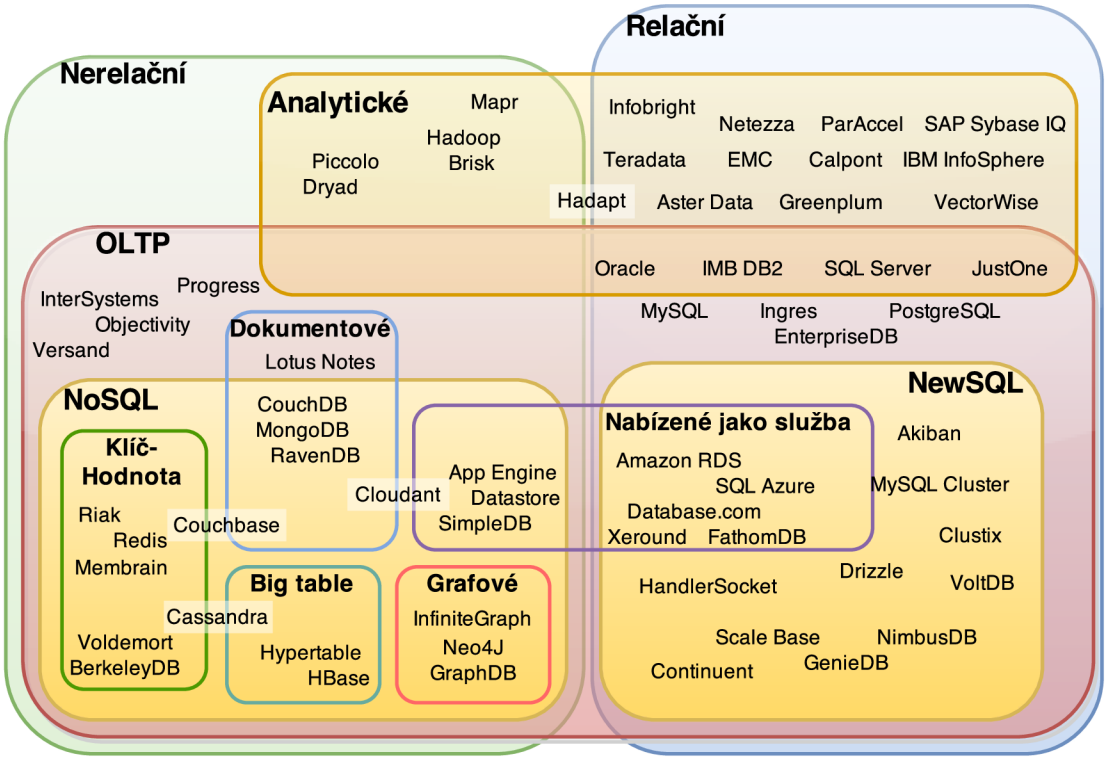
Typu klíč-hodnota, kde jsou data reprezentována jako asociativní pole. Vyhledává se pomocí klíče, přes hash tabulky, což bývá velice rychlé.

Dokumentové, které jsou podobné jako typ klíč-hodnota, ale hodnota je v tomto případě strukturována a jde přistupovat do položek této „struktury“.

Grafové jsou používány pro data, která se dají reprezentovat jako objekty s vzájemnými vztahy.

Mimo těchto základních skupin se někdy vyskytují ještě skupiny XML databáze, které uchovávají data jako XML dokumenty a RDF databáze, které podporují standard W3C RDF.

V diagramu 4.1 [3] jsou zobrazeny typy databází a některé z existujících databázových řešení.



Obrázek 4.1: Diagram databází

Kapitola 5

Návrh databáze pro inteligentní domácnost

Návrh každé databáze začíná sběrem požadavků. V této kapitole jsou uvedeny požadavky na databázi, které byly získány na základě analýzy případů použití a diskuzí s konzultantem. Na základě těchto požadavků a existujících databázových systémů byl vybrán systém databáze inteligentní domácnosti. Dále je v této kapitole ukázáno konceptuální schéma, ve kterém jsou zobrazeny entity a vztahy mezi nimi. Závěrečná část kapitoly je věnována popisu logického schématu, které obsahuje primární klíče, cizí klíče, vazební tabulky a datové typy jednotlivých atributů. Navazujícím krokem je fyzický návrh, jehož výsledkem je způsob uložení objektů a přístup k nim v konkrétním databázovém systému.

5.1 Požadavky na databázi

Databáze musí být schopna uchovávat informace o uživatelích, u kterých je možno uložit jejich jméno, příjmení, pohlaví a odkaz na jejich profilový obrázek. Uživatelé budou mít více možností registrace a přihlášení: pomocí mailu a hesla, pomocí služeb Google a Facebook. Zároveň se jim bude počítat, kolikrát se do aplikace přihlásili.

Uživatelé budou využívat komunikační zařízení, které budou mít jedinečný identifikátor. K zařízením se dále bude ukládat jejich typ, lokalizace a identifikaci sezení se serverem. Posledním parametrem zařízení bude řetězec sloužící serveru k iniciaci komunikace se zařízením pomocí takzvaných push notifikací. Zprávy notifikací se budou ukládat společně s úrovní důležitosti notifikace, časovým razítkem a informací, zdali si uživatel zprávu přečetl.

Uživatel může mít přístup k více inteligentním domácnostem, které jsou reprezentovány adaptérem. Na každém přístupném adaptéru musí mít uživatel jednu z rolí: host, uživatel, správce, nebo majitel domácnosti. K adaptérům se ukládají informace o jménu domácnosti, verzi jeho software, čísle socketu, časovém pásmu a době ukládání historických dat.

K adaptéru mohou být připojena koncová zařízení, která jsou identifikována svou MAC adresou a jsou k nim uloženy informace o době spánku zařízení, stavu baterie, kvality bezdrátového signálu, inicializaci, času připojení do sítě a volitelném zařazení zařízení do některého z pokojů. Pokoje jsou přiřazeny k dané domácnosti a mohou mít svůj typ, který určuje, zdali se jedná o kuchyň, obývák, koupelnu apod.

Koncová zařízení se skládají z čidel a aktivních prvků, které jsou rozlišovány svým typem. Uživatelé je mohou pojmenovat a nastavit jejich viditelnost ve svém prezentačním zařízení. Příchozí data z čidel a aktivních prvků budou ukládány společně s časovým ra-

zítkem. Historická data senzorů bude databáze uchovávat po dobu jednoho měsíce. Starší data se budou z databáze mazat.

Kromě výše zmíněných pokojů, může uživatel rozdělovat čidla a aktivní prvky do takzvaných pohledů. Uživatel může pohledy vytvářet, pojmenovat je, přiřadit k nim typ ikony a určit jaká čidla a aktivní prvky bude pohled obsahovat.

V inteligentních domácnostech budou uživatelům dostupné funkce, které například dokáží ovládat inteligentní domácnost na základě uživatelských pokynů nebo upozornit uživatele na událost, která v domácnosti nastala. Funkce mají své jméno, typ a mohou být nastavovány v rámci celé domácnosti, takže s nimi mohou pracovat všichni uživatelé domácnosti a algoritmus může pracovat se všemi zařízeními v domácnosti. Druhou možností používání funkcí je individuální, kdy každý uživatel ovládá algoritmus zvlášť, definuje podmnožinu zařízení, se kterými může funkce pracovat v rámci domácnosti a ostatní uživatelé o tomto nastavení neví a nemůžou jej měnit.

Pro podporu gamifikace v řešení inteligentní domácnosti je potřebné uchovávat úspěchy, které jednotliví uživatelé získávají, když pracují s prvky inteligentní domácnosti. Úspěchy jsou rozdělovány na úspěchy uživatele a úspěchy celé domácnosti. Mají svůj identifikátor, kategorii, dílčí body pokroku v daném úspěchu, počet bodů k naplnění úspěchu a výši zisku virtuální odměny v případě splnění. Obsahují informaci o tom zdali má být uživatel informován, že úspěch existuje a je splnitelný za daných podmínek, nebo jestli o existenci úspěchu neví a bude pro něj jeho případné splnění překvapením. Obsahuje položku, která označuje, jestli má být splnění úspěchu odesláno jako notifikace.

Hlavní metrika řešení databáze je její odezva na dotazy. Z rozhovorů s konzultantem a členy týmu se došlo k předpokládanému zatížení řešení, podle kterého musí databáze obsluhovat všechny dotazy do 1000 milisekund a to z důvodu dostatečně rychlé odezvy na zařízení prezentační vrstvy.

5.2 Výběr databáze

Výběr databáze ovlivňuje mnoho prvků a její dobrý výběr má dopad na rychlost vývoje a kvalitu vyvíjeného systému.

Vzhledem k požadavkům na co nejvyšší výkon databáze nelze do užšího výběru zařadit objektové databáze, které na úkor svého výkonu podporují objektové principy a vytvářejí, pro programátora v objektovém jazyce, bohatší prostředí. Další databáze jsou typu noSQL, kde nelze využít potenciálu sloupcových databází, protože nejsou požadavky na zpracovávání velkého množství hodnot v sloupcích. Obdobné je to s grafovými databázemi, protože v systému inteligentní domácnosti není požadavek na udržování husté sítě mezi entitami. U dokumentových databází by také nešlo využít jejich výhod. Poslední z rodiny noSQL jsou databáze typu klíč-hodnota, nebo případně databáze, které kombinují více přístupů dohromady. Tento typ se přibližuje požadavkům, přesto však ani tento typ databáze nebyl zvolen, protože v dotazech na historická data, se vyhledává velký počet položek podle časového intervalu a tento typ databází, respektive jeho největší představitelé nejsou na tento typ přístupu optimalizovány.

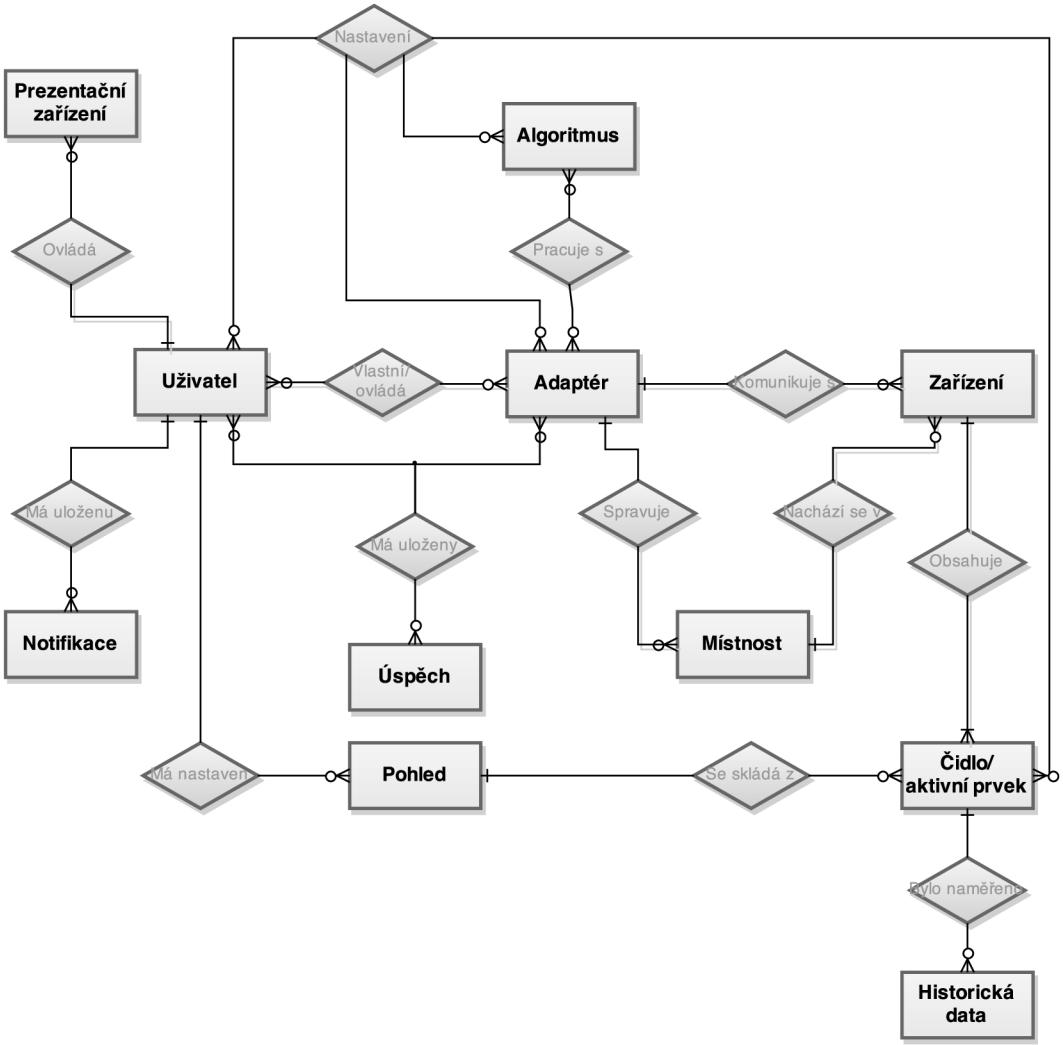
Zbylé typy databází jsou relační, případně objektově-relační. Jejími nejznámějšími zastupiteli jsou komerční databázové systémy: Oracle, IBM DB2, Microsoft SQL Server a open-source: PostgreSQL, MySQL a SQLite.

Při výběru mezi těmito představiteli bylo rozhodováno podle toho, zdali je databázový systém dostupný zdarma a jestli dokáže generovat výstup ve formátu XML, pomocí kterého

se bude komunikovat na serveru v systému inteligentní domácnosti. Vlastnost databáze poskytnout výstup ve formátu XML usnadňuje práci aplikační logice. Jedinou databází, která splnila tyto požadavky byla databáze PostgreSQL. Tento systém je již vyvíjen velice dlouho a existuje pro něj množství pomocných nástrojů. Za dobu jejího vývoje a používání vzniklo mnoho návodů jak s databází pracovat, vytvářet tyto databáze a jak řešit vzniklé problémy, což urychluje vývoj. PostgreSQL se vyvíjí jako univerzitní projekt už téměř dvacet let, pravidelně vycházejí nové verze tohoto software. Je považováno za jedno z předních řešení relačních databází.

5.3 Konceptuální model inteligentní domácnosti

Na základě požadavků na databázi byla identifikována množina entit a vztahů mezi nimi a byl vytvořen konceptuální model, který je vidět na obrázku 5.1. Model obsahuje entity inteligentní domácnosti a ukazuje vztahy mezi nimi.



Obrázek 5.1: Konceptuální model inteligentní domácnosti

5.4 Popis entit a vztahů mezi nimi

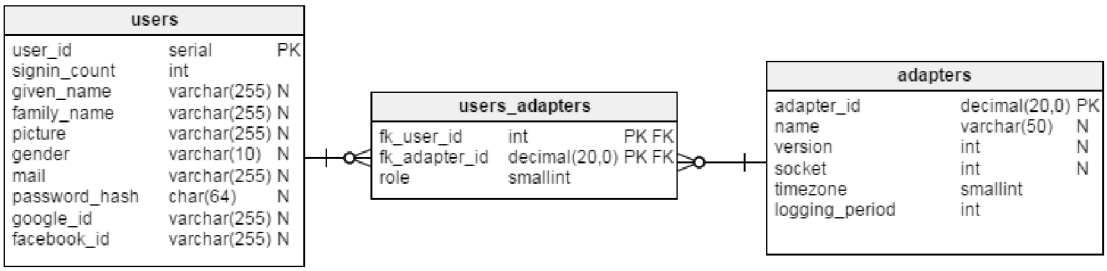
Tato část se bude věnovat entitám a vztahy mezi nimi, které jsou vyznačeny v konceptuálním modelu 5.1. Při popisu bude diagram dekomponován na logické celky, které budou podrobněji popsány.

Přístup uživatele k domácnosti

Základními entitami jsou uživatel a adaptér. Entita uživatele reprezentuje osoby, které používají a ovládají prvky inteligentní domácnosti prostřednictvím adaptéru. Uživatelé mohou mít přístup k více adaptérům a na každém z nich mohou mít jiné pravomoci. Pravomoci se rozdělují na čtyři úrovně: host, uživatel, správce a majitel. Stupeň pravomoci ovlivňuje míru možností ovládání a nastavování dané inteligentní domácnosti.

Adaptér v tomto systému reprezentuje i celou domácnost jako celek, proto se k této entitě váže i pojmenování dané domácnosti. Pomocí vazby mezi entitami uživatel a adaptér je umožněno uživateli pracovat se zařízeními, které se vyskytují v dané domácnosti.

Obrázek 5.2 obsahuje tabulky databáze a jejich atributy odpovídající uživatelům a adaptérům. Tyto dvě tabulky jsou propojeny pomocí vazební tabulky, která uchovává úroveň přístupu uživatele k adaptéru.



Obrázek 5.2: Uživatelé a adaptéry

Zařízení a data v inteligentní domácnosti

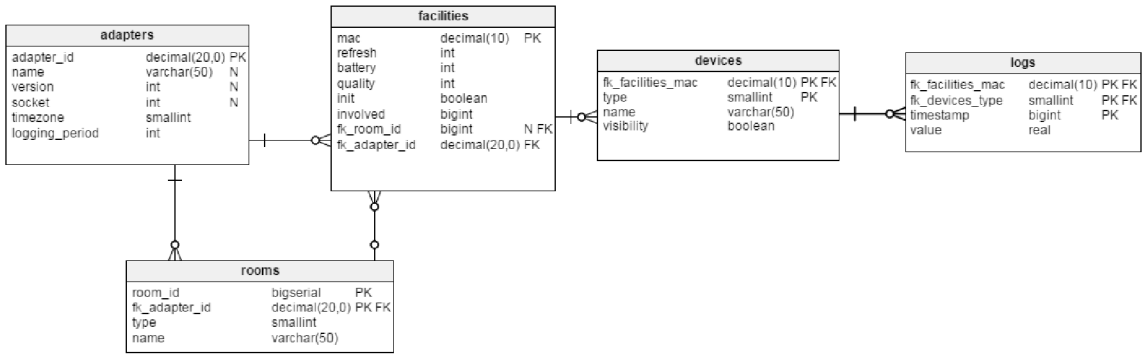
Celá inteligentní domácnost reprezentována adaptérem, se dále člení na koncová zařízení, které v sobě můžou agregovat více komponent. Těmito komponentami jsou čidla a aktivní prvky. Senzory snímají veličiny svého okolí a aktivní prvky se používají pro změnu svého okolí. Jedná se o jednoduché prvky, které většinou nemají pokročilejší aplikační logiku a jsou řízena řídící jednotkou, která je společná pro všechny komponenty na koncovém zařízení.

Data čidel a aktivních prvků se posílají na server pomocí adaptéru. Na serveru se naměřené hodnoty ukládají do databáze s časovým razítkem jako historická data. Ta slouží pro vizualizaci stavu domácnosti prostřednictvím grafů, které si uživatel může prohlížet například v mobilu nebo počítači. Další využití těchto dat je v algoritmech, které mohou data z minulosti používat pro svůj (přesnější) výpočet.

Pro zvýšení přehlednosti grafického uživatelského rozhraní v prezentačních aplikacích se může inteligentní domácnost rozdělit na místnosti, případně jiné části jako garáž, zahrada a podobně. Do těchto místností se poté mohou přiřadit koncová zařízení.

Na obrázku 5.3 jsou zobrazeny tabulky odpovídající entitám vyskytujících se v inteligentních domácnostech. Jedná se o domácnost reprezentovanou adaptérem, která se roz-

děluje na místnosti, koncová zařízení obsahující čidla/aktivní prvky a naměřená historická data.



Obrázek 5.3: Entity inteligentní domácnosti

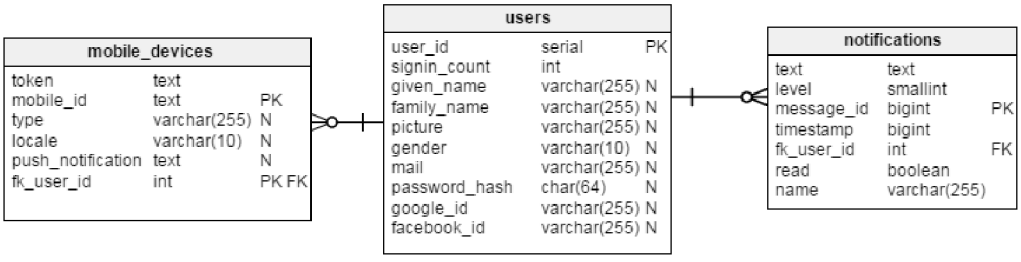
Uživatelská data

Kromě dat konkrétního uživatele se ukládají i informace o prezentačním zařízení, se kterým server komunikuje. Pro správu všech používaných zařízení, v systému inteligentní domácnosti uživatelem, se ukládá typ připojeného zařízení do databáze.

Aby server mohl iniciovat komunikaci s klientským zařízením, využívá se služeb poskytovatelů tzv. push notifikací. Adresace zařízení se provádí pomocí speciálního řetězce, který se ukládá do databáze.

Notifikace mohou uživatele okamžitě informovat o různých událostech a změnách v domácnosti. Notifikace se ukládají do databáze, aby je měl uživatel k dispozici, pokud se je nepovede odeslat na jeho mobilní zařízení, například z důvodu vypnutí aplikace na telefonu nebo nedostupnosti internetu. Tyto notifikace obsahují informace o důležitosti, času vytvoření a obsahu poslané zprávy.

Na obrázku 5.4 jsou zobrazeny tabulky uživatele, mu přidružených zařízeních a poslaných notifikací.



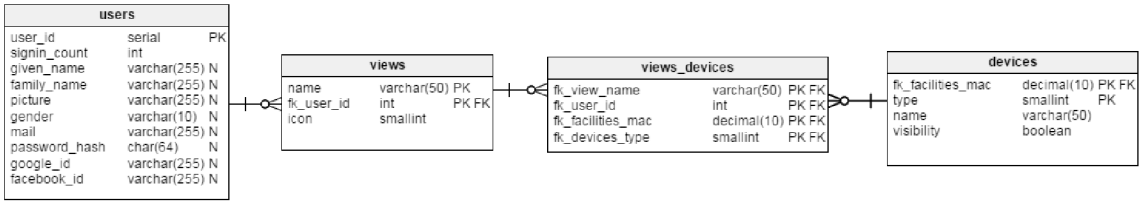
Obrázek 5.4: Osobní data uživatele

Uživatelské pohledy

Pohledy uchovávají množinu čidel/aktivních prvků, které si uživatel zvolil. Slouží pro přehlednější zobrazení dat v aplikacích prezentační vrstvy a umožňují uživateli příjemnější kontrolu stavu své domácnosti.

Každý pohled může obsahovat libovolný počet čidel nebo aktivních prvků. Čidla a aktivní prvky mohou být součástí více pohledů. Tento vztah vzniká a zaniká na uživatelův podnět.

Obrázek 5.5 obsahuje tabulky uživatele, jeho pohledů, sledovaných čidel, aktivních prvků a vazební tabulku mezi pohledy a čidly/aktivními prvky.



Obrázek 5.5: Uživatelské pohledy

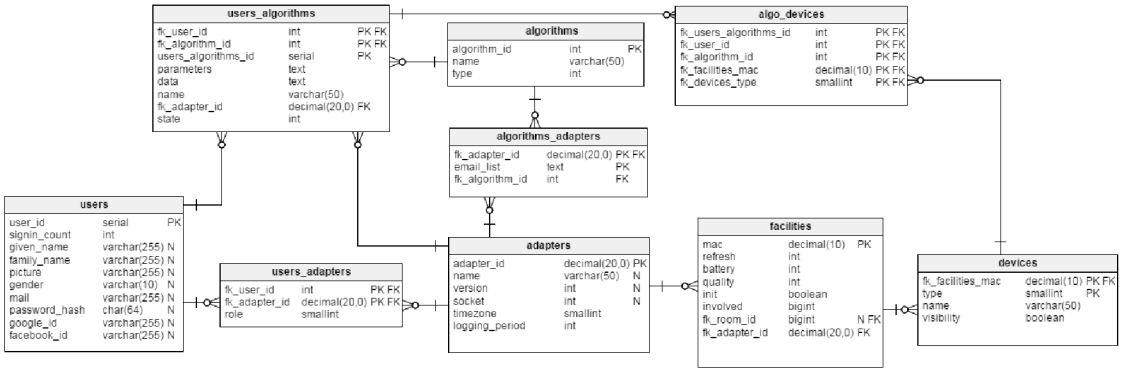
Algoritmy

Entita algoritmy reprezentuje aplikační logiku na serveru, ta přináší do domácnosti další funkce, které dokáží získat odvozené informace z dostupných dat. Tuto logiku mohou uživatelé nastavovat a používat ve své domácnosti. Rozděluje se na algoritmy, které slouží pouze jednomu uživateli a na algoritmy, které jsou pro celou domácnost.

Uživatelé si mohou nastavovat různé algoritmy, které pak sledují hodnoty na vybraných zařízeních, tato data zpracovávají a podle uživatelem definované akce mohou zasáhnout do chodu inteligentní domácnosti nebo uživatele informují o vzniklé události pomocí notifikace.

Některé algoritmy pracují v rámci jedné domácnosti a proto se vztahují na daný adaptér. Mají k nim přístup uživatelé dané domácnosti a algoritmus může pracovat s daty v celé domácnosti.

Obrázek 5.6 zobrazuje tabulku algoritmů a vazebních tabulek, které vytvářejí vztahy s uživateli a inteligentními domácnostmi.



Obrázek 5.6: Algoritmy v inteligentní domácnosti

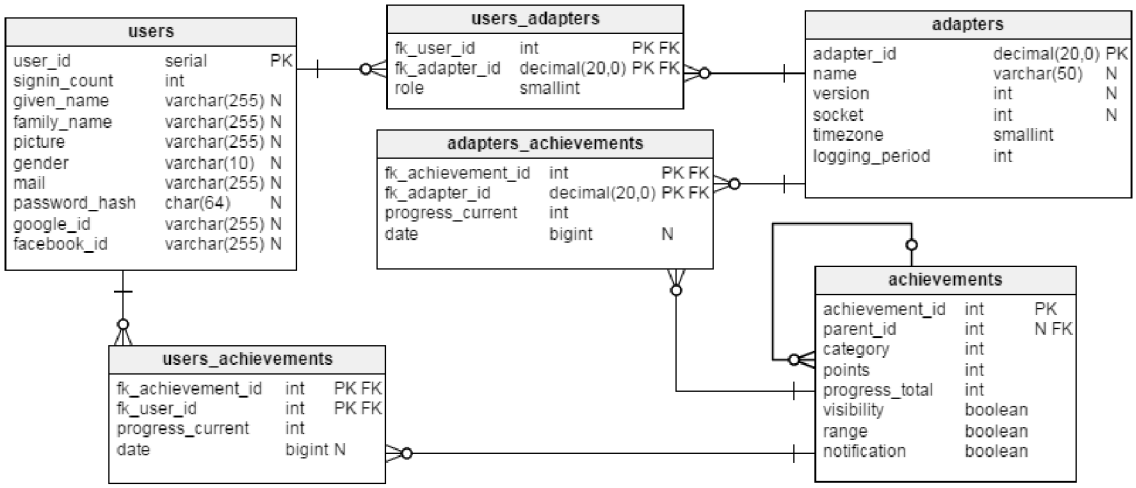
Úspěchy uživatele

Úspěch reprezentuje čin vykonaný uživatelem při používání inteligentní domácnosti. Může se jednat o zakoupení nového senzoru, první použití aktivního prvku nebo pravidelné používání mobilní aplikace. Úspěchy jsou prvky gamifikace, které vkládají do používání inteligentních domácností měřitelný prvek. Pomocí úspěchů mohou uživatelé mezi sebou soutěžit a porovnávat se.

Úspěchy, které uživatelé dosáhli ve svém užívání chytré domácnosti se zaznamenávají do databáze a jsou k dispozici pro uživatele k nahlédnutí. Každý uživatel má přiřazeno více úspěchů, některé z nich mohou mít delší a postupný průběh, proto se zaznamenává i jejich dílčí stav.

Úspěchy mohou být zaznamenávány i v rámci jedné domácnosti, kdy se na něm mohou podílet všichni uživatelé a najednou jej také získají.

Na obrázku 5.7 jde vidět tabulka úspěchů a vazební tabulky, které vytváří vztah s uživatelem nebo adaptérem.

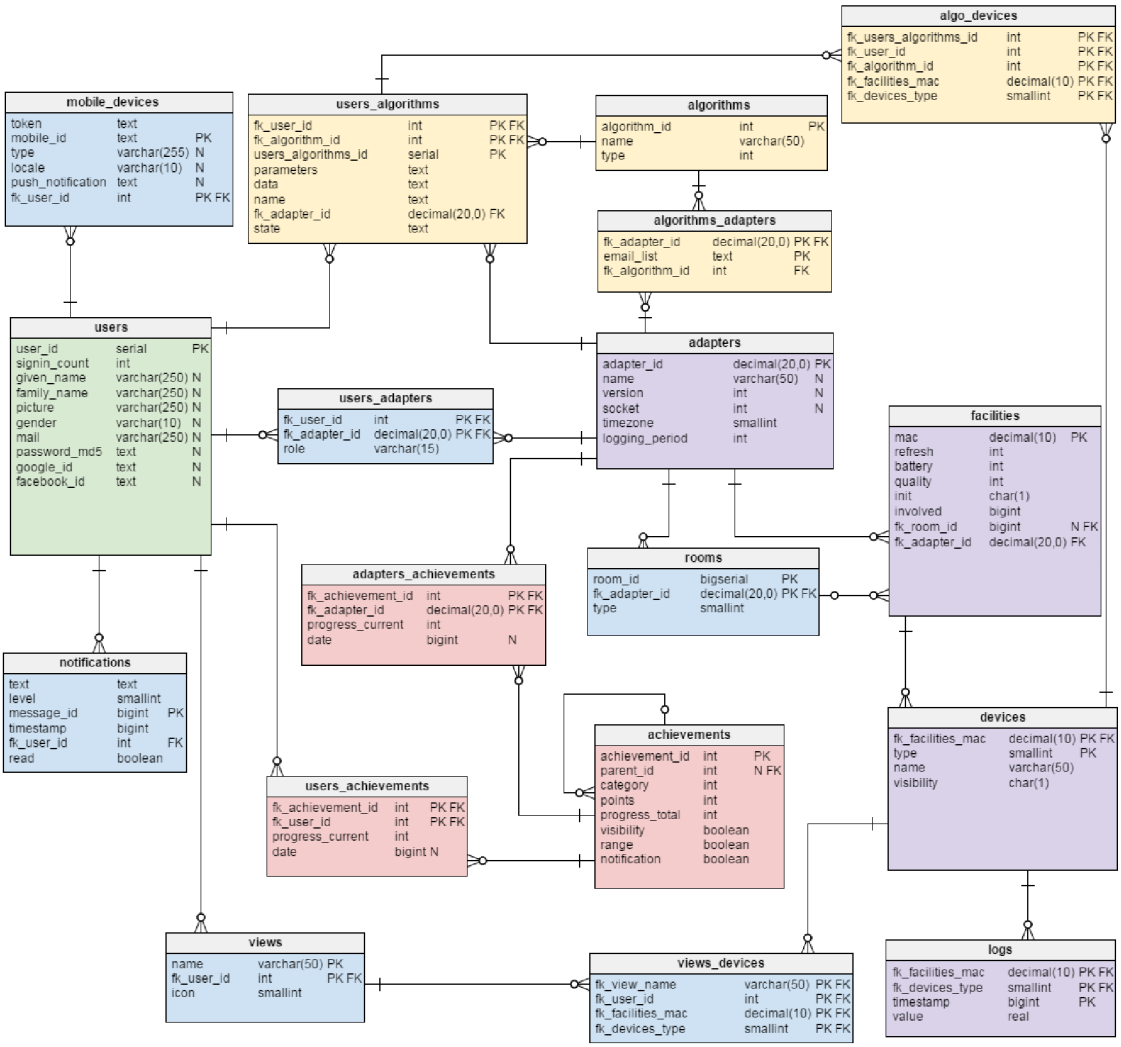


Obrázek 5.7: Úspěchy uživatelů inteligentní domácnosti

5.5 Logický model databáze

Z konceptuálního modelu vychází logický model 5.8, který kromě všech entitních množin a vztahů obsahuje i definice všech atributů, primárních a cizích klíčů.

Model je barevně rozdělen na dříve popsané části: uživatelé, uživatelská data a pohledy, zařízení inteligentní domácností, historická data, algoritmy a úspěchy. Model ukazuje návaznost tabulek databáze inteligentní domácnosti.



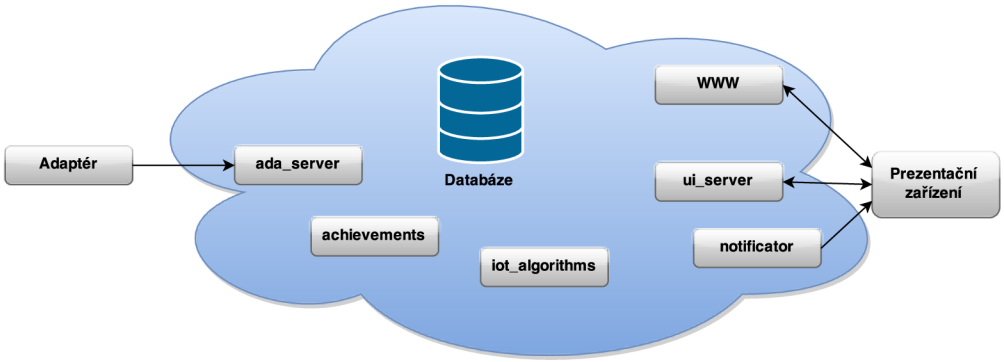
Obrázek 5.8: Logický model inteligentní domácnosti

Kapitola 6

Server

Server je vrstva, která komunikuje s prvky inteligentních domácností skrz adaptéry, ukládá data do databáze, poskytuje data uživatelům prostřednictvím jejich telefonů, počítačů nebo jiných zařízení a umožňuje nad daty v databázi vykonávat různé výpočty.

Na obrázku 6.1 jsou zobrazeny jednotlivé části serveru, jejich pracovní názvy a šipkami nastíněná komunikace s adaptéry a prezentačními zařízeními. V této kapitole jsou tyto části a jejich úlohy popsány. Podrobněji se kapitola věnuje serverové aplikaci ui_server, která byla v rámci této práce implementována.



Obrázek 6.1: Zjednodušené schéma serveru inteligentní domácnosti

Komunikace s adaptéry

Serverová aplikace ada_server má za úkol obousměrnou komunikaci s adaptéry. První komunikaci naváže adaptér ihned po připojení do sítě, aby zaregistroval novou inteligentní domácnost na server. Ada_server přijímá data z koncových zařízení inteligentních domácností a ukládá data do databáze. Komunikace směrem k adaptéru pak probíhá dvojím způsobem v závislosti na typu koncového zařízení, se kterými chce ada_server komunikovat. Pokud se jedná o zařízení s baterií musí komunikovat v době, kdy se zařízení nenachází v režimu spánku. Ada_server se vůči těmto zařízením chová pasivně a čeká až tyto zařízení iniciují komunikaci a pošlou svá naměřená data. Odpovědí ada_serveru je potvrzení, že

data dorazila a zároveň je možná případná změna nastavení zařízení, jako například doba spánku. Jestliže zařízení má stálý přísun napájecího napětí, není tento způsob řešení nutný, a proto může ada_server iniciovat spojení a posílat data asynchronně. Tento způsob komunikace u stále napájených zařízeních je stěžejní, protože se jedná zejména o zařízení, která nastavují aktivní prvky domácností a nízká doba odezvy je pro tuto komunikaci důležitá.

Algoritmy inteligentní domácnosti

Aplikace iot_algorithm slouží jako prostředí a rozhraní pro různé algoritmy, které budou vykonávat výpočty nad inteligentními domácnostmi. Výpočty se budou moci provádět nad aktuálními příchozími daty nebo nad historickými daty uloženými v databázi. Tato aplikace se skládá ze dvou částí, a to takzvaný framework, který komunikuje s ada_serverem nebo databází a z obsluhovaných algoritmů. Algoritmy jsou zásuvné moduly frameworku a uživatel si je může spouštět a nastavovat skrz svůj mobilní telefon nebo jiné zařízení prezentační vrstvy. Algoritmy se dělí na dva druhy. Prvním jsou algoritmy, které jsou přístupné každému uživateli zvlášť a každý uživatel jej ovládá a nastavuje podle sebe. Druhým typem algoritmů jsou takové, které se nastavují na adaptéru, tedy pro každou inteligentní domácnost a všichni uživatelé této domácnosti mají přístup k nastavení těchto algoritmů.

Knihovna notifikací mobilních telefonů

Notificator je knihovna , kterou mohou využívat aplikace na serveru pro komunikaci s mobilními telefony pomocí takzvaných push notifikací. Jedná se o způsob komunikace, který neiniciuje klient, ale server. Push notifikace umožňují okamžité doručení zpráv na mobilní telefony. Protože se u mobilních zařízení dbá na jejich spotřebu, není vhodné, aby každý poskytovatel aplikace vytvářel s mobilním telefonem trvalé spojení nebo aby se mobilní aplikace pravidelně dotazovala na server, zdali není přítomna nová notifikace Tyto notifikační služby jsou přímo implementované poskytovateli operačních systémů, kteří tímto přístupem zamezí přetěžování mobilu. Knihovna využívá služeb poskytovatelů a nabízí jednotné API, které zapouzdřuje odlišné služby různých poskytovatelů. API je pak využíváno ostatními aplikacemi serveru pro okamžitou komunikaci s mobilním telefonem.

Webové rozhraní

Webové stránky tohoto projektu budou sloužit pro propagaci a prezentaci vyvíjeného systému inteligentních domácností. Uživatel je bude moci používat pro registraci, a po přihlášení uživatele budou sloužit i pro ovládání a nastavování inteligentní domácnosti.

Gamifikace

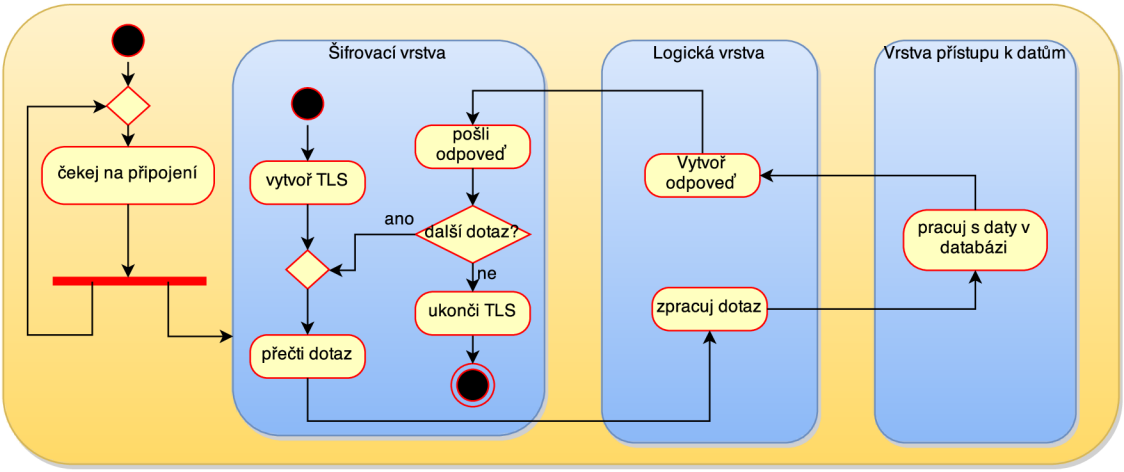
Gamifikační aplikace achievements monitoruje uživatelovu aktivitu a zaznamenává jeho dílčí úspěchy v práci s inteligentní domácností. Uživatel různými akcemi jako může být pravidelné používání aplikace, spouštění různých algoritmů, nákup nových senzorů nebo i jinými akcemi získává virtuální odměnu, se kterou se může porovnávat se svými kamarády, přáteli, spolubydlícími nebo se chlubit na sociálních sítích. Kromě odměny bude moci získávat i virtuální peníze, za které si bude moci pořídit některé prémiové služby. Celý systém gamifikace slouží pro uživatele jako zábavný průvodce aplikací, který motivuje uživatele nejen k aktivnímu používání aplikace a vyzkoušení si všech ovládacích prvků, ale i k nákupu dalších zařízení do domácnosti.

Komunikace s prezentační vrstvou

Ui_server slouží zejména pro komunikaci s prezentační vrstvou a poskytuje ji všechna potřebná data pro zobrazování stavu a nastavení inteligentní domácnosti uživateli. Komunikace je šifrována a se serverem probíhá přes protokol typu dotaz–odpověď. Prezentační zařízení pošle zprávu ve formátu XML a server poté adekvátně odpoví.

6.1 Serverová aplikace

V rámci této práce byla implementována serverová aplikace ui_server, která obsluhuje požadavky aplikací prezentační vrstvy. Tato aplikace neuchovává žádné stavové informace a jediná perzistentní vrstva je databáze.



Obrázek 6.2: Diagram aktivity aplikace ui_server

Aktivita ui_serveru je zjednodušeně znázorněna na obrázku 6.2. Zařízení prezentační vrstvy (klienti) mohou s aplikací ui_server komunikovat přes socket. Při připojení nového zařízení se vytvoří nové vlákno, které začne zařízení obsluhovat, nejdřív zabezpečí komunikaci pomocí TLS a následně přijme klientův požadavek. Ověří se zdali má klient pravomoc k zaslaní svého požadavku a jedná-li se o práva na daném adaptéru a s ním spárovanými koncovými zařízeními. Pokud klient tyto práva má, začne se požadavek vykonávat. Přistoupí se k databázi, do které se data buď vloží nebo přečtou a pošlou zpět klientovi. Jelikož při vytváření zabezpečené komunikace si musí klient vyměnit několik zpráv, je tato rutina poměrně časově náročná. Z tohoto důvodu server neuzavírá spojení okamžitě po odeslání odpovědi, ale chvíli počká a zkontroluje, zdali klient nemá zájem poslat další dotaz. Pokud nemá, zabezpečené spojení se uzavře a vlákno se ukončí.

Bezpečnostní vrstva

Bezpečnostní vrstva zajišťuje šifrovanou komunikaci mezi serverem a prezentačními zařízeními. V této vrstvě se používá knihovna OpenSSL, pomocí které se vytvoří SSL kontext,

jehož nastavení se skládá z privátního a veřejného klíče šifrované komunikace a metody šifrovaného přenosu dat.

Jakmile je šifrování nastaveno, čeká tato vrstva na připojení zařízení prezentační vrstvy na předem zvoleném portu. Jakmile se zařízení připojí, začne si server domlouvat s protistranou způsob zabezpečení. Potom, co se komunikace zabezpečí, může po tomto kanálu začít probíhat komunikace aplikační vrstvy a server začne přijímat zprávu od prezentačního zařízení.

Komunikace přes socket je uzpůsobena tak, že server nemůže vědět, jak dlouhá je příchozí zpráva a musí být definován buď koncový symbol, který se ve zprávě nikde jinde nevyskytuje a označí konec zprávy, nebo se na začátku zprávy pošle informace o její délce. Protože komunikace používá formát XML, je používána první varianta, kde koncovým symbolem je definovaný element XML, který zahrnuje veškerou komunikaci.

Šifrovaná komunikace mezi serverem a zařízeními prezentační vrstvy je realizována pomocí knihovny OpenSSL, která je open source implementace protokolů SSL a TLS. Základ knihovny, který poskytuje kryptografické funkce je napsán v jazyce C. Knihovna však lze používat i v jiných programovacích jazycích pomocí speciálních adaptérů.

OpenSSL podporuje mnoho různých kryptografických algoritmů, jedná se o šifry: Blowfish, Camellia, DES, RC2, RC4, RC5, IDEA a AES. Kryptografické hashovací funkce: MD5, MD2, SHA a MDC-2. A asymetrickou kryptografii: RSA, DSA, Diffie-Hellman a eliptické křivky. Ui_server používá tuto knihovnu pro šifrovanou komunikaci, která je realizována pomocí certifikátů, které jsou uloženy na serveru. Pomocí privátního klíče dešifruje příchozí zprávy zařízení prezentační vrstvy a šifruje zprávy, které zpět odesílá na tato zařízení.[16]

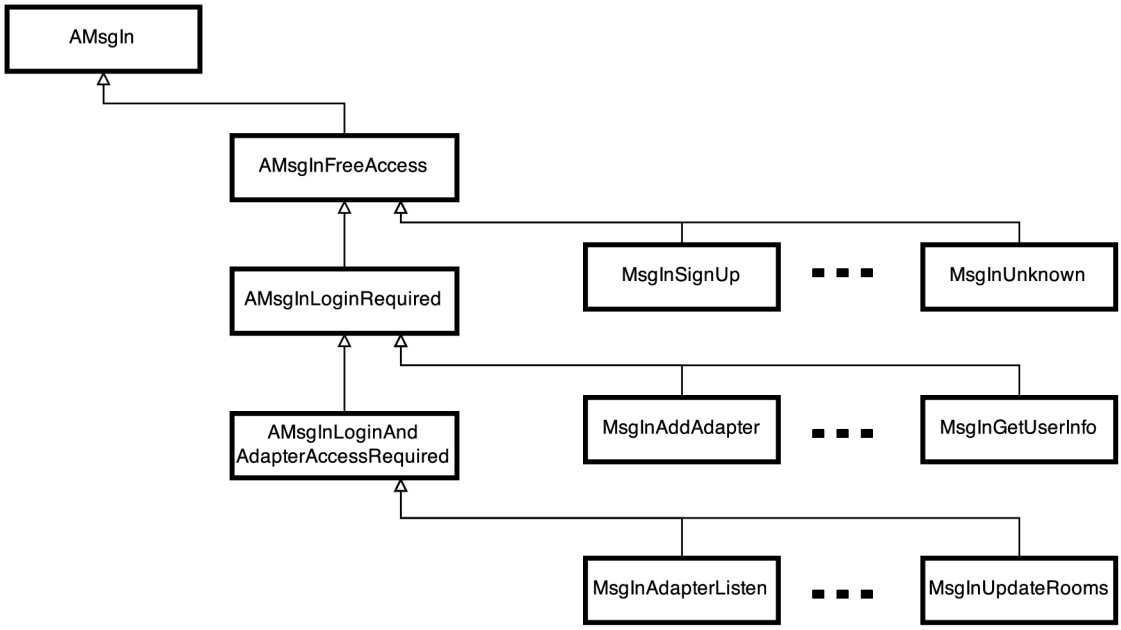
Logická vrstva

Hlavní aplikační logika v této vrstvě zpracovává data, která přišla z bezpečnostní vrstvy. Podle definovaného protokolu vykonává operace nad databází a vrací odpověď. Jestliže jsou příchozí zprávy určeny pro jiné moduly na serveru, přeposílá je dále do jiných aplikací na serveru.

Zpracování příchozích zpráv ve formátu XML probíhá pomocí knihovny pugixml. Jedná se o jednoduchou knihovnu zpracovávající XML dokumenty. Pugixml zpracovává dokumenty pomocí rozhraní podobnému DOM s bohatými rozšířeními. Vstupní soubor zpracuje rychlý XML parser, který vytvoří DOM strom, nad kterým se dále pracuje. Pro složitější dokumenty umožňuje použít implementaci XPath 1.0, která může usnadnit zpracování. Knihovna dále plně podporuje formát Unicode a umožňuje automatické konverze.[10]

Podle hlavičky příchozí zprávy se rozhodne o jaký typ zprávy se jedná a vytvoří se objekt, který odpovídá funkci přijaté zprávy. Objekt zpracovává přijatou zprávu na několika úrovních. Prvně se zkontroluje, jestli odpovídá verze komunikačního protokolu, dále se provede kontrola, zdali má uživatel právo na provedení zadané operace.

Kontroly oprávnění se provádí ve dvou fázích, první fáze kontroluje, jestli uživatel musí být k provedení akce přihlášen nebo musí mít i přístup k adaptéru. Tato fáze je v aplikační logice implementována na úrovní hierarchie abstraktních tříd, kterou lze vidět na diagramu tříd 6.3. Tato hierarchie zahrnuje čtyři úrovně příchozích zpráv. Nejvyšší úroveň je pouze vzorem pro ostatní zprávy. Druhá úroveň nekontroluje autentizaci uživatele a slouží pro zprávy jako je registrace. Třetí úroveň vyžaduje autentizaci uživatele na úrovni komunikačního řetězce, který identifikuje sezení mezi serverem a klientem. Tento druh ověření obvykle odpovídá zprávám, které pracují s daty pouze daného uživatele. Vyšší oprávnění je realizováno poslední úrovní, ve které se provádí i druhá fáze kontroly, ta kontroluje i stupeň



Obrázek 6.3: Diagram tříd

oprávnění nad daným adaptérem.

Zprávy registrace a přihlášení umožňují autentizaci uživatele i pomocí služeb třetích stran jako je Google nebo Facebook. Pro komunikaci s poskytovateli externí autentizace uživatele je použita knihovna libcurl. Libcurl poskytuje funkce pro zisk dokumentů nebo souborů ze serveru.[12]

Pro potřeby serverové aplikace a komunikace s poskytovateli externí autentizace uživatele stačí zabezpečená komunikace pomocí protokolu HTTPS. Většina poskytovatelů nabízí ověření autentizace pomocí webového rozhraní, které na základě tajemství uživatele zpřístupní některé jeho data. Knihovna dokáže korektně zpracovat hlavičku protokolu HTTP a usnadňuje práci s odpovědí poskytovatele.

Poskytovatelé externí autentizace uživatele posílají data o uživatelích ve formátu JSON. Tato uživatelská data se na ui_serveru zpracovávají pomocí knihovny Jansson, což je knihovna pro kódování, dekódování a zpracovávání dat ve formátu JSON napsaná v jazyce C. Nabízí jednoduché a intuitivní API, není závislá na jiných knihovnách a podporuje kódování Unicode.[11]

Vrstva přístupu k datům

Vrstva pro přístup k datům v databázi je složena z objektů DAO(database access object), které slouží k oddělení aplikační logiky od dat nacházejících se v databázi. Tímto přístupem DAO objekty poskytují abstraktní rozhraní pro přístup k datům, aniž by aplikační logika věděla jakým způsobem jsou data uložena. DAO objekty jsou implementovány podle vzoru singleton a zapouzdřují v sobě funkci fondu spojení a tím snižují časovou náročnost na provedení operace nad databází.

Server inteligentní domácnosti využívá pro komunikaci protokol XML, a proto existuje možnost využívat tyto objekty i pro serializaci dat do formátu XML. Dalšími možnostmi přístupu k datům z aplikační vrstvy je zisk datových struktur, které odpovídají n-ticím v databázi nebo zisk dat v elementárních datových typech.

Pro přístup a ovládání databáze se v aplikaci ui_server používá knihovna SOCI, jedná se o knihovnu pro manipulaci a přístup k datům nacházejících se v databázích pomocí jazyka C++. Díky této knihovně se mohou vytvářet SQL dotazy v C++ kódu pomocí řetězců, které se posílají jako dotazy do databáze.

I přesto, že SOCI je hlavně C++ knihovna, tak umožňuje použití přes jiné programovací jazyky. SOCI podporuje větší okruh databází, mezi které patří: DB2, Firebird, MySQL, ODBC, Oracle, PostgreSQL, SQLite.[18]

Knihovna poskytuje dobře použitelné rozhraní využívající principů objektového programování, podporuje techniky jako je ORM(Object-relational mapping), který zpříjemňuje uživateli používání databáze tím, že dokáže mapovat řádek v tabulce na objekt nebo strukturu v jazyce C++ definovanou programátorem. K této struktuře se může naprogramovat další funkcionalitu atd.

Další vlastností knihovny je přítomnost fondu spojení (connection pool), který umožňuje znovu-používání již navázaných sezení k dané databázi a tím šetří zdroje. Jedná se o přístup, kdy aplikace nenavazuje sezení vždy, když chce provést dotaz, ale vytvoří fond, který obsahuje sadu sezení. Sezení ve fondu se používají pro vykonání dotazu, a po jeho ukončení se nechá sezení otevřené a navrátí se do fondu, ze kterého může být znovu použito. Tímto se šetří nejen zdroje, ale i snižuje čas, který uplyne od vznesení požadavku k jeho ukončení. Použití knihovny je ukázáno na zdrojovém kódu 6.1 [18], kde je znázorněno vytvoření fondu sezení o velikosti deset, a následné navázání sezení s databází `mydb`. Druhá část kódu ukazuje poslání dotazu SELECT do databáze. Nejdříve se z fondu odebere jedno sezení a nad ním se vykoná dotaz, pokud skončí platnost objektu sezení, je sezení automaticky navráceno zpět do fondu.

```
// 1:Vytvoreni fondu pripojeni

const size_t poolSize = 10;
connection_pool pool(poolSize);

for (size_t i = 0; i != poolSize; ++i)
{
session & sql = pool.at(i);
sql.open("postgresql://dbname=mydb");
}

// 2:Pouziti pripojeni z~fondu

{
session sql(pool);

int count;
int id = ...;
string name;
```

```
int salary;

sql << "select name, salary from persons where id = :person_id" ,
use("person_id",id),
into(name), into(salary);

} // Pripojeni je vraceno do fondu automaticky
```

Zdrojový kód 6.1: Ukázka použití knihovny SOCI

Podpora kontinuální integrace

Protože je ui_server součástí většího systému, na kterém pracuje větší počet vývojářů, je dobré, aby byl vývoj částečně řízen některým z nástrojů kontinuální integrace, která dokáže některé chyby včas odhalit a upozornit na ně vývojáře.

Jako nástroj pro kontinuální integraci vývoje inteligentní domácnosti na fakultě informačních technologií se používá Jenkins, což je open source nástroj, používaný pro kontinuální integraci softwarových projektů, napsaný v Javě. Jenkins pozoruje spouštění opakovaných procesů, jako je například překlad, testování nebo balíčkování softwaru. Jenkins se zejména soustředí na dva procesy. Prvním je průběžný překlad a testování, čímž usnadňuje integraci změn projektu a umožňuje uživatelům pohodlněji získávat nové verze. Druhým procesem je sledování provádění externě spouštěných procesů, jako jsou úlohy spouštěné cronem nebo službou procmail, a to i v případě, že běží na vzdálených strojích.[14]

Vývojové prostředí

Pro vývoj ui_serveru bylo použito integrované vývojové prostředí Netbeans, které je sice primárně určeno pro programovací jazyk Java, ale díky dalším přídavným modulům jej lze použít i pro vývoj jazyků PHP, C, C++ nebo webových technologií HTML, CSS, JavaScript. Pro vývoj aplikace ui_server je využita pouze část pro C++. Další zásuvné moduly mohou rozšířit funkcionalitu prostředí o modelovaní v UML, refaktoring, poskytování přístupu a obsluhu repozitářů jako jsou SVN, Git nebo Mercurial. [17]

Aplikace ui_server je poměrně rozsáhlá a obsahuje mnoho tříd. Proto je použití takového vývojového prostředí ulehčení práce, zvýšení efektivity práce a zlepšení přehlednosti.

Kapitola 7

Optimalizace

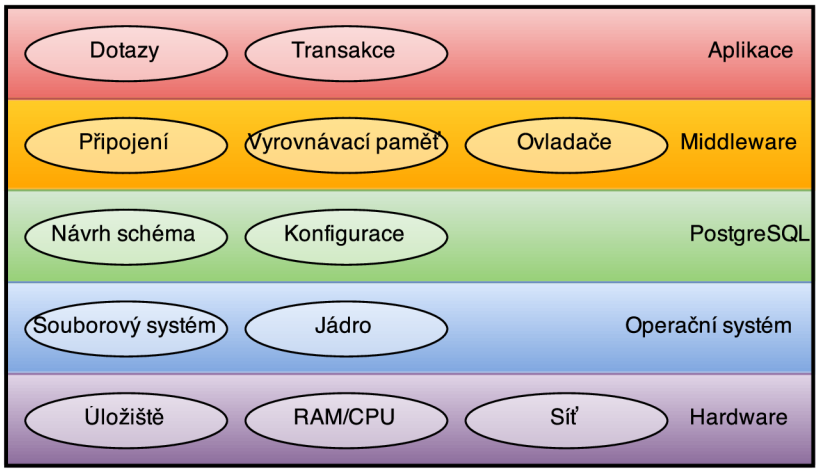
Před použitím databáze je vhodné se zamyslet, jakým způsobem bude databáze využívána a jaké dotazy se budou provádět nejčastěji. Snažit se identifikovat úzká hrdla, která by mohla systém omezovat a snažit se je odstranit vhodným nastavením, změnou hardwaru nebo jinými úpravami. Podle stroje, na kterém bude databáze spuštěna a na způsobu užívání, se volí vhodná konfigurace databáze. Některé dotazy se dají urychlit použitím vhodně nastavených indexů. Rychlost přístupu k datům se dá ovlivnit způsobem uložení datových souborů na disku atd.

V této kapitole je popsáno několik úrovní optimalizace produkčního systému a jsou podrobněji popsány metody optimalizace databáze, které mohou zvýšit její výkon.

7.1 Optimalizace rychlosti databáze

Zrychlení přístupu k datům v databázi se dá přistoupit na různých úrovních. Na obrázku 7.1 jsou úrovně optimalizací, které uvedl Josh Berkus ve své prezentaci „Five steps for PostgreSQL performace“[4]. V této prezentaci rozděluje optimalizaci celého systému do několika vrstev:

- Hardware - Jedná se o fyzické prvky ovlivňující rychlost práce s databází. Vylepšení na tomto stupni zahrnuje výměnu procesoru za rychlejší, zvětšení velikosti operační paměti nebo zvýšení výkonu úložiště pomocí rychlejších disků, technologií RAID, NAS, SAN apod.
- Operační systém a souborový systém - Na této úrovni je možno zvýšit výkon použitím vhodnějšího souborového systému, výběru vstupně/výstupního plánovače nebo nastavením jádra, kdy například zvolíme vyšší sdílenou paměť(`shmmax, shmall`).
- Databáze - Nastavení databáze se provádí zejména pomocí konfiguračních souborů, které ovlivňují různé aspekty chování databáze nebo připojených klientů.
- Middleware - Přidaná funkčnost k databázi, mezi kterou může patřit fond připojení, další vrstva vyrovnávací paměti, partitioning a další.
- Aplikace - Optimalizace, které záleží na využívání databáze aplikací, která s ní pracuje. Jedná se o vhodně navržené schéma databáze, indexování, optimalizace napsaných dotazů nebo používání předpracovaných dotazů.



Obrázek 7.1: Vrstvy optimalizace produkčního systému

7.2 Optimalizace databáze

Před začátkem používání databáze je vhodné upravit některé proměnné, které ovlivňují její vlastnosti a výkon. Databáze PostgreSQL umožňuje upravit svou konfiguraci pomocí souboru postgresql.conf, ten se rozděluje na několik bloků. Základní z nich jsou nastavení připojení klientů a jejich případnou autentizaci, logování, využití operační paměti a disku, WAL (Write-Ahead Logging) a vakuování.

Zde budu popisovat možnosti nastavení konfiguračního souboru a jejich doporučované hodnoty, které byly čerpány z oficiálních stránek databáze PostgreSQL[15].

Prvním důležitým nastavením je možnost připojení k databázi, ta ve výchozím nastavení povoluje připojení pouze z lokálního počítače. Pokud je žádoucí, aby se k databázi připojovalo i vzdáleně, povolí se připojení z žádoucích adres.

Velikost vyrovnávací paměti databáze se doporučuje nastavit na 25% dostupné operační paměti. V této paměti se ukládají data, ke kterým databáze často přistupuje a tím zmenšuje čas potřebný k přečtení. Ve vyrovnávací paměti se můžou u menších databází držet celé tabulky, ale u větších databází se nejčastěji uchovávají pouze indexy. Pro lepší výběr prováděcího plánu dotazů se nastavuje parametr, který je založen na odhadu volné paměti pro vyrovnávací paměť souborového systému. Doporučuje se nastavit podle vzorce $TRAM - (DRAM - PGRAM)$, kde $TRAM$ je celková paměť, $DRAM$ je paměť potřebná operačním systémem a aplikacemi a $PGRAM$ je paměť potřebná pro PostgreSQL.

Velikost paměťového prostoru každého klienta využívaného pro řazení dotazovaných dat se nastavuje podle používaných dotazů. Pokud je tato hodnota malá, musí řazení probíhat na disku a značně se zpomalí.

Operace údržby databáze jako jsou vytvoření indexu, cizího klíče nebo vakuování potřebují ke své funkci jistou část paměti. Nastavení větší části paměti pro tyto úlohy vede k jejich urychlení.

PostgreSQL zapisuje data nejprve do rychlého logu a po nějakém čase do datových souborů, této události se říká kontrolní bod(checkpoint). Kontrolní bod se skládá z několika WAL segmentů, jehož počet i velikost se dá nastavit. Aby se zápisy těchto logů do datových souborů rozptýlily v čase a tím se snížilo nárazové zatížení disku, se používá parametr udá-

vající rozložení zápisu v čase před dalším kontrolním bodem. Okamžitý zápis dat na disk ovlivňuje parametr synchronního commitu, který při zapnutí zaručuje, že všechny provedené transakce nemohou být ztraceny při výpadku. Vypnutí tohoto nastavení vede k použití asynchronního commitu, který zrychlí systém, ale při pádu databáze vede ke ztrátám dat posledních transakcí. Přesto garantuje konzistenci databáze při pádu a zápis dat na disk nejpozději do trojnásobku maximálního zpoždění zápisu, které lze taky nastavit. Ve výchozím nastavení se jedná 600 milisekund. U některých aplikací může být tento kompromis ztráty posledních dat akceptovatelný.

Nastavení indexů

Dalším nastavením databáze jsou indexy, které slouží pro rychlejší vyhledávání dat. Bez indexu musí totiž databáze procházet všechna data v tabulce sekvenčně a to je velice neefektivní a časově náročné.

Použitím indexu se v databázi vytvoří pomocné vyhledávací struktury jako je hash, B-strom, GiST, SP-GiST, GIN a jiné. Každá z těchto vyhledávacích struktur je vhodná na jiný typ dat, protože k vyhledávání používají jiné algoritmy. Tyto vyhledávací struktury nemění samotná základní data, ale ovlivňují práci databáze, která dokáže data rychleji dohledat. Samotné struktury však zabírají další prostor na disku a v případě vkládání dat nebo aktualizace sloupců, na kterých je index postaven se musí kromě tohoto uložení aktualizovat i struktura indexů, což zvyšuje časovou náročnost.

Shlukování

Shlukování(Clustering) je metoda, při které se data fyzicky uspořádají na disku do stanovených posloupností a tím se zvýší efektivita při čtení dat z disku. Tato metoda je vhodná pokud se používá na tabulky, u kterých se vyhledává větší počet řádků podle intervalu hodnot některého z atributů.

Dosáhnutí shlukovaných tabulek lze pomocí takzvaného shlukovaného indexu, který říká databázi, aby ukládala data fyzicky na disk seřazena ve stejném pořadí jako je definovaný index. Tato vlastnost vynucuje použití maximálně jednoho shlukovaného indexu na tabulku, protože můžeme vynutit uchovávání dat pouze v jednom pořadí. Tento index zvyšuje efektivitu čtení většího množství řádků s použitím operátorů jako jsou BETWEEN, <, >=, <, a <=. Jakmile se pomocí indexu najde první hodnota, je garantováno, že následující hodnoty budou fyzicky sousedit a proto může databáze data procházet sekvenčně. Index však zároveň může zpomalovat zápis, protože vklad nových dat může vynutit fyzické přeuspořádání dat. Databáze PostgreSQL bohužel nepodporuje shlukovaný index, ale přeuspořádání dat lze dosáhnout pomocí příkazu CLUSTER nebo jiných nástrojů.

Tuto metodu je vhodné použít i u databáze inteligentní domácnosti zejména u tabulky historických hodnot, která je používána téměř výhradně pro výběr dat určitého časového intervalu na daném senzoru nebo aktivním prvku. Data budou většinou vybírána po větších sadách a metoda by měla zvýšit její rychlost. Zároveň se očekává, že příchozí data budou vkládána podle času pořízení a tím nebudou zpomalovat vklady dat v případě, že by se použil shlukovaný index.

Partitioning

Partitioning je technika logického nebo fyzického rozdělení dat na oddíly, ke kterým se dá lépe přistupovat a udržovat je. Rozdělování dostatečně velkých tabulek databáze na více oddílů vede ke zvýšení výkonu databáze díky možnosti práce pouze s části dat. Databázové dotazy se mohou provádět pouze nad relevantními oddíly a tím se zmenší počet přístupových operací k datům. Operace správy, jako vytváření indexů nebo vytváření zálohy, se mohou provádět pouze nad daným oddílem a budou stát databázi méně času i prostředků.[22] Způsob partitioningu se rozděluje na:

- Horizontální partitioning - Rozděluje oddíly podle řádků. Tento způsob je vhodný u rozdělení například na často čtená nová data a na starší data, ke kterým je přistupováno zřídka. Oddíly novějších dat lze pak ukládat na rychlejší disk a tím zrychlovat doby přístupu.

- Vertikální partitioning - Rozděluje oddíly podle sloupců. Obdobně jako u horizontálního, jenom se data rozdělují například podle četnosti používání dat v jednotlivých sloupcích.

- Statický partitioning - Počet oddílů je předem znám a nemění se. U tohoto způsobu rozdělování je vhodné jej zvolit způsob tak, aby byly všechny oddíly stejně využity, lze například použít vhodná hashovací funkce.

- Dynamický partitioning - Počet oddílů se s během databáze mění. Používá se v případech, kdy se data rozdělují podle rostoucího atributu. Podle něj se provádí dělení, a v případě překročení určité hranice se vytvoří nový oddíl.

Partitioning se v objektově-relačních databázích často provádí pomocí dědičnosti mezi tabulkami. Vytvoří se základní tabulka a další oddíly z ní dědí a díky tomu mají stejnou datovou strukturu jako základní tabulka. Tento přístup dovoluje provádět dotazy typu SELECT a UPDATE na základní tabulce a dědičnost se již postará o vhodný výběr dat z jednotlivých oddílů.

Pro vkládání nových dat nelze dědičnost využít, protože základní tabulka nemůže určit kam má daná data uložit a podle čeho se rozlišuje, kam mají být data vložena. Tento problém se dá vyřešit více přístupy: pomocí triggeru s dynamickými SQL dotazy, pomocí triggeru bez dynamických SQL dotazů, pomocí pravidel a vkládání do správných oddílu na aplikační úrovni. Jednotlivé metody se liší svými vlastnostmi i rychlostmi při používání.[2]

Triggery s dynamickými SQL dotazy vyhodnotí v době začátku vkladu do jakého oddílu se mají data umístit a vloží do něj data, původní vklad do základní tabulky nebude proveden. Rozdíl v použití triggeru bez dynamického SQL je v tom, že místo dynamického vyhodnocení se vytvoří řada explicitních dotazů INSERT do adekvátních oddílů. Takový trigger má pak počet podmínek roven počtu oddílů, do kterých je možno vkládat.

V případě používání pravidel se pro každý nový oddíl vytvoří nové pravidlo, které přesměrovává data ze základní tabulky do patřičného oddílu. Pravidla jsou založena na přepisování vyhodnocovacího plánu v době, kdy se klient snaží vložit data do základní tabulky. Pravidla se vytvářejí způsobem jaký je znázorněn ve zdrojovém kódu 7.1. Výhodou oproti triggerům je, že se nespouští žádné procedury, ale plán dotazu je přepsán ještě v době plánování.

```
CREATE OR REPLACE RULE rule_p0 AS ON INSERT TO table_base
        WHERE (id > 0 AND id <= 86400)
        DO INSTEAD INSERT INTO table_p0 VALUES (NEW.*);
```

Zdrojový kód 7.1: Vytvoření pravidla pro partitioning

Řízení partitioningu již v aplikaci vede ke zvýšení rychlosti, protože se nemusí provádět žádná kontrola na úrovní databáze, nemusí se kontrolovat pravidla ani spouštět speciální triggery, ale zvyšují se tím nároky na samotnou aplikaci a používání databáze již není tak jednoduché a transparentní.

Kapitola 8

Výsledky

Dalším krokem, po navržení databázového schématu, je testování výkonu databáze a případná optimalizace. V této kapitole je popsána analýza předpokládaného využití celého systému a testovací scénář. Je ukázán vliv optimalizací na databázi a vyhodnocení měření.

8.1 Analýza vytížení databáze

Databáze je vytěžována aplikacemi na serveru, které ji využívají pro uchování dat. Jedná se o aplikace sběru naměřených dat, komunikace s prezentačními zařízeními, správy algoritmů a gamifikační aplikace.

Zatížení vzniklé ukládáním naměřených dat není tolik závislé na počtu připojených adaptérů, ale spíše na počtu připojených senzorů, které pravidelně skrz adaptéry posílají naměřená data. Čas pravidelného zasílání je zde pro vytížení databáze velice důležitý. Druhými připojenými prvky k adaptéru jsou aktory, u kterých se však nepředpokládá vyšší zatížení vůči databázi, protože svá data zasílají pouze v případě změny svého stavu. Což v případě nejčastějšího použití na ovládání světel nebo zásuvek nebude příliš časté.

Komunikace s prezentačními zařízeními vytěžuje databázi v momentu, kdy uživatel na prezentačním zařízení pracuje. Jedná se především o rutinní dotazy jako přihlášení, zisk seznamu dostupných adaptérů, zisk koncových zařízení, čidel a aktivních prvků společně s poslední naměřenou hodnotou apod. Tyto operace však budou z databáze číst menší množství dat z relativně malých tabulek. Větším vytížením bude posílání historických dat, které se v prezentačním zařízení budou zobrazovat do grafů. Tento typ dotazů již není tak častým, ale jedná se o dotazy na velké množství položek do tabulky historických hodnot, která zabírá naprostou většinu databáze. Efektivní výběr historických dat z databáze bude klíčový pro plynulou práci uživatele s prezentačním zařízením. Pokud se aplikace prezentačních zařízení dotazuje na historická data, většinou posílá požadavek na denní, týdenní a měsíční statistiky.

Z testovacího provozu systému inteligentní domácnosti vyplývá, že nejčastější dotaz je na aktuální hodnotu, několika málo zařízení, v pravidelných intervalech, který vytvářejí takzvané widgety v uživatelově mobilu. Dále bylo zjištěno, že uživatel aplikaci používá jenom párkrát denně a na grafy historických domácností se dívá jen zřídka, jednou až třikrát týdně. Ukázalo se, že uživatelé nemění nastavení spánku senzoru a využívají přednastavený pětiminutový interval. Tato perioda je více než dostatečná pro jemnost zobrazení dat v grafu, zároveň příliš nezatěžuje baterii zařízení a uživatel ji nemusí často vyměňovat.

Druhy domácností

Pro účely testování je vhodné odhadnout, jakým způsobem bude celý systém pracovat a s jakými problémy se může potýkat. Pro zjednodušení reálného stavu byly definovány tři druhy domácností, které se mohou vyskytovat v našem systému. Druhy domácností, počty jejich uživatelů, čidel a aktivních prvků jsou ukázány v tabulce 8.1.

Typ domácnosti	**#uživatelů**	**#zařízení**	**#senzorů**	**#aktivních prvků**
Začínající domácnost	1	2	4	0
Pokročilá domácnost	2	5	12	2
Plně vybavená domácnost	4	18	18	20

Tabulka 8.1: Typy domácností

Tyto typy domácnosti se budou v systému vyskytovat proměnlivým zastoupením v závislosti na čase života inteligentní domácnosti. Dá se očekávat, že ze začátku budou mít největší zastoupení začínající domácnosti a časem, až uživatelé získají důvěru v dané řešení, lze předpokládat, že počty zařízení v jejich domácnostech budou narůstat, stejně jako poměr pokročilých domácností v systému. Odhaduje se, že plně vybavené domácnosti budou zabírat nejmenší podíl, ale jejich zastoupení bude mít postupný a mírný nárůst.

8.2 Testovací konfigurace

Testování a práce s databází probíhala vzdáleně na stroji ant-1 na Fakultě informačních technologií. Konfigurace stroje ant-1:

- Operační systém: centos 5.3 (kernel 2.6.18-194.26.1.el5)
- Procesor: Intel Xeon CPU E5410 2.33GHz
- Operační paměť: 12GB
- Disk: Seagate Barracuda ES.2 - 500GB ST3500320NS
- Databáze: PostgreSQL 9.3

Testování výkonu databáze bylo prováděno dvěma způsoby. Pomocí programu pgbench, který byl použit na testování maximálního počtu transakcí za sekundu. A pomocí vytvořeného programu, který simuloval chod systému inteligentní domácnosti.

Inteligentní domácnost je nejvíce zatěžována připojenými adaptéry a uživatelskými prezentačními zařízeními, a proto i program simulace systému inteligentní domácnosti vytváří tyto dva zdroje zátěže současně. Zátěž generována připojenými adaptéry je zejména tvořena senzorickými daty, které musí databáze ukládat. Množství uložených dat je kromě počtu připojených senzorů ovlivňována intervalem spánku. Protože nejčastější nastavení intervalu spánku je pět minut, je v simulaci použit stejně dlouhý interval. Druhý typ zátěže vytvářejí uživatelská prezentační zařízení, které přistupují k datům v databázi pomocí serverové aplikace. Tento druh zátěže je vytvářen pomocí náhodného přístupu k datům v databázi, pomoci předem připravených dotazů, které prezentační zařízení posílají. Počet

dotazů v této části simulace je ovlivňován počtem uživatelů v systému. Průměrný uživatel kontroluje domácnost jednou denně a na grafy senzorických dat se kouká dvakrát týdně.

Program pgbench řídí spouštění výkonnostních testů na databázi PostgreSQL. Je instalován společně s balíčkem databáze PostgreSQL a nabízí testování pomocí předpřipravených testů, které využívají předem navržené tabulky. Druhým způsobem testování pomocí pgbench je použitím uživateli napsaných skriptů, které nabízí širší možnosti. Při spuštění začne pgbench spouštět SQL dotazy a testuje počty provedených transakcí za sekundu. Umožňuje upravovat nastavení jako je počet současných databázových připojení, počet vláken, ve kterých se spouští SQL skripty, zdali se budou spouštět pouze dotazy typu SELECT a další.

Při použití základních testů, které nabízí pgbench se použije schéma databáze, které má čtyři tabulky: branch, teller, account a history. Tyto tabulky slouží k testování výkonu na příkladu bankovního institutu obsahující pobočky(branch), na kterých jsou úředníci(teller) a ti se starají o peníze svých klientů na založených účtech(account). Výběry a vklady peněz jsou uchovávány v tabulce historie(history).

Připravené tabulky se plní daty pomocí příkazu `pgbench -i -s <škála>`, kde `škála` určuje velikost naplněné databáze. Každá jednotka škály je rovna naplnění databáze jednou pobočkou, deseti úředníky a 100000 účty, což na disku odpovídá přibližně velikosti 16MB. Zvyšováním čísla škály můžeme dosáhnout požadované velikosti databáze.

Základní skript, který používá pgbench pro testy výkonu je znázorněn ve zdrojovém kódu 8.1 [13]. Jeho funkce je taková, že simuluje jednu bankovní transakci, kdy klient přijde na pobočku(bid) za úředníkem(tid) a chce na svém učtu(aid) provést vklad nebo výběr(delta). Jednotlivé parametry této transakce jsou nastaveny funkcí `\setrandom`, která je vybírá podle zadané škály. Po nastavení těchto parametrů se provádí SQL transakce, ve které se atomicky provádí několik dotazů, jedná se o aktualizaci zůstatku na klientově účtu, dotaz na aktuální zůstatek, aktualizaci zůstatku peněz u úředníka, na celé pobočce a poté se zaznamená tato bankovní transakce do tabulky historie současně s časovým razítkem.

Chování tohoto testu se dá ovlivnit pomocí přepínače `-N`, který nebude provádět aktualizaci zůstatku peněz u úředníka a na pobočce, nebo přepínačem `-S`, který spouští pouze dotazy typu SELECT a tím se provádí test, který se zaměřuje jenom na čtení.

```
\set nbranches :scale
\set ntellers 10 * :scale
\set naccounts 100000 * :scale
\setrandom aid 1 :naccounts
\setrandom bid 1 :nbranches
\setrandom tid 1 :ntellers
\setrandom delta -5000 5000
BEGIN;
UPDATE pgbench_accounts SET abalance = abalance + :delta
        WHERE aid = :aid;
SELECT abalance FROM pgbench_accounts WHERE aid = :aid;
UPDATE pgbench_tellers SET tbalance = tbalance + :delta
        WHERE tid = :tid;
UPDATE pgbench_branches SET bbalance = bbalance + :delta
        WHERE bid = :bid;
INSERT INTO pgbench_history (tid, bid, aid, delta, mtime)
        VALUES (:tid, :bid, :aid, :delta, CURRENT_TIMESTAMP);
END;
```

Zdrojový kód 8.1: Základní test pgbench

8.3 Testování

Serverová aplikace byla vyvíjena iteračně a byla vydávána ve dvou verzích: alfa a beta. K testování byly použity mobilní aplikace na Android a Windows Phone, které ovládali testeři. V alfa verzích se přidávala nová funkcionalita systému a byla v průběhu vývoje testována skupinou 22 vývojářů. Alfa testování odhalilo velký počet chyb, které se průběžně opravovaly a po jejich opravení byla vydána stabilní beta verze. Beta verze aplikace byly testovány pěti uživateli, kteří měli doma nainstalovány prvky inteligentní domácnosti a ovládali je pomocí svého mobilního telefonu.

8.4 Optimalizace

V této části jsou popsány optimalizace databáze pomocí konfiguračního souboru, shlukování, partitioningu a optimalizace jsou vyhodnoceny.

Nastavení databáze PostgreSQL

Databáze PostgreSQL se nastavuje pomocí konfiguračního souboru postgresql.conf, který se standardně nachází ve složce dat databáze, případně jde jeho umístění nalézt pomocí dotazu v databázi `SHOW config_file;`. Nastavení vychází ze stroje, na kterém se databáze spouští a na aplikaci, která databázi využívá. Tabulka 8.2 vypisuje nejdůležitější změny, které byly udělány oproti přednastavenému konfiguračnímu souboru.

Hodnota shared_buffers vychází z doporučeného nastavení a to 25% dostupné operační paměti, maintenance_work_mem je nastavena na vyšší hodnotu než bývá, kvůli rychlejšímu vykonání operací údržby. Vlastnost synchronního commitu synchronous_commit je v případě databáze inteligentní domácnosti vypnuta, protože se nejedná o důležitá data a jestliže dojde k výpadku, tak ztráta dat, která se vložila do databáze za posledních 600 milisekund nebude nijak kritická. Asynchronní commit zvýší výkon databáze, co se týče vkládání do ní, což je důležité vzhledem k zátěži databáze ze směru koncových zařízení, které neustále budou posílat svá data. Effective_cache_size je odvozen ze vzorce $TRAM - (DRAM - PGRAM)$ popsaného v kapitole 7.1. Paměťový prostor každého klienta work_mem je nastaveno na 8MB. Nastavení kontrolních bodů checkpoint_segments, checkpoint_completion_target a wal_buffers je nastaveno podle doporučení při neustálé zátěži databáze vkladem dat.

Nastavení	**hodnota optimalizovaného konf. souboru**	**hodnota původního konf. souboru**
shared_buffers	3GB	32MB
maintenance_work_mem	1GB	16MB
synchronous_commit	off	on
effective_cache_size	6GB	128MB
work_mem	8MB	1MB
checkpoint_segments	32	3
checkpoint_completion_target	0.9	0.5
wal_buffers	16MB	2MB

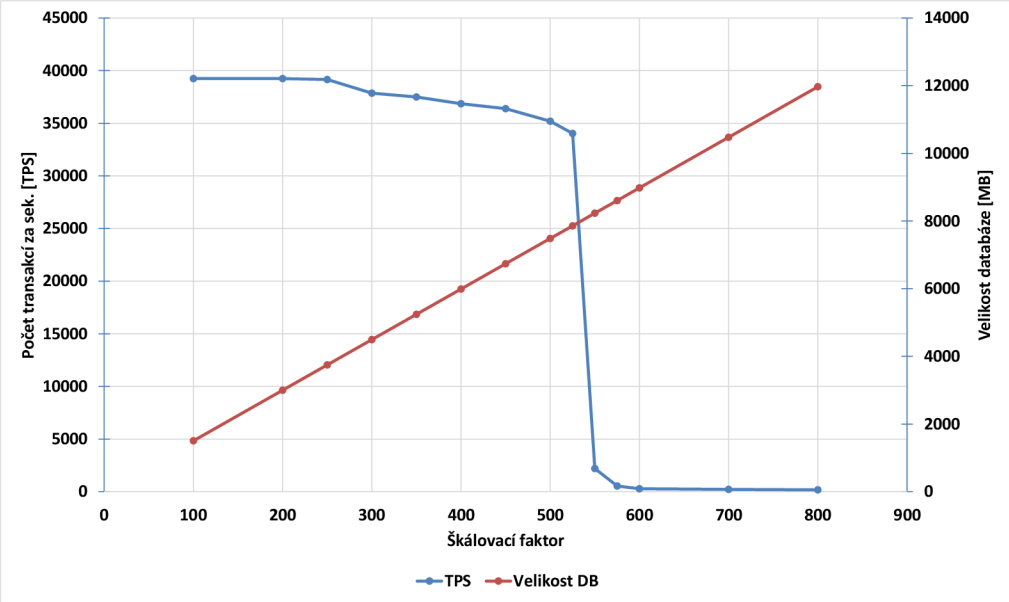
Tabulka 8.2: Změny konfiguračního souboru

Po změně konfiguračního souboru byly spuštěny testy pomocí programu pgbench, který měřil počet čtecích transakcí za sekundu.

Testování probíhalo s nastavením 16 klientů, kteří využívali 4 jádra procesoru na testovacím stroji ant-1. Testy byly spouštěny se škálou(viz 8.2) od 100 do 800 a s krokem 25 až 100 podle míry změn mezi jednotlivými kroky.

Na grafu 8.1 jde vidět vliv operační paměti na chod databáze a počty transakcí za sekundu. Pokud má databáze velikost menší než 8GB, data se uchovávají na různých úrovních vyrovnávacích pamětí systému a jejich čtení je velice rychlé, pokud databáze překročí tuto hranici velikosti, počet transakcí se rapidně sníží.

Další testy v této kapitole používají databáze, které obsahují tisíc nebo více domácností. Databáze obsahující tisíc domácností, má přibližně 13GB, takže není možné, aby byla celá uložená do operační paměti na stroji ant-1.



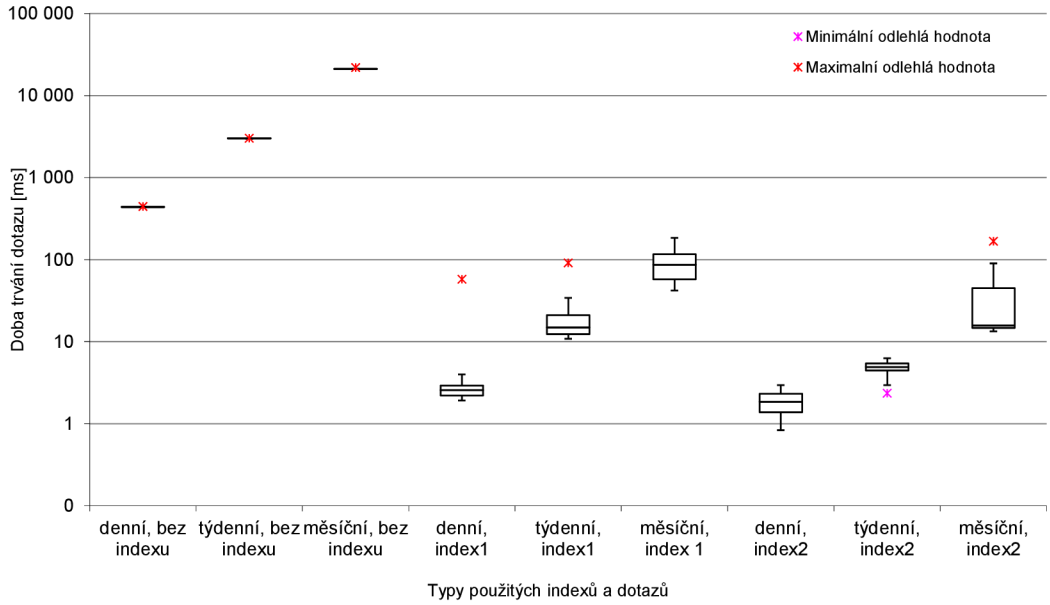
Obrázek 8.1: Graf závislosti TPS a velikosti databáze

Nastavení indexů

Indexy databáze byly nastaveny podle používaných dotazů, většinou se jednalo o dotazování přes cizí klíče. Výjimkou je tabulka users, která obsahuje další jednoznačné identifikátory od externích poskytovatelů autentizace. Tyto identifikátory jednoznačně identifikují uživatele a často se podle nich uživatelé vyhledávají, proto je zde vhodné použít unikátní index, který kromě zrychleného vyhledávání garantuje, že v databázi nemohou být dva uživatelé se stejným identifikátorem od stejného poskytovatele.

Ve verzi PostgreSQL 9.2 a výše se vyskytuje funkce vyhledávání dat nazvaná Index-Only scan, která umožňuje zvýšit výkon vyhledávání dat a to takovým způsobem, že pokud jsou vyhledávaná data obsažena rovnou v indexu, nemusí již databáze přistupovat k datovým souborům.

V krabicovém grafu 8.2 je porovnání používání dotazů bez indexu a s použitím klasického indexu, který je znázorněn v grafu jako index1 a k indexaci používá sloupce vyskytující se v dotazu SELECT za klauzulí WHERE, v případě historických dat je to mac adresa koncového zařízení, typ senzoru nebo aktivního prvku a časové razítko. V grafu je porovnání s druhým indexem, který je v grafu zapsán jako index2 a umožňuje použití novějšího prohledávání pomocí Index-Only scan. Tento graf byl měřen na databázi s jedním tisícem pokročilých domácností. Logaritmický graf porovnává tři druhy indexu na třech typech dotazu SELECT, ty se dotazují na historická data v časovém rozsahu posledního dne, týdne a měsíce. Krabicová část grafu je shora ohraničena 3. kvartilem, zespodu 1. kvartilem a mezi nimi se nachází čára představující medián. Čáry vycházející z krabice se označují jako vousy a znázorňují jakým způsobem se vyskytují data pod spodním kvartilem nebo nad horním kvartilem, jsou však omezeny 1,5 násobkem rozpětí mezi horním a spodním kvartilem, dále je na grafu znázorněna minimální a maximální doba trvání dotazu, ale to pouze v případě, že nebyla shodna s vousem.



Obrázek 8.2: Graf trvání dotazů v závislosti na typu dotazu a použitém indexu

Z grafu lze vidět, že nepoužití indexu vede k velice dlouhým dobám vyhodnocení dotazu, který v případě měsíčních historických hodnot je vyšší než dvacet sekund. Tento stav je způsoben tím, že se data hledají v tabulce sekvenčně a musí se prohledat celá tabulka. Sekvenční prohledávání má dopad na to, že všechny dotazy stejného typu mají velice podobnou dobu vyhodnocení.

Z porovnání dvou druhů indexů jde vidět, že index2 má rychlejší doby vykonání dotazů, a proto bude v databázi inteligentní domácnosti použit.

Shlukování

Metodu shlukování je vhodné využít u tabulky historických hodnot. U této tabulky dochází k dotazům na mnoho řádků v databázi a v případě, že tyto řádky budou za sebou řazeny podle stejného klíče, jakým se data vyhledávají, mohlo by dojít ke zrychlení čtení těchto dat.

Databáze PostgreSQL umožňuje shlukování pomocí příkazu CLUSTER, který fyzicky přeuspořádá data v tabulce podle vybraného indexu, který je nad tabulkou nastaven. Bohužel tento příkaz používá při provádění zámek, který znemožní provádění jakýkoliv jiných operací, což je u databáze inteligentní domácnosti nemyslitelné, protože data z koncových zařízení proudí neustále a omezovat uživatele tím, že se mu znemožní přístup k historickým hodnotám je velice nevhodné.

Alternativa k příkazu CLUSTER je přídavný modul do databáze PostgreSQL, který se jmenuje pg_repack. Ten poskytuje podobnou funkci, ale s tím rozdílem, že nepoužívá exkluzivní zamykání tabulky. Tento modul se navíc dá použít i bez definovaného indexu a způsob uložení dat na disku se dá zvolit pomocí vyjmenování sloupců v tabulce. Při spuštění vytvoří pg_repack stínovou tabulku, na kterou se začnou kopírovat data z původní tabulky takovým způsobem, aby bylo dosaženo definované umístění dat na disku. Zároveň se vytvoří trigger, který jakoukoliv zápisovou operaci na původní tabulce provede i na tabulce stínové, tím je zaručeno, že všechna data, která se vložila či modifikovala po spuštění programu pg_repack, nebudou zahozena. Po skončení kopírování do stínové tabulky se tabulky prohodí a původní tabulka je smazána.

Na krabicovém grafu 8.3, který byl měřen pomocí simulace na databázích s daty dvou tisíc pokročilých domácností, lze na ose x vidět časy dotazů na denní, týdenní a měsíční historická data senzorů. Na levé půlce jsou časy bez použití shlukování a na pravé s jeho použitím. Vliv shlukování na časy dotazů jsou minimální.

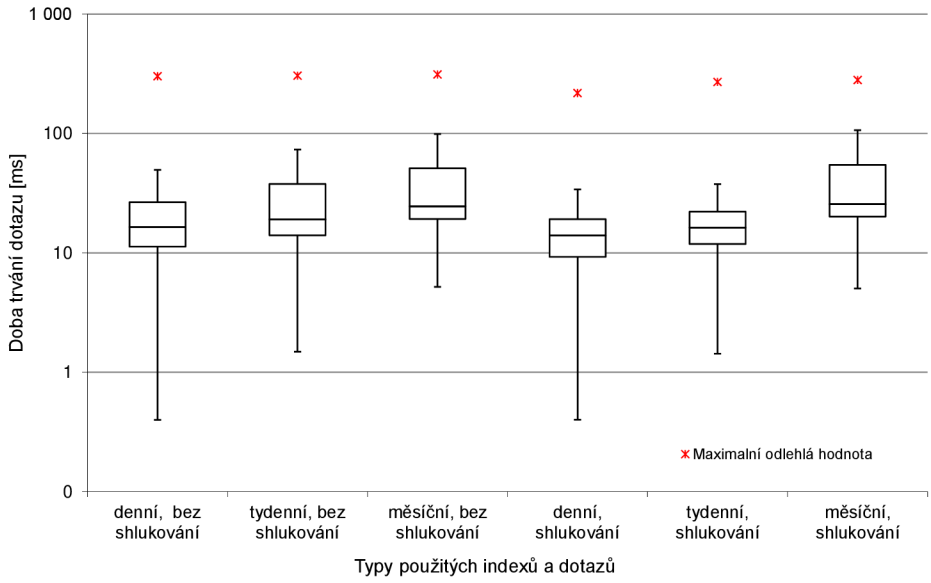
Domnívám se, že minimální vliv shlukování je způsoben použitím vhodného indexu, který používá Index-Only scan. Negativem shlukování je nutnost jej provádět pravidelně, protože nově uložená data se již neukládají uspořádaně. Provádění shlukování zvyšuje nároky na disk a to jak na jeho výkon tak i na kapacitu, protože kopíruje celou tabulku. Protože je vliv shlukování na rychlosti nepatrný nebude v databázi inteligentní domácnosti používat.

Partitioning

U databáze inteligentní domácnosti je vhodné použít horizontální, dynamický partitioning, který rozděluje tabulku historických dat na oddíly podle dnů, protože je denní objem příchozích dat ze senzorů velký a je to vhodné logické rozdělení.

Dalším důvodem použití partitioningu je nutnost starší data ze systému odstraňovat nebo přemisťovat k zálohování, pokud bychom to dělali v rámci jedné velké tabulky a odstraňovali nebo přemisťovali nejstarší data, musela by databáze přeindexovávat svá data a její výkon by rapidně klesl.

Partitioning v databázi PostgreSQL lze implementovat různými způsoby, které byly popsány v kapitole 7.2. Pro tuto práci bylo zvoleno použití přídavného modulu pg_partman, který umožňuje statický a dynamický partitioning podle číselného atributu nebo času. Pg_partman používá pro partitioning jednu základní tabulku a na základě definice této tabulky vytváří oddíly, do kterých se data přesměrovávají pomocí triggerů bez dynamického SQL.



Obrázek 8.3: Graf trvání dotazů v závislosti na typu dotazu a použití shlukování

U dynamického partitioningu v programu pg_partman se triggery nepoužívají jenom na přesměrování dat, ale obsahují i kontroly, zdali není potřeba vytvořit nový oddíl. Protože není vhodné vytvářet nový oddíl až v době nutnosti, vytváří pg_partman nové oddíly již v předstihu, když je předchozí oddíl zaplněn z poloviny. Tento přístup zaručuje, že nově příchozí data nemusí čekat, až se vytvoří nový oddíl.

Dalšími možnostmi modulu pg_partman je nastavení počtu aktivních oddílů. Jakmile je oddíl označen jako neaktivní, data z něj jdou číst, ale již nejdou vložit pomocí přesměrování přes trigger. Tento přístup šetří náročnost používání triggeru a zamezuje ukládání dat, které by se v systému již neměly vyskytovat. Volitelně se dá nastavit retenční politika, podle které se starší oddíly budou z databáze automaticky mazat.

Nastavení modulu pg_partman vychází z požadavků na databázi inteligentní domácnosti. Používá se dynamický partitioning nad tabulkou historických dat, který rozděluje data senzorů a aktivních prvků do oddílů podle čísla časového razítka. Protože je časové razítko rovno počtu sekund od půlnoci 1. 1. 1970 GMT, jsou oddíly vytvářeny s rozdílem 86400 sekund, což je počet sekund v jednom dni. Jako nastavení počtu aktivních oddílů stačí dvě, protože data budou přicházet pouze v aktuálním dni. Výjimkou je stav, kdy se adaptér odpojí od sítě internet a data si ukládá do své paměti. Po odeslání opožděných dat nemusí být už odpovídající oddíl aktivní. Pro vkládání opožděných dat by vyhovoval pouze maximální počet oddílů, protože se nedá očekávat, jak stará data se budou v paměti adaptéru vyskytovat, a proto je vhodnější data starší půldne zahazovat, vkládání ošetřit aplikačně nebo data vkládat do základní tabulky, kde mohou být pomocí rutiny přesměrovávány do příslušných neaktivních oddílů. Protože není vhodné data uživatelů zahazovat

ani zvyšovat nároky na aplikační logiku je zvolen třetí způsob a to využívání rozřazovacích rutin.

Vliv partitioningu na časy odstraňování historických dat lze vidět v tabulkách 8.3 a 8.4. Odstranění jednoho dne v databázi, kde je partitioning použit vede ke smazání celé tabulky. Pokud se však odstraňují data z databáze bez partitioningu, musí se stará data dohledat a odpovídajícím způsobem změnit index,. Kromě vyššího času spotřebuje odstraňování dat z databáze bez partitioningu významně víc procesorového času a více zatíží disk. Dalším důvodem je rychlejší operace údržby a lepší škálovatelnost celého systému při použití partitioningu.

#pokročilých domácností	**čas s partitioningem**	**čas bez partitioningu**
1000	0,71 s	12,19 s
2000	5,32 s	204,82 s
3000	4,34 s	221,98 s
4000	6,9 s	618,15 s

Tabulka 8.3: Časy odstranění jednoho dne historických dat

#pokročilých domácností	**čas s partitioningem**	**čas bez partitioningu**
1000	5,51 s	130,45 s
2000	13,06 s	482,41 s
3000	15,28 s	540,51 s
4000	16,25 s	859,93 s

Tabulka 8.4: Časy odstranění jednoho týdne historických dat

8.5 Shrnutí a výsledky měření

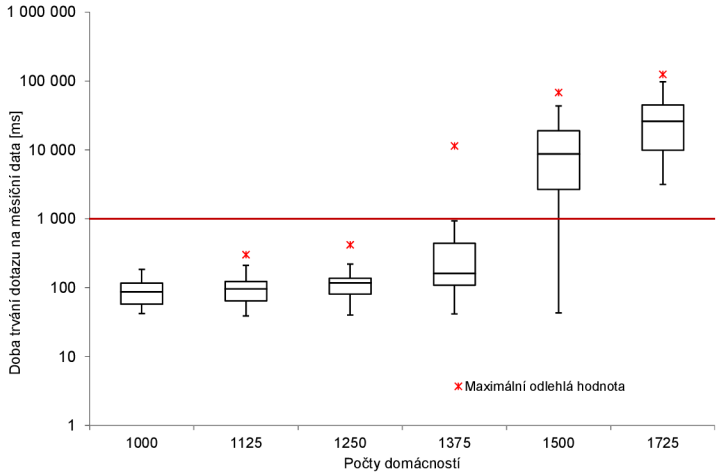
Na základě zjištěných dopadů optimalizací databáze byl pro databázi inteligentní domácnosti použit optimalizovaný konfigurační soubor, index využívající techniku index-only scan a horizontální dynamický partitioning.

Pro testování maximálního počtu připojených domácností se používaly databáze pokročilých domácností naplněné daty, které odpovídaly odhadu po měsíci používání. Při testu byla databáze vytěžována simulací provozu systému inteligentní domácnosti.

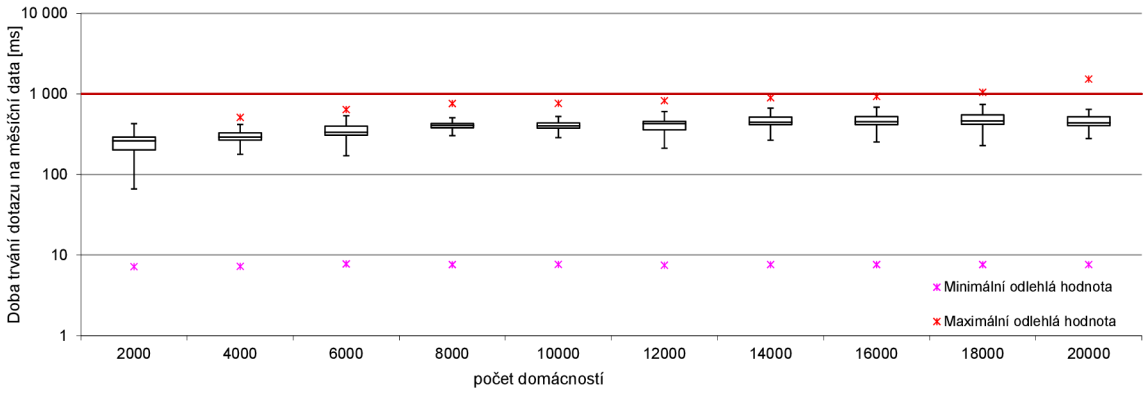
V testech se měřily časy dotazů a kontrolovalo se, zdali je databáze schopna vkládat příchozí data. Neuložení příchozích dat by znamenalo selhání databáze inteligentní domácnosti. Testy byly zaměřeny na nejsložitější dotazy, což byly dotazy na historická data senzorů v rozmezích jednoho měsíce, které musely být vykonány do jedné sekundy.

Na grafu 8.5 jsou zobrazeny časy trvání dotazů na historická data jednoho měsíce v závislosti na počtu pokročilých domácností v optimalizované databázi. Z naměřených hodnot vyplývá, že maximální počet pokročilých domácností na testovacím stroji ant-1 je přibližně 16000 (asi 200GB na disku). Což je výsledek asi 13krát lepší než při použití řešení, které využívá nastavený konfigurační soubor a index v tabulce historických hodnot používající k vyhledávání MAC adresu, typ a časové razítko. Časy dotazů na měsíční historická data v méně výkonném řešení lze vidět na grafu 8.4. Z těchto výsledků soudím, že v případě

databáze s 1250 domácností se skoro celá databáze vešla do operační paměti, a proto byly dotazy dostatečně rychlé. Pokud se významná část databáze do paměti nevešla, časy dotazů se výrazně prodloužily.



Obrázek 8.4: Graf trvání dotazů v závislosti na počtu domácností



Obrázek 8.5: Graf trvání dotazů v závislosti na počtu domácností při použití optimalizovaného řešení

Kapitola 9

Závěr

Cílem práce bylo seznámit se s řešením inteligentní domácnosti, které je vyvíjeno v rámci projektu IoT na Fakultě informačních technologií, definovat požadavky, navrhnout a implementovat databázi inteligentní domácnosti. Všechny cíle práce se mi povedlo splnit. Navíc byla v rámci práce implementována serverová aplikace, která obsluhuje uživatelská komunikační zařízení.

Řešení práce začalo studiem systému inteligentní domácnosti, které se vyvíjí na FIT a pokračovalo studiem již existujících systémů. Ze získaných znalostí byly definovány požadavky na databázi, které jsou shrnuty v kapitole 5.1.

Byly nastudovány současně používané typy databází a vytvořen přehled, který se nachází v kapitole 4. Na základě získaných informací, o současných databázových systémech a požadavků na databázi, byl použit systém PostgreSQL.

Dalším krokem byl návrh a implementace databáze. Byly identifikovány entity a vztahy v inteligentní domácnosti a na jejich základě byl vytvořen konceptuální model. Model je rozdělen do logických celků, podrobně popsán a následně vytvořen logický model. Na základě logického modelu byl vytvořen fyzický návrh databáze.

Navržená databáze byla testována na stroji s procesorem Intel Xeon E5410, 12GB RAM a jedním diskem. Výkonnostní testy ukázaly, že je databáze schopna obsloužit přibližně 16000 domácností, což je asi 13krát více než při použití neoptimalizovaného řešení. Optimalizované řešení navíc vykazuje výrazně lepší škálovatelnost.

V rámci této práce byla, kromě návrhu a implementace databáze, vytvořena serverová aplikace a softwarová vrstva, která slouží pro přístupu k datům v databázi. Aplikace vytváří šifrovanou komunikaci mezi ní a mobilními zařízeními a přes tento komunikační kanál zprostředkovává data inteligentní domácnosti uživatelům, a případně mění uživatelská nastavení, která se v databázi vyskytují.

Serverová aplikace a databáze byla průběžně testována pomocí alfa testování, které bylo dostupné pouze mezi vývojáři a pomocí beta testování, které ověřovalo funkčnost pravidelně vydávaných stabilních verzích systému. Serverová aplikace byla zároveň zasazena do kontinuální integrace pomocí softwaru Jenkins.

V pokračování této práce by mohly být další možnosti optimalizace jako například ukládání senzorických dat přímo do souborového systému bez použití databázového systému nebo použití specializovaných databází určených pro internet věcí.

Literatura

[1] Objektové databáze. http://www.fit.vutbr.cz/study/courses/VPD/public/0203VPD-Svec.pdf, 2003 [cit. 2015-1-4].

[2] Automatická správa partition v PostgreSQL. http://www.fuzzy.cz/cs/clanky/automaticka-sprava-partitions-v-postgresql/, 2009 [cit. 2015-5-1].

[3] Updated database landscape graphic. http://blogs.the451group.com/information_management/2012/11/02/updated-database-landscape-graphic/, 2011 [cit. 2015-15-5].

[4] Five steps for PostgreSQL performace. http://www.slideshare.net/PGExperts/five-steps-perform2013, 2013 [cit. 2015-5-1].

[5] Financing SmartThings' Future. http://blog.smartthings.com/news/, 2014 [cit. 2015-1-3].

[6] Cortana Lights Up Your Home. http://www.newzfeed.org/technology/22346-cortana-lights-home.html, 2014 [cit. 2015-1-3].

[7] Architektura Haidy home. http://www.haidyhome.cz/downloads/HH_product_sheet_1309_CZE.pdf, [cit. 2014-1-3].

[8] Datasheet mFi Machine-to-Machine Management System. http://dl.ubnt.com/datasheets/mfi/mFi_DS.pdf, [cit. 2015-1-3].

[9] Relační vs. objektově-relační vs. objektové databáze. http://www.fi.muni.cz/~xbatko/oracle/compare.html, [cit. 2015-1-4].

[10] Dokumentace knihovny pugixml. http://pugixml.org/docs/manual.html, [cit. 2015-4-11].

[11] Dokumentace knihovny Jansson. https://jansson.readthedocs.org/en/2.7/, [cit. 2015-4-2].

[12] Knihovna libcurl. http://curl.haxx.se/, [cit. 2015-4-2].

[13] Dokumentace PostgreSQL: pgbench. http://www.postgresql.org/docs/9.3/static/pgbench.html, [cit. 2015-5-1].

[14] Stránky projektu Jenkins. https://wiki.jenkins-ci.org/display/JENKINS/Home, [cit. 2015-5-1].

[15] Tuning Your PostgreSQL Server. https://wiki.postgresql.org/wiki/Tuning_Your_PostgreSQL_Server, [cit. 2015-5-1].

[16] Dokumentace knihovny OpenSSL. https://www.openssl.org/docs/, [cit. 2015-5-15].

[17] Stránky projektu NetBeans IDE. https://netbeans.org/, [cit. 2015-5-15].

[18] Dokumentace knihovny SOCI. http://soci.sourceforge.net/doc/3.2/, [cit. 2015-5-3].

[19] Atzori, L.; Iera, A.; Morabito, G.: The Internet of Things: A Survey. *Comput. Netw.*, Říjen 2010: s. 2787–2805, ISSN 1389-1286.
URL http://dx.doi.org/10.1016/j.comnet.2010.05.010

[20] Brychta, T.: *Bezdrátové senzory pro inteligentní domácnost*. bakalářská práce.

[21] Kennedy, J. B.: WHEN WOMAN IS BOSS: An interview with Nikola Tesla [online]. http://www.tfcbooks.com/tesla/1926-01-30.htm, 1926 [cit. 2014-12-30].

[22] Liddle, S.: *Database and expert systems applications 23rd International Conference, DEXA 2012, Vienna, Austria, September 3-6, 2012. Proceedings*. Berlin New York: Springer, 2012, ISBN 978-3-642-32596-0.

[23] Mareček, J.: *NoSQL database for the storage of OWL data*. Master's thesis.

[24] Turing, A. M.: Computing Machinery and Intelligence [online]. http://cogprints.org/499/1/turing.html, 1950 [cit. 2014-12-30].

Příloha A

Logický model databáze inteligentní domácnosti

A.1 Entity

Entity databáze inteligentní domácnosti jsou:

Uživatelé (users)
reprezentují osoby, kteří používají a ovládají prvky inteligentní domácnosti prostřednictvím adaptéru.

Adaptéry (adapters)
jsou vstupními branami inteligentních domácností. Reprezentují nejen fyzický prvek, který slouží ke komunikaci se zařízeními, ale i celou domácnost, protože v jedné domácnosti je jeden adaptér, který ji řídí a přes které proudí všechna data koncových zařízeních v dané domácnosti.

Mobilní zařízení (mobile_devices)
reprezentují zařízení prezentační vrstvy prostřednictvím kterých komunikují uživatelé s inteligentními domácnostmi, jedná se především o mobily, tablety nebo chytré televize.

Koncové zařízení (facilities)
reprezentují jednotlivá zařízení inteligentní domácnosti. Jedná se o reprezentaci fyzických prvků, které v sobě můžou agregovat více komponent jako jsou čidla nebo aktivní prvky. Koncová zařízení lze taky chápat jako řídicí a komunikační jednotka, která dokáže pracovat s čidly/aktivními prvky, zpracovávat jejich data a poslat na adaptér.

Čidla a aktivní prvky (devices)
reprezentují umístěna na koncových zařízeních, používají se pro monitorování svého okolí(čidla) nebo změnu svého okolí(aktivní prvky). Jedná se o jednoduché prvky, které většinou nemají schopnost komunikovat se svým okolím a jsou řízena řídící jednotkou, která je společná pro koncové zařízení.

Místnosti (rooms)
reprezentují rozdělení domácnosti na místnosti, případně jiné části jako garáž, zahrada

nebo jiné. Používají se pro rozřazení zařízení podle toho, kde jsou v domácnosti nainstalovány a tím zvyšují přehlednost grafického uživatelského rozhraní a aplikacích.

Historická data (logs)
reprezentují hodnoty naměřené v minulosti čidly nebo minulé stavy aktivních prvků. Slouží k vizualizaci stavu domácnosti prostřednictvím grafů, které si uživatel může prohlížet například v počítači. Druhé využití těchto dat je v algoritmech, které mohou data z minulosti používat pro svůj (přesnější) výpočet.

Pohledy (views)
uchovávají množinu čidel/aktivních prvků, které si uživatel zvolil. Slouží pro přehlednější zobrazení dat v aplikacích prezentační vrstvy a umožňují uživateli příjemněji kontrolovat stav své domácnosti.

Algoritmy (algorithms)
reprezentují typy algoritmů, které mohou uživatelé používat ve své domácnosti. Ty se rozdělují na algoritmy, které slouží pouze jednomu uživateli nebo na algoritmy, které jsou pro celou domácnost.

Úspěchy (achievements)
reprezentují úspěchy, které uživatelé dosáhli ve svém užívání chytré domácnosti.

Notifikace (notifications)
reprezentují notifikace, které se zasílají na mobilní zařízení.

A.2 Vztahy

Vztahy mezi entitami inteligentní domácnosti jsou:

Uživatel–Adaptér (N:M)
Každý uživatel může mít přístup k více adaptérům. Každý adaptér může být sdílen mezi více uživateli. Tento vztah obsahuje atribut role, která omezuje možnosti používání inteligentní domácnosti.

Uživatel – Mobilní zařízení (1:M)
Každý uživatel může ovládat inteligentní domácnost přes mobilní zařízení. Není očekáváno, že by byl mobil sdílen mezi více uživatelů.

Adaptér–Místnost (1:M)
Adaptér zastupuje inteligentní domácnost. Zde je proto přiřazován k více místnostem, které se v domácnosti nacházejí. Neočekává se, že v jedné domácnosti budou dva adaptéry, proto místnost nemůže mít vztah s více adaptéry.

Adaptér – Koncové zařízení (1:M)
Přes adaptér může komunikovat více koncových zařízeních(musí být spárována s adaptérem). Koncová zařízení nemohou být sdílená mezi více adaptéry a z bezpečnostních důvodu přes ně nesmí ani komunikovat.

Koncové zařízení – Místnosti (M:1)
Více zařízení se může nacházet ve stejné místnosti. Není možné umístit zařízení zároveň do více místnosti, na podobnou funkcionalitu se dají použít pohledy.

Koncové zařízení – Čidla a aktivní prvky (1:M)
Koncové zařízení může obsahovat více čidel nebo aktivních prvků. Čidla ani aktivní prvky nemohou být sdíleny mezi více koncovými zařízeními.

Čidla a aktivní prvky – Historická data (1:M)
Čidla a aktivní prvky mohou mít libovolný počet naměřených dat v minulosti. Jedna hodnota nemůže být spojena s více čidly/aktivními prvky.

Uživatel–Pohled (1:M)
Každý uživatel si může nastavit více pohledů. Pohledy není možno mezi uživateli sdílet, proto pohled nemůže mít více uživatelů.

Pohled – Čidla a aktivní prvky (M:N)
Každý pohled může obsahovat libovolný počet čidel nebo aktivních prvků. Čidla a aktivní prvky mohou být součástí více pohledů. Tento vztah vzniká a zaniká na uživatelův podnět, který si nastavuje obsah pohledů podle svého uvážení.

Algoritmus – Uživatel – Zařízení (M:N:O)
Uživatelé si mohou nastavovat různé algoritmy, které pak sledují hodnoty na vybraných zařízeních. Zařízení mohou být součástí více algoritmů a algoritmus může sledovat libovolný počet zařízení.

Algoritmus – Adaptér (M:N)
Některé algoritmy pracují v rámci jedné domácnosti a proto se vztahují na daný adaptér. Mají k nim přístup uživatelé dané domácnosti a algoritmus může sledovat zařízení v celé domácnosti

Úspěchy – Uživatel (M:N)
Každý uživatel má přiřazeno více úspěchů. Některé úspěchy se mohou mít delší průběh a proto se zaznamenává i jejich dílčí stav.

Úspěchy – Adaptér (M:N)
Úspěchy mohou být zaznamenávány i v rámci jedné domácnosti, kdy se na něm mohou podílet všichni uživatelé. V jedné domácnosti a tedy na adaptéru se může nacházet libovolné množství úspěchů.