



**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**

BRNO UNIVERSITY OF TECHNOLOGY

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**

FACULTY OF INFORMATION TECHNOLOGY

**ÚSTAV INFORMAČNÍCH SYSTÉMŮ**

DEPARTMENT OF INFORMATION SYSTEMS

**NÁSTROJ PRO SPRÁVU MODULŮ NA PLATFORMĚ  
SMART CITY**

TOOL FOR MANAGING MODULES ON SMART CITY PLATFORM

**BAKALÁŘSKÁ PRÁCE**

BACHELOR'S THESIS

**AUTOR PRÁCE**

AUTHOR

**TOMÁŠ SOUČEK**

**VEDOUcí PRÁCE**

SUPERVISOR

**Ing. JIŘÍ HYNEK, Ph.D.**

BRNO 2023

## Zadání bakalářské práce



146397

Ústav: Ústav informačních systémů (UIFS)  
Student: **Souček Tomáš**  
Program: Informační technologie  
Specializace: Informační technologie  
Název: **Nástroj pro správu modulů na platformě Smart City**  
Kategorie: Informační systémy  
Akademický rok: 2022/23

### Zadání:

1. Prostudujte oblast internetu věcí (*Internet of Things*) a koncept chytrých měst (*Smart City*).
2. Proveďte průzkum současných populárních redakčních systémů a zaměřte se na způsoby jejich modularizace a rozšiřitelnosti. Prozkoumejte řešení přístupů k modulům systému, k uživatelským akcím modulů a zpřístupnění obsahu jednotlivým uživatelům nebo skupinám uživatelů.
3. Analyzujte současný stav modularizace a správy přístupu na platformě Smart City firmy Logimic. Zhodnoťte nedostatky a definujte požadavky.
4. Navrhněte rozšíření platformy Smart City umožňující uživatelsky přívětivou správu modulů klientské aplikace platformy a následného přístupu skupin uživatelů k těmto modulům (včetně uživatelských akcí poskytovaných jednotlivými moduly).
5. Navržené rozšíření implementujte.
6. Proveďte testování funkčnosti a použitelnosti implementace na platformě Smart City firmy Logimic.

### Literatura:

- Greengard, S. (2015). *The Internet of Things*. MIT Press.
- Kiritat, A., Krejcar, O., Kertesz, A., & Tasgetiren, M. F. (2020). Future trends and current state of smart city concepts: A survey. *IEEE access*, 8.
- Interní dokumentace firmy Logimic.

Při obhajobě semestrální části projektu je požadováno:  
Body 1 až 4.

Podrobné závazné pokyny pro vypracování práce viz <https://www.fit.vut.cz/study/theses/>

Vedoucí práce: **Hynek Jiří, Ing., Ph.D.**  
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.  
Datum zadání: 1.11.2022  
Termín pro odevzdání: 10.5.2023  
Datum schválení: 25.10.2022

## Abstrakt

V posledních letech p íbývá razantním tempem po et digitálních za ízení, kterými je lo- v k, jak v domácnosti, tak v práci a pr myslu, dennodenn obklopen. Aby se dala data jednoduše vizualizovat a zpracovávat, vzniká množství platform, které to umož ůjí. Cílem této práce bylo navrhnout rozší ení platformy SmartCity, které by umož ůvalo uživatelsky p ív tivou správu modul ů a zároveň umož ůnilo zásahy do aplika ní konfigurace platformy za jejího b hu. Rozší ení platformy povede k lepším možnostem personalizace dashboard a pomocí grafického rozhraní umož ůní zm ny nastavení aplikace i uživatel m – a to bez zna- losti vnit ní programové logiky. P ínosem rozší ení bude také eliminace op tovné kompilace aplikace p í každé zm n nastavení, jak tomu je nyní. Výsledkem praktické ásti práce je nástroj implementovaný formou rozší ujícího modulu, postaveného na technologii Angular. Tento modul umož ůje zm nu aplika ní konfigurace za pomoci formulá ového a tabulko- vého zobrazení. Vytvo ený modul byl zakomponován do platformy SmartCity a na jeho základech bude probíhat následný interní vývoj

## Abstract

In recent years there has been a rapid increase in the number of digital devices by which we humans are surrounded. Whether it is at work, at home, or in an industry. In order to e ectively process and visualize the gathered data, there has been a number of platforms emerging to make this possible. The aim of this thesis was to prepare an extension tool for the SmartCity platform which would not only allow user-friendly module management but also provide the capability of application configuration changes on the fly at runtime. These changes could be made through a graphical user interface even without the knowledge of internal program logic. That way there rises the possibility of better dashboard customi- zation. Another benefit of the extension tool is the elimination of application re-compilation every time there is a change in the application setting. The result of the practical part of this thesis is a tool implemented as an extension module, built on Angular technology. This module allows changing the application configuration using form and table views. This cre- ated module has been deployed to the SmartCity platform and will be the basis for further subsequent internal development.

## Klí ová slova

IoT, internet v cí, chytrá m sta, CMS, metamodel, modularizace, konfigurace za b hu

## Keywords

IoT, Internet of Things, Smart Cities, CMS, metamodel, runtime configuration

## Citace

SOU EK, Tomáš. *Nástroj pro správu modul ů na platform Smart City*. Brno, 2023. Ba- kalá ská práce. Vysoké u ení technické v Brn , Fakulta informa ních technologií. Vedoucí práce Ing. Ji í Hynek, Ph.D.

# Nástroj pro správu modulů na platformě Smart City

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jiřího Hynka, Ph.D. Další informace mi poskytli vývojáři firmy Logimic, s.r.o. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....  
Tomáš Souček  
3. května 2023

## Poděkování

Chtěl bych poděkovat vedoucímu této práce Ing. Jiřímu Hynkovi, Ph.D. za velmi vstřícný a obětavý přístup, rychlé reakce na dotazy a pomoc při řešení jak technické záležitosti, tak implementace. Také bych rád poděkoval Ing. Michalovi Valnému, Ph.D. za možnost spolupráce a nahlédnutí do reálného procesu tvorby SW.

# Obsah

<b>1 Úvod</b>	<b>2</b>
<b>2 Internet v cí</b>	<b>3</b>
2.1 Architektura . . . . .	4
2.2 Komunikace . . . . .	5
2.3 Bezpečnost . . . . .	6
2.4 Chytrá města . . . . .	7
2.5 Existující IoT platformy . . . . .	9
<b>3 Systémy pro správu obsahu</b>	<b>11</b>
3.1 Nástroje pro správu obsahu . . . . .	11
3.2 Modularizace . . . . .	12
3.3 Role a pravomoce . . . . .	13
3.4 Bezpečnost . . . . .	15
3.5 Současné open-source systémy . . . . .	16
<b>4 Analýza současného stavu</b>	<b>19</b>
4.1 Logimic a SmartCity . . . . .	19
4.2 Analýza uživatelů a definice problém . . . . .	21
4.3 Požadavky na změnu . . . . .	22
<b>5 Návrh</b>	<b>24</b>
5.1 Architektura rozšíření . . . . .	24
5.2 Datový model . . . . .	25
5.3 Uživatelské rozhraní . . . . .	27
<b>6 Implementace</b>	<b>30</b>
6.1 Architektura rozšiřujícího modulu . . . . .	30
6.2 Funkcionalita komponent . . . . .	32
6.3 Napojení na existující systém . . . . .	38
<b>7 Testování</b>	<b>39</b>
7.1 Testovací prostředí . . . . .	39
7.2 Testovací scénáře . . . . .	39
7.3 Výsledky testování . . . . .	41
<b>8 Závěr</b>	<b>42</b>
<b>Literatura</b>	<b>43</b>
<b>A Snímky obrazovky</b>	<b>47</b>

# Kapitola 1

## Úvod

V posledních letech probíhá razantním tempem proces digitálních transformací, kterými je život dennodenně obklopen. Není výjimkou nosit u sebe chytrý telefon, nicméně probíhá i proces lidí s chytrými hodinkami, bezdrátovými sluchátky, tablety a do budoucna třeba i brýlemi s rozšířenou realitou. Stále více lidí si do svých domácností instaluje chytré spotřebiče, které jsou schopni ovládat na dálku (a už se jedná o termostat kotle, elektronické žaluzie či kávovar). Chytrá transformace se také čím dál tím více stává nedílnou součástí praxe. V rámci tzv. průmyslu 4.0 bude díky chytrým transformacím docházet k velké míře automatizace a robotizace. To bude, mimo jiné, umožněno díky větší rychlosti přenosu dat a 5G sítím. S tím se ruku v ruce dostává pojem Internet věcí a chytrých měst do podvědomí odbornější veřejnosti. Města budou moci využívat inteligentního řízení dopravy, šetřit náklady i prostředky pomocí regulace ve veřejném osvětlení či zapojit více občanů do rozhodování díky digitalizaci různých systémů. Nicméně, aby byla schopna tato chytrá transformace spolu komunikovat, vzniká potřeba existence platform toto umožňujících. Nejde jen o samotnou komunikaci, ale často je také potřeba data vhodně vizualizovat, aby byly monitoring a vzdálená správa transformací uživatelsky přístupivější.

Jednou z firem zabývajících se vývojem a zprostedkováním IoT řešení pro města a průmysl je firma Logimic a jejich produkt SmartCity. Platforma SmartCity je v současné době stále ve vývoji a agilně se mění podle momentálních požadavků zákazníků. Aby mohlo docházet k personalizaci řešení pro jednotlivé zákazníky, musí v současnosti programátoři měnit nastavení zásahem do kódu, na což musí dojít k následnému překladač. Víze firmy je možnost měnit nastavení aplikace dynamicky za běhu a delegovat část úprav na zákazníka. Proto se tato práce zabývá zlepšením modularizace a možnostmi nastavení uživatelských práv. Cílem je vytvořit rozšiřující modul klientské části SmartCity tak, aby mohla probíhat změna nastavení přímo z aplikace za jejího běhu a nemuselo kvůli každé změně docházet k úpravě kódu a opětovnému překladač. Změna musí probíhat uživatelsky přístupivým způsobem. Tato práce také slouží jako příprava pro možnost vytváření plně interoperabilních dashboardů, čímž dojde k lepší personalizaci zákaznických řešení.

První část práce se věnuje teorii, která je potřebná pro pochopení práce jako celku. Kapitola 2 se zabývá vysvětlením esenciálních pojmů ohledně internetu věcí a chytrých měst. Další nezbytnou částí jsou principy fungování CMS. Výsledky průzkumu lze najít v kapitole 3. Po získání všech potřebných informací došlo k analýze současného řešení firmy Logimic. Více informací o používaných technologiích, výhodách i nedostatcích implementace a ostatních detailech se nachází v kapitole 4. V praktické části jsou v kapitole 5 nastíněna možná řešení. Navazující kapitola 6 popisuje implementační detaily. V neposlední řadě došlo i na testování, jehož průběh a výsledky lze najít v kapitole 7.

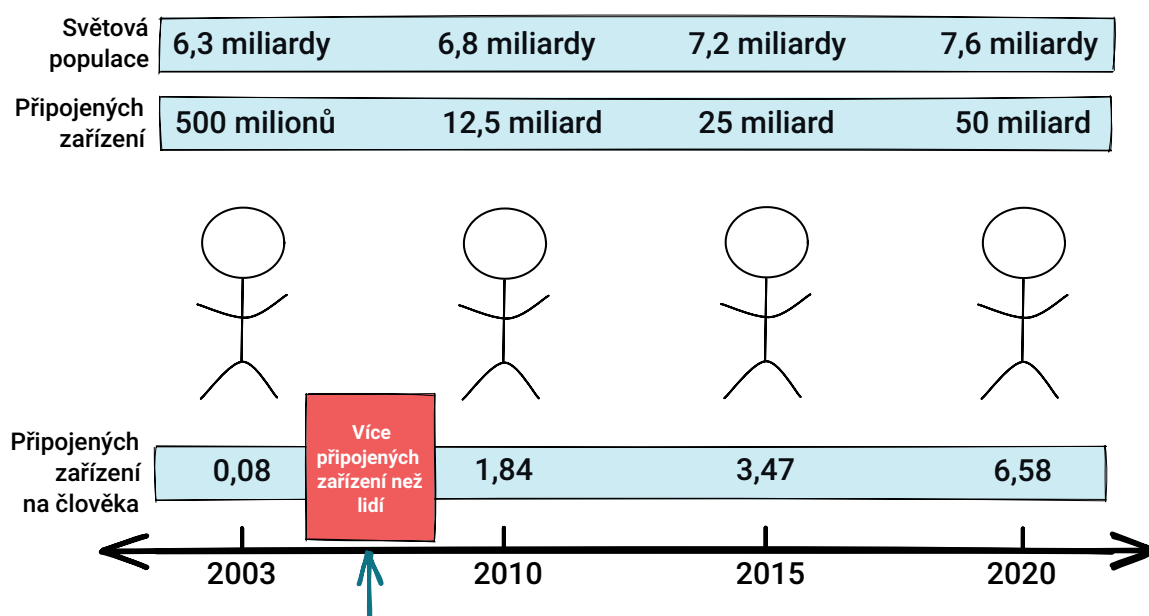
## Kapitola 2

# Internet v cí

A koliv se m že zdát, že pojem Internet v cí (anglicky *Internet of things* – zkrácen IoT) je pom rn nová v c, historie chytrých za ízení sahá již do za átku 80. let, kdy bylo v roce 1982 p ipojeno první „chytré“ za ízení k síti Arpanet (nyní známé spíše jako internet). Jednalo se o automat na nápoje Coca-cola, který um l na dálku sd lovat informace, zda je prázdný a jestli jsou nápoje po dopln ní již vychlazené [1].

Cisco Internet Business Solutions Group (IBSG) pojmenovala termín IoT jako „bod v áse, ve kterým je k internetu p ipojeno více za ízení než lidí“. Což Cisco odhaduje, že je fáze, do které jsme se dostali mezi lety 2008–2009 [18].

Obecn by se dalo íct, že internet v cí je sí mnoha propojených za ízení, které mezi sebou komunikují a sdílí data k dosáhnutí spole ného cíle [32]. V praxi se pak nap . m že jednat o teplotní senzory, které když zaznamenají propad teploty pod nastavenou hranici, dají signál kotli, aby spustil vytáp ní.



Obrázek 2.1: O ekávaný r st po tu IoT za ízení, inspirováno [18]

Pojem Internet v cí m že p sobit také lehce zavád jícím dojmem, že za ízení komunikují p es internet. Za ízení sice spolu musí komunikovat prost ednictvím propojené sít , nemusí se nicmén jednat p ímo o ten všeobecn známý internet [15].

Podle Samuela Greengarda firma IoT Analytics p edpokládá, že v roce 2025 bude existovat 21,5 miliardy IoT za ízení [22]. Dle výše zmín ného obrázku 2.1 lze íct, že Cisco bylo se svým odhadem z roku 2011 ještě optimisti t jší. A již predikce vyjde, í nikoliv, z ísel plyne, že se jedná o velký pojem, který má potenciál zm nit lidské životy.

## 2.1 Architektura

Architektura IoT je dle [16] tvo ena množinou komponent, jako jsou senzory, protokoly, cloud a vrstvy tvo ící sí ové systémy. A koliv se nedá p ímo ur it počet vrstev a mnohé lánky uvádí r zné architektonické vzory, jsou v této práci dle [3] uvažovány 2 hlavní základní typy: t ívrstvá a p tivrstvá architektura. Jednotlivé vrstvy mají za cíl umož ovat správce m monitorovat a udržovat integritu systému a klient m pomáhat s požadovanými úkoly.

Nejnižší úrove se skládá z množství senzor . Dané senzory m žou pomocí komunika - ních protokol (nejznám jší zmín né v kapitole 2.2) komunikovat 3 zp soby: mezi sebou, na p ímo s cloudem nebo s hubem (gateway). Hub p íjímá a agreguje data od množství senzor a následn je zasílá do cloudu ke zpracování. N které výkonn jší huby jsou schopny data p ed odesláním p edzpracovat a zmenšit tím režíe na cloudu. Cloud b žící na pozadí následn využívá databázi pro práci s perzistentními daty a pro komunikaci s klientskou ástí aplikace používá nej ást ji API.

### 2.1.1 T ívrstvá

Díky svojí obecnosti je považována za základní architekturu, ze které vychází další specifické odnože. Jak již název napovídá, skládá se ze t í vrstev a to: fyzické, sí ové a aplika ní.

Fyzická vrstva, anglicky také nazývaná jako *perception layer*, obsahuje konkrétní za ízení a senzory. Jejím ú elem je detekovat a sbírat informace ze svého okolí a následn je posílat sí ové vrstv . Tato vrstva zároveň také zajiš uje spolupráci v malých a lokálních sítích a p ípojení k lokálním hub m [32].

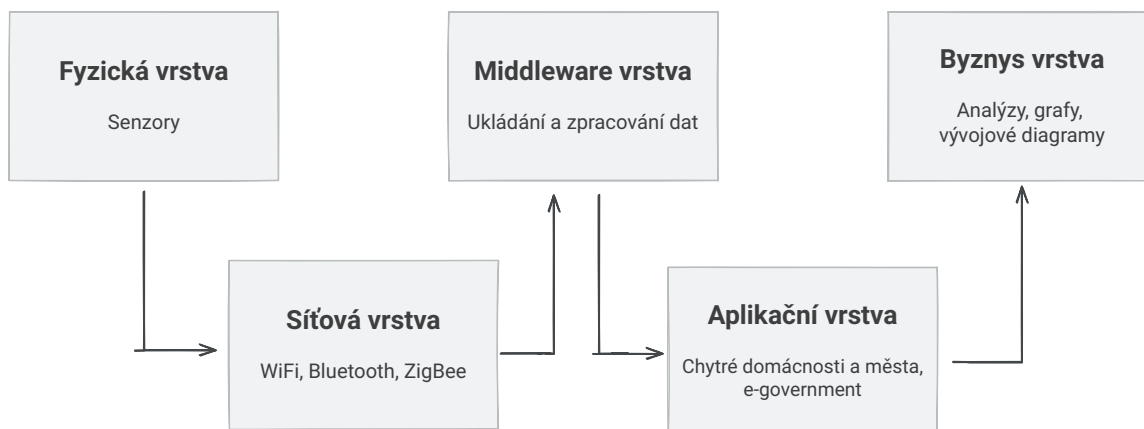
Sí ová vrstva je spojovníkem mezi datovou a aplika ní vrstvou. Pomáhá datové vrstvě p enášet data skrze brány a huby nap í v tšími sít ími, jako je nap . internet. Na této vrstvě se komunikuje pomocí technologií jako jsou Bluetooth, Zigbee a další [32] o kterých se práce detailn ji zmi uje v následující kapitole.

Nejvyšší, aplika ní, vrstva pak zajiš uje autenticitu, integritu a d v ryhodnost dat. Jedná se o vrstvu, ve které jsou p íjatá data analyzována a zpracována, aby mohla poskytnout požadovanou službu [32, 3]. Na této vrstvě dochází ásto ke cloudovému zpracování dat a samotné interakci uživatele se za ízeními.

### 2.1.2 P tivrstvá

P tivrstvá architektura vychází z obecn jší t ívrstvé a p ídává k ní další 2 mezivrstvy: *middleware*, *byznys*.

*Middleware*, esky by se dalo p eložit jako prost edník, zajiš uje flexibiln jší propojení mezi hardwarem a aplikací. Stará se o zp sob ukládání a p id lování dat a podniká akce, kterými se snaží dosáhnout optimalizace proces . Byznys vrstva je nejvyšší vrstvou a zapouzd uje tak celou aplikaci a správu služeb. Obsahuje grafy, vývojové diagramy a analýzy [13]. Na obrázku 2.2 lze vid t její schéma.



Obrázek 2.2: P tivrstvá architektura IoT za ízení. Inspirováno z [40]

## 2.2 Komunikace

Aby spolu mohla za ízení efektivn komunikovat, je t eba dodržovat ur itá pravidla a využívat jednotné standardy na zpracování dat. K tomu slouží tzv. protokoly. Komunika ní protokoly, používané pro IoT za ízení, se dají rozd lit podle délky dosahu signálu na 2 kategorie: LPWAN a Short range [4].

### 2.2.1 LPWAN

Zkratka LPWAN stojí za anglickým výrazem *Low Power Wide Area Network*, což by se dalo do eštiny p eložit jako nízkoenergetické rozlehlé síť . Tedy kategorie protokol sloužících pro komunikaci na v tší vzdálenosti za používání malého množství energie.

Mezi nejznám jší protokoly se zde adí:

- SigFox – vyvinut pro posílání velmi malého množství dat na nízkých frekvencích s nízkou energetickou náročností. Hlavní výhodou je spolehlivost doručení dat a odolnost vůči interferencím a kolizím, jelikož se každý paket posílá zároveň na 3 komunikačních kanálech [30].
- Celulární rádiové síť – vhodná technologie pro aplikace vyžadující velkou propustnost dat na velké vzdálenosti. Kvůli vysoké rychlosti přenosu dat se však spíše hodí pro za ízení, která jsou připojena k elektrické síti. Primárně se síť používají pro mobily, takže výhodou tohoto řešení je velká míra pokrytí signálem [4].
- LoraWan – další rádiová technologie sloužící pro komunikaci na velmi dlouhé vzdálenosti. Koncová za ízení komunikují skrze bránu, která má výkonnou anténu schopnou pokrýt až 100 km<sup>2</sup> [14].

## 2.2.2 Short range

Ne vždy je žádoucí přenášet signál na dlouhé vzdálenosti, typickým příkladem může být například placení platební kartou, nebo odemknutí auta v blízkosti klíče. Dalším příkladem využití může být poslouchání hudby prostřednictvím bezdrátových sluchátek. Jedná se o situace, kdy vyžadujeme velkou datovou propustnost a nevadí nám tolik energetická náročnost. I z těchto důvodů vznikly také protokoly umožňující komunikaci na krátkou vzdálenost.

Mezi nejpoužívanější patří následující:

- RFID – *Radio Frequency Identification* systémy se skládají z čtečky a tagu. Tag je pasivní část, která nemá své vlastní napájení. Čtečka zastupuje v tomto případě aktivní roli, vysílá rádiové vlny, které napájí tagy a díky tomu zajišťuje komunikaci. Tag může spolu se svým identifikačním číslem obsahovat také malé množství informací [4]. Využití si technologie najde například ve skladech, zabezpečení zboží proti krádeži nebo například u sportovních akcí pro měření času.
- NFC – neboli *Near Field Communication* je technologie operující na nízké frekvenci (13,56 MHz) využívaná pro komunikaci na velmi krátkou vzdálenost (4–10 cm) a přenos malého množství dat. Topologie sítě umožňuje pouze takzvanou P2P (*Peer to Peer*) komunikaci, tedy komunikace pouze mezi dvěma uzly. NFC může být buď v aktivním nebo v pasivním módu [11]. V současné době se využívá pro platby chytrými telefony, rychlé párování zařízení a odemknutí dveří.
- Bluetooth – na rozdíl od předchozích komunikuje na rádiově vyšší frekvenci (2,4 GHz). Jeho nevýhodou je ovšem velká spotřeba, pomalé párování a limitované množství komunikujících zařízení. Čím dál tím více se přechází na BLE (*Bluetooth Low Energy*), který byl vytvořen jako doplňková technologie pro klasický Bluetooth snažící se odstranit jeho nedostatky. Převážně se pak jedná o snížení spotřeby a zvýšení počtu připojených uzlů [24].
- ZigBee – vytvořeno jako standard pro vysokoúrovňové protokoly a jeho účelem bylo sloužit k vytváření domácích sítí. Používá se pro zařízení vyžadující menší přenosy dat, delší výdrž baterie a bezpečnost. Další výhodou ZigBee je topologie podporující zapojení v takovém množství zařízení [4, 11].

## 2.3 Bezpečnost

Abychom mohli v informatice tvrdit o systému, že je bezpečný, musí splňovat kritéria, jako jsou důvěryhodnost, integrita, dostupnost a autentizace. Dále bychom pro IoT mohli zařadit také vytvoření výpočetní bezpečnosti, možnost heterogenní komunikace a systém pro správu klíče [32]. S rostoucím počtem chytrých zařízení je potřeba zvyšovat i náklady na rozvoj bezpečnosti, jelikož se internet v cíl stává čím dál tím větší cílem útoků. Obzvláště pak v chytrých domácnostech, kde mohou zařízení zaznamenávat velmi citlivé informace, je potřeba na bezpečnost dbát. Důležité je také mít na paměti, že útoky nemusí být jen softwarového, ale také i fyzického charakteru.

Mezi časté útoky na IoT zařízení můžeme dle [32, 2] zařadit:

- fyzická manipulace (*physical tampering*),
- Man-in-the-Middle (MITM),
- odposlech (*eavesdropping*),
- brute-force útoky na hesla,
- vložení škodlivého uzlu (*malicious node injection*),
- sociální inženýrství (*social engineering*),
- DDoS (distribuované přetěžování serverů).

Etický hacker Ken Munro ve svém *TEDx talk* zmínil několik nedostatků zabezpečených chytrých zařízení a jak se jeho tým byl schopný dostat skrze jejich bezpečnostní trhliny k datům lépe zabezpečených systémů – například získání přístupu k wifi přes chytrou varnou konvici, přístup do domu skrze chytrý zámek, jehož heslo byla MAC adresa Bluetooth rozhraní daného zámku a dalších popsanych v [37].

Jako základní ochranu před útoky uvádí článek [12] následování doporučených bezpečnostních postupů (změna výchozího hesla, zálohování obsahu nebo použití šifrování), provedení inventury vlastních zařízení a nastavení systému pro lepší dohled, omezení přístupu jen pro autorizované správce, provádění pravidelných penetračních testů a kontrol zranitelnosti. V neposlední řadě se pak doporučuje kontrolovat zařízení fyzicky, zda nebyla modifikována nebo poškozena.

## 2.4 Chytrá města

Internet v ní hraje důležitou roli v rozvoji chytrých měst. Zařízení, jako jsou senzory a kamery, mohou být rozmístěny v rámci města pro sběr dat. Nasbíraná data lze pak následně použít pro efektivnější řízení města a zlepšování kvality života občanů.

Autoři [38] považují za chytré město takové, které prosazuje automatickou a efektivní správu městských infrastruktur a služeb. Dále by podle nich mělo usilovat o snížení veřejných útrat, zlepšení kvality služeb a zaměřovat se na klíčové aspekty jako je šetření energií, udržitelná přeprava, e-government a v neposlední řadě sociální a zdravotní zabezpečení. V ideálním případě by také mělo podporovat kreativitu a inovaci, aby bylo atraktivní pro vzdělané pracovníky, kteří budou schopni řešit nadcházející výzvy.

V článku [29] byl koncept chytrých měst rozebrán na následujících 6 sekcích:

- Chytří obyvatelé – pojem chytří obyvatelé může být na první pohled lehce zavádějící. Autoři článku [29] se nesnaží vykládat, že v moderních chytrých městech mají žít pouze inteligentní lidé, ale jde především o snahu popisu jejich propojení. Vychází z logiky, že města jsou stavěna primárně pro lidi, tudíž by měly systémy být tzv. *people-centric*, neboli primárně zaměřené na lidi. Nemělo by být opomíjeno, že obyvatelé u sebe nosí již v současné chvíli spousty senzorů ve formě chytrých telefonů. Ty mohou sloužit k lepšímu propojení obyvatel a městských systémů a také pomáhat sbírat cenná data, která se dají využít pro lepší plánování rozvoje. O jednom z případů se zmíňuje Gregory Mone ve svém článku [36], kde popisuje davovou studii, která měla za úkol

nastítnit možné dlouhodobé výhody pro cyklisty. Na jejím základě došlo k rozšíření jízdních pruhů a přidání stojanů, aby se optimalizovala dopravní infrastruktura.

- Chytrá ekonomika – chytrá ekonomika, neboli přesněji chytré podnikání a *e-commerce* by mohly zažít velkou expanzi v budoucnosti. Kromě optimalizace logistiky přibude také množství cest, jak se dostat k zákazníkovi. Studie [28] nastiňuje možnost nákupního systému, který by potenciálním zákazníkům poskytoval informace o produktech, o které by mohl mít zájem. Takové nabídky už máme nyní v současnosti fungují například v newsletterech. Předpokladem budoucího trendu by mohlo být zobrazení upozornění, když se zákazník přiblíží k danému obchodu, který má momentálně v nabídce zákazníkům šáňný produkt.
- Chytré řízení – díky chytrému řízení, anglicky *Smart governance*, by mohli obyvatelé lépe zasahovat do chodu obce. A se již jedná o návrh, implementaci a evaluaci procesu projektu, jako se tomu nyní děje například ve Španělské Barceloně [20]. Díky elektronickému přístupu k potřebným dokumentům, ukládání a sdílení opakujících se dat a vyřizování náležitostí na dálku by došlo k snížení byrokratické zátěže a úspore lidských i finančních zdrojů. Ovšem, aby takováto funkce fungovala, musela by být implementace podpořena vhodnou legislativou, silné zabezpečení a v ideálním případě také transparentnost.
- Chytrá mobilita – již v současné době se můžeme setkat s chytrými semafovy, které řídí křižovatku podle aktuálního vytížení, nebo například zastaví idíe, pokud překročí rychlostní limit. To je výzva, kterou má na starost chytrá mobilita. Jejím cílem je optimalizace dopravy tak, aby byly cesty využívány co možná nejefektivněji a nedocházelo k dopravním zácpám. Doktor Xiao-Feng Xie uvádí, že při pilotním průzkumu použití chytrých semaforů v americkém Pittsburghu došlo ke zkrácení doby nezbytné pro přejetí skrzem sto o 25 % a ke zkrácení doby čekání na semaforu dokonce o 40 % [46]. Nejen, že by takovouto optimalizací došlo k finanční úspoře obyvatel za palivo, ale také by došlo ke snížení emisí ze spalovacích motorů.
- Udržitelné životní prostředí – chytré město by nemělo opomíjet ani udržitelnost, která se stala v poslední době velmi diskutovaným tématem. Jedním ze závažných problémů novodobých měst je znečištění ovzduší, na které podle WHO umírájí ročně miliony lidí [45]. Díky IoT zařízení budou systémy schopny monitorovat kvalitu ovzduší a vody, měřit teplotu a zaznamenávat zdraví škodlivé látky a sledovat stav odpadu. Chytré město by také mělo být schopno řídit noční osvětlení tak, aby nedocházelo k narušení rovnováhy přírody.
- Chytré budovy – jako poslední ze zmíněných bodů je na místě zmínit skupinu chytrých budov. Fenomén, který se dostává čím dál tím více do běžných životů množství lidí, jsou chytré domácnosti. A již se jedná o přístroje zlehčující každodenní úkoly (automatické otevírání brány například při přiblížení, nastavení vytápění na základě odpovědí po aší a užívání domácích hlasových asistentů) nebo využití senzorů na ochranu majetku (například detekce pohybu a rozpoznání neznámého obličeje). Budov plných chytrých zařízení se dá využít samozřejmě i v průmyslu. V současné době přibývá počet skladů implementujících různé IoT technologie, aby docházelo k zvýšení efektivity i bezpečnosti. V Amazonu to dotáhli na úroveň, kdy není třeba robotů vyčlenit zvláštní prostor, ale jsou schopni sdílet pracoviště s lidmi [6]. Významnými jsou také chytrá zařízení

pomáhat ve zdravotnictví, kde je potřeba nepřetržitý monitoring životně důležitých funkcí.

## 2.5 Existující IoT platformy

Podle [23] jsou IoT platformy důležitě v mnoha aspektech našeho života. Považují se za páteř chytrých měst. Jsou vnímány jako prostředek monitorování okolního prostředí (například teplota, vlhkost), jako inteligence dopravních systémů včetně chytrých parkovišť. Cloudové servery sbírají data a ukládají je do distribuované databáze, aby mohly provádět filtrování, analýzu, výpočty, rozhodování, správu, ukládání a vizualizaci dat v koncových aplikacích službách. Článek [8] z roku 2021 řadí mezi nejpoužívanější platformy následující: AWS IoT, Apple HomeKit, Google Cloud IoT Core, ThingWorx a OpenHab.

Řešení od Amazon Web Services, **AWS IoT**, je platforma pro všeobecnou práci s IoT zařízením. Připojuje zařízení k serverům AWS, aby k nim měl následně uživatel vzdálený přístup. Programování zařízení probíhá pomocí API a vývojových kitů (SDK). Řešení obsahuje i webový portál pro základní správu zařízení. Díky propojení zařízení s hubem Alexa je uživatel schopen benefitovat z funkcionalit hlasového ovládání a dalších. AWS IoT je open-source řešení cloudového stylu a topologie, podporující škálu programovacích jazyků. Pro zabezpečení používá certifikáty a přístupové tokeny [8].

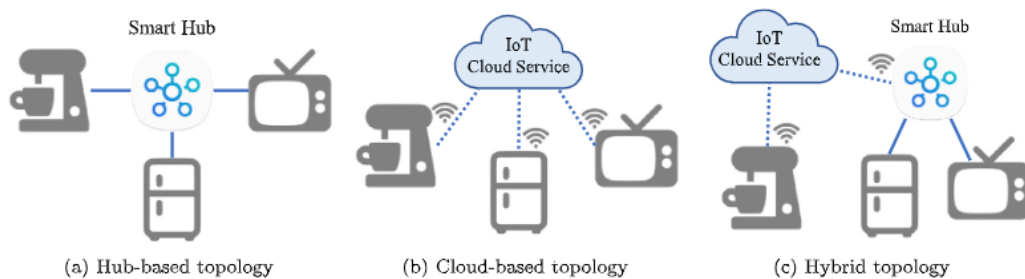
**Apple HomeKit** je proprietární platforma od společnosti Apple, jejímž cílem je propojení chytrých zařízení. Jedná se spíše o řešení pro chytré domácnosti než pro průmysl. Na rozdíl od ostatních zmíněných nevyžaduje centrální bránu ani hub pro zaručení komunikace, nýbrž jako centrální prvek využívá zařízení typu iPhone, iPad, MacBook a jiné zařízení s běžným operačním systémem od společnosti Apple. Jedná se o platformu založenou na cloudové topologii, fungující na principu *publisher/subscriber*, jež používá koncové šifrování (anglicky *end-to-end encryption*) pro zabezpečení přenášených dat. Apple k platformě dodává i aplikaci pro monitorung a správu připojených zařízení [8].

Řešení od technologické firmy Google se nazývá **Google Cloud IoT Core**. Hlavními komponentami jsou správce zařízení pro registraci a bridge pro připojení ke cloudu. Využívá se zde princip *publisher/subscriber*, tudíž server je v roli odběratele a zařízení v roli vydavatele. Platforma podporuje množství skriptovacích jazyků a pro zabezpečení se využívá princip sdílení tokenů a TLS. Pro integraci řešení existují API a SDK [39].

Platforma **ThingWorx** je zaměřená na průmyslové IoT řešení, snaží se pomocí automatizovat a vzdáleně sledovat procesy. Platforma je postavena na centrální cloudové aplikaci, ke které se dají skrze API a SDK připojit zařízení a následně s nimi pracovat. ThingWorx je open-source řešení, kde se platí za přenesená data [8] a využívá pro zabezpečení adresářovou službu LDAP [23].

Dalším open-source řešením spoléhajícím především na integraci hubu je **OpenHab**. Platforma pro chytré domácnosti založena na Java a Eclipse SmartHome frameworku. Díky své obecné architektuře jde o, z výše uvedených, nejvíce univerzální řešení, což se potvrdilo podporovanými zařízeními. Platforma využívá 3 typy spouštěčů pro zachycení změn v prostředí: událostní, časové a systémové. Zabezpečení řeší SSL certifikacími autority [8]. Pro platformu existují nativní aplikace pro Android, iOS a Windows.

Všechny platformy dle [8] podporují kromě výše zmíněných preferovaných topologií (hub a cloud) i topologii hybridní. Ta se vyznačuje kombinací hubu a připojení ke cloudu. Část zařízení může být například propojena do hubu typu Alexa a část může komunikovat přímo s cloudem viz obrázek 2.3.



Obrázek 2.3: 3 základní typy topologií IoT platform, převzato z [39]

## Kapitola 3

# Systemy pro správu obsahu

V roce 1992 existovalo podle Susan McKeever z Dublinské univerzity pouze 1 000 webových stránek na Internetu. O 8 let později uvádí záznamy informaci o více než 2 miliardách webů [33]. Takový prudký růst, rapidní změny obsahu stránek a nedostatečná automatizace vyvinuly velký tlak na vytvoření nástrojů pro správu obsahu.

System pro správu obsahu (anglicky *Content management system*, dále referovaný zkratkou CMS) je software poskytující určitou úroveň automatizace úkolů potřebných k efektivní správě obsahu. Vytváří obsah na serveru a interaguje s obsahem uloženým na úložišti nebo v databázi [9]. CMS dovoluje uživateli vytvářet, upravovat, publikovat a spolupracovat na digitálním obsahu. Typicky se CMS používají pro správu podniku nebo správu webového obsahu [7].

Podstatnou funkcionalitou a také velkou výhodou, kterou CMS často implementují, je možnost rozšíření funkcionality o tzv. moduly, což vede k ještě větší míře automatizace. Dále se v systémech dají nadefinovat uživatelské skupiny a přidat jim určitá práva pro interakci s obsahem. To vede k větší míře často vyhledávané datové abstrakce a k lepší formální definici celého systému.

Mezi hlavní klíčové prvky CMS se dají dle [19, 41] zařadit následující:

- nástroje pro správu obsahu,
- nastavení rolí a jejich pravomocí,
- modularizace,
- analytické a SEO nástroje,
- škálovatelnost,
- zabezpečení.

### 3.1 Nástroje pro správu obsahu

Vytváření a správa obsahu jsou základními prvky a důvodem vzniku CMS. Z toho důvodu je při výběru CMS nutné vybrat takový, který tvorbu umožní co možná nejpočetněji. V současné době už často nestačí mít pouze nástroj na psaní textu, nýbrž je žádoucí, aby umožňoval vkládat obrázky, videa, formuláře a další interaktivní grafické prvky,

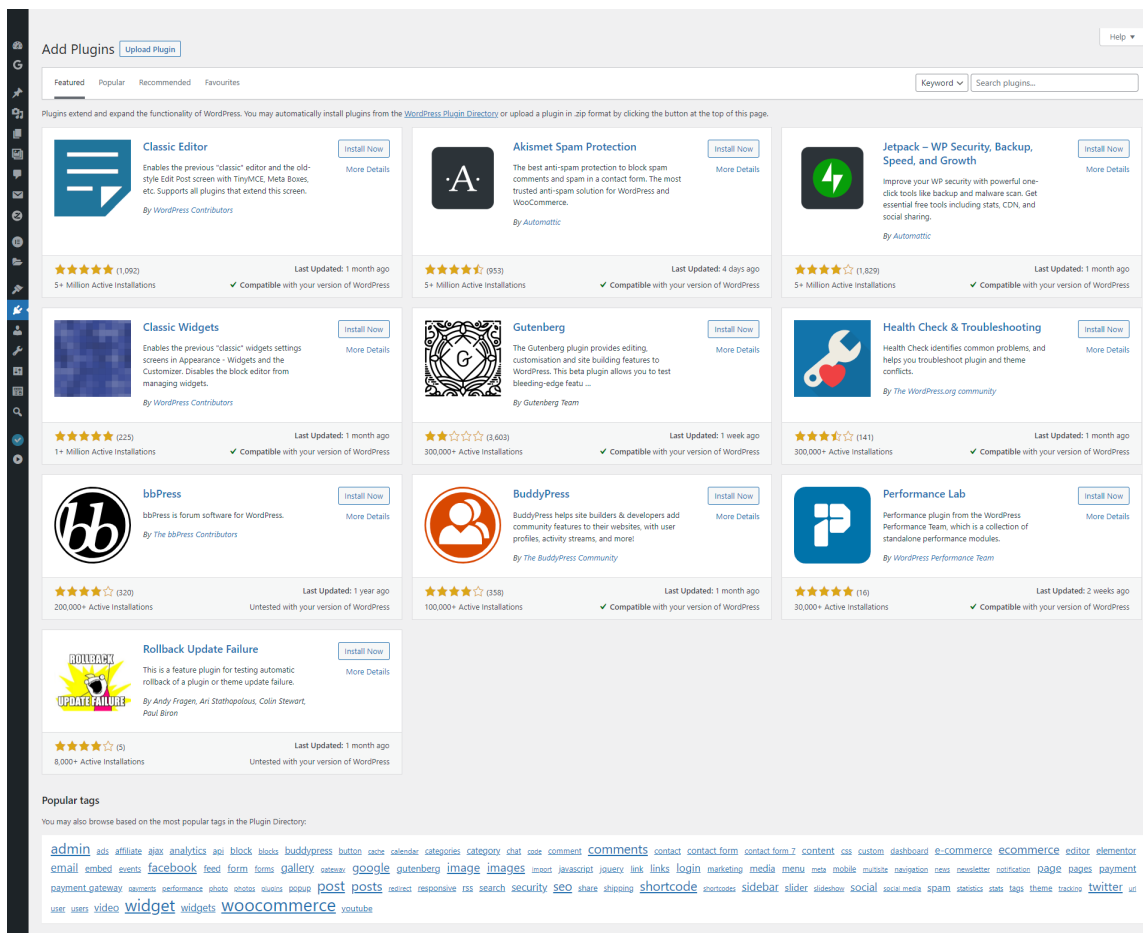
případně komponenty těchto stran (náhled Youtube videa a odkaz na příspěvek na Twitteru). Z toho důvodu za jiná hodnota CMS implementovat tzv. „WYSIWYG“ funkcionalitu. Anglická zkratka pro „*what you see is what you get*“, neboli *drag&drop* vizuální editory, které při psaní článku rovnou zobrazují náhled výsledku. Krom samotného vytváření obsahují moderní CMS často i škálu možností na publikaci článku – publikace jen pro určitou skupinu, přístup pouze přes odkaz a přístup po omezenou dobu [19].

Jelikož jsou CMS primárně vytvořeny pro snadné sdílení obsahu, existují kromě modulů na rozšíření funkcionality také šablony, které se zabývají úpravou vizuální stránky a interakcí s uživatelem. Taková šablona dokáže změnit vzhled i rozložení prvků na stránce a tím, po nastavení, pomáhá držet vizuální identitu daného webu. Šablona slouží k definici struktury, rozmístění prvků a nastavení vzhledu. Například máme-li blog, tak po nastavení šablony se dá daná šablona aplikovat na všechny články v dané kategorii. Tím autorovi odpadá nutnost starat se o vzhled a místo se plně soustředí na vytváření obsahu. Díky tomu dochází k zefektivnění práce. Jednotlivé šablony se dají nastavit různě pro jednotlivé stránky, tudíž díky nim dochází k lepší automatizaci různých typů obsahu. Samotná šablona může ve většině CMS upravovat i funkcionalitu, nicméně není k tomu primárně určena. Šablony se na své základní úrovni skládají z jednoho souboru implementujících kaskádové styly a hlavního souboru obsahujícího potřebná nastavení. Dále může obsahovat také soubory se skripty, i obrázky.

## 3.2 Modularizace

Modul, občas nazývaný také jako plugin, je typ počítačového software, který přidává již hotovému programu novou, nebo rozšiřuje jeho stávající funkcionalitu [42]. Díky rozšířením modulů se pak stávají již existující systémy robustnější a zároveň usnadňují práci vývojářům, kteří mohou již hotový generický modul použít ve svém řešení. A kolik není modularizace standardní vlastností každého CMS, jsou nyní implementovány téměř v každém hojně používaném systému. Typickými příklady rozšiřujících modulů pro CMS mohou být například moduly pro zabezpečení, SEO, automatické zálohy nebo optimalizace výkonu. Časté využití rozšíření může být také pro proměnlivého CMS třeba na e-shop. Na obrázku 3.1 lze vidět obchod s moduly, díky kterému lze procházet a instalovat rozšíření do systému Wordpress.

Základním souborem a funkcionalitami, které CMS obsahuje po instalaci a které tvoří nerozšířenou, ale plně funkční, verzi systému, se říká jádro. Jádro má, mimo nezbytné soubory, v sobě také zabudované množství základních funkcí pro správu obsahu, často také pro správu uživatelů a jejich rolí, přístupových práv nebo kaskádové styly pro vizuální reprezentaci dat. Je-li žádoucí rozšířit funkčnost systému, dochází k instalaci modulů. Jelikož jsou rozšíření často vytvářeny těmito stranami, definuje architektura CMS umístění, kam se mají zdrojové soubory rozšiřujícího modulu ukládat. Soubory rozšíření bývají obsaženy uvnitř jedné složky s názvem pluginu, která se musí uložit na konkrétní místo v systému (často podsložka *plugins*). Následně je nezbytné daný modul v nastavení CMS aktivovat. Aktivní stav se podle typu implementace konkrétního systému nejčastěji ukládá do databáze formou serializovaného textového řetězce (název, cesta k hlavnímu souboru a další metadata), nebo do konfiguračního souboru (například při azení hodnoty 1 do asociativního pole na index se jménem daného pluginu). Aby mohl modul se systémem správně komunikovat a vyměňovat si data, používají se předdefinované funkce i aplikační rozhraní (API). Aby si jádro s externím modulem správně poradilo, je třeba, aby dodržoval předem definovanou architekturu a syntaxi. Mnohé systémy také implementují funkci tzv. *hooks*. Jádro



Obrázek 3.1: Obchod s moduly v systému WordPress

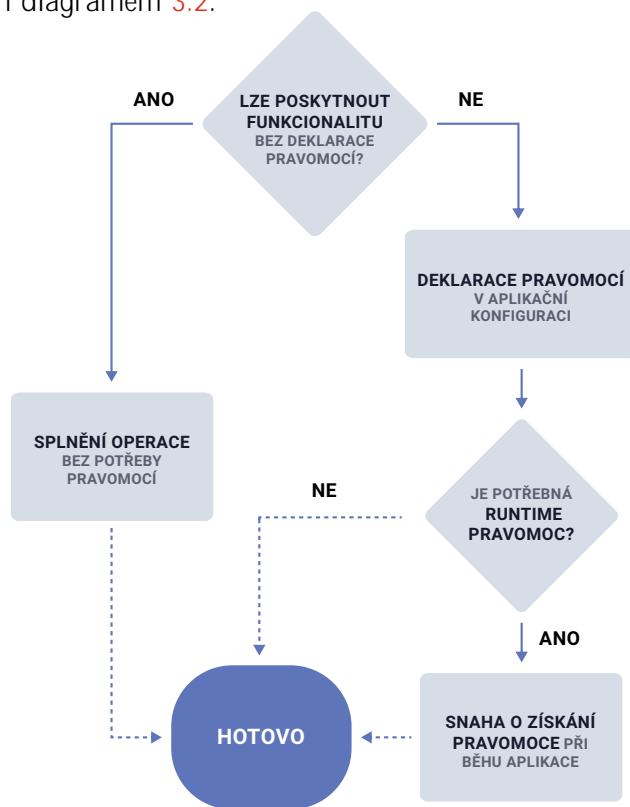
přítomnosti určitých specifických vlastností (např. vytvoření odkazu, změna jazyka nebo odstranění uživatele) vyvolá událost, která je díky *hooku* modulem zachycena a následně je vykonána požadovaná akce. Jedná se tedy o techniku komunikace mezi externím kódem a jádrem.

Nadužívání velkého množství modulů může bohužel systému i uškodit. S každým přidáním modulu roste výpočetní náročnost, a tak se může stát, že začne docházet k prodávám v načítání nebo zpracování obsahu. Přeci jen nejsou rozšíření stavěná na míru, ale genericky, takže se nedá mluvit o 100% optimalizaci pro daný systém. Druhým problémem pak mohou být bezpečnostní rizika, která mohou nedostatečně zabezpečená rozšíření přinést a mohou tak sloužit útočným hledajícím slabá místa pro proniknutí do celého systému.

### 3.3 Role a pravomoce

Kontrola, kdo má jaký typ přístupu k jednotlivým stránkám, je jednou ze silných výhod CMS a nezbytnou součástí drtivě většiny informačních systémů. Není například žádoucí, aby běžný uživatel mohl upravovat obsah stránek nebo měnit vnitřní data systému. Z toho důvodu jsou v systémech implementovány různé úrovně autorizací a kontrolu přístupu k částem systému a akcím. Mezi klasické základní oprávnění může žadatel definovat zobrazení, editaci a mazání obsahu i dat. Nicméně existuje množství detailnějších práv (např. rozlišení, kdo

m že editovat obsah a kdo uživatele). Ob akce sice spadají, z výše zmíněné trojice základních, do editace, ale jejich rozlišením že být pro určité systémy klíové. Práva se nenastavují pouze pro přístup, ale také k množinám prováděných akcí. Samozřejmě s každým novým modulem mohou vznikat další potřeby pro specifická oprávnění. Na která oprávnění lze definovat píplekladu aplikace, na která se mohou mnit za bhu. Příklad procesu kontroly pravomocí je popsán diagramem 3.2.



Obrázek 3.2: Stavový diagram kontroly pravomocí, převzato z [35]

Přidávat každému uživateli tato oprávnění zvlášť by bylo velmi neefektivní a z toho důvodu vznikly tzv. uživatelské skupiny (občas nazývané také jako role). Jednotlivým rolím lze, v závislosti na systému, nadefinovat různou detailnost jejich práv. Uživatelé jsou následně rozděleni do předem definovaných skupin, čímž se docílí lepší míry abstrakce a následně jde snáze jednotlivým skupinám přidávat a mnit pravomoce v rámci systému. Toto se velmi hodí například na fóra, kde zpravidla bývá většinou neregistrovaný návštěvník jen v roli čtenáře a až po registraci může začít přidávat obsah, či jinak s již publikovanými články interagovat. Posune-li se do role administrátora, může pak provádět další operace nad obsahem či uživateli, jako jsou například mazání, editace, atd.

Systémy často pro zjednodušení úvodního nastavení obsahují již pár předdefinovaných rolí, a to včetně jejich pravomocí (viz obrázek 3.3). Mezi typické základní role by šlo zařadit například neregistrovaného uživatele, autora, editora či administrátora. Na které systémy nahlíží na jednotlivé pravomoce hierarchicky – čtenář má menší práva než editor, ale pokud je uživatel již definovaný jako editor, získává automaticky i pravomoce čtenáře... Jiné zase řeší danou problematiku více do detailu a to tak, že definují pro každou roli oprávnění zvlášť, čímž eliminují kaskádovou závislost pravomocí. U některých systémů může také

docházet k přidání více rolí jednomu uživateli, čímž dochází ke kombinaci pravomocí. Role se mohou hodit i z bezpečnostního hlediska. Například pokud dojde k nepovolené změně dat, dá se snadno vyfiltrovat potenciální viníky, dle skupiny, která má k daným datům přístup. Autorizace lze kromě přidání práv jednotlivým rolím provádět také na základě zápisu nebo lokace.

Asset Title	Asset Name	Site Login	Administrator Login	Web Services Login	Offline Access	Super User	Configure Options Only	Access Administration Interface	Create	Delete	Edit	Edit State	Edit Own	Edit Custom Field Value	LFT	ID
Root Asset	root.1	✓	✓	✗	✓	✗	✗	✓	✓	✓	✓	✓	✓	✗	0-167	1
System Information	-com_admin	✓	✓	✗	✓	✗	✗	✓	✓	✓	✓	✓	✓	✗	1-2	2
Banners	-com_banners	✓	✓	✗	✓	✓	✗	✓	✓	✓	✓	✓	✓	✗	3-6	3
Uncategorised	-com_banners.category.3	✓	✓	✗	✓	✓	✗	✓	✓	✓	✓	✓	✓	✗	4-5	28
Cache	-com_cache	✓	✓	✗	✓	✓	✗	✓	✓	✓	✓	✓	✓	✗	7-8	4
Check-in	-com_checkin	✓	✓	✗	✓	✓	✗	✓	✓	✓	✓	✓	✓	✗	9-10	5
Configuration	-com_config	✓	✓	✗	✓	✗	✗	✓	✓	✓	✓	✓	✓	✗	11-12	6
Contacts	-com_contact	✓	✓	✗	✓	✓	✗	✓	✓	✓	✓	✓	✓	✗	13-16	7
Uncategorised	-com_contact.category.4	✓	✓	✗	✓	✓	✗	✓	✓	✓	✓	✓	✓	✗	14-15	29
Articles	-com_content	✓	✓	✗	✓	✓	✗	✓	✓	✓	✓	✓	✓	✗	17-38	8

Obrázek 3.3: Tabulka správy pravomocí pro roli Administrator v systému Joomla!

### 3.4 Bezpečnost

Jelikož jsou CMS často využívány jako veřejné systémy dostupné z internetu, je potřeba na zabezpečení klást o to větší důraz, jelikož k nim má přístup velké množství potenciálních útočníků. Aby se dala všeobecně webová aplikace považovat za bezpečnou, musí splňovat základní 4 kritéria zmíněná na začátku kapitoly 2.3. Bezpečnost systému také do značné míry ovlivňuje frekvence publikování zpráv o zranitelnostech a včasnost jejich bezpečnostních záplat a také typy na protiopatření. Jako často užívané útoky lze dle [34] zmínit:

- manipulace s parametry,
- skriptování na stránkách (cross-site scripting),
- SQL injection,
- autentizační útoky,
- nahrání nebezpečného souboru,
- eskalace oprávnění (privilege Elevation),
- SPAM.

Pod manipulací s parametry si lze představit útoky na superglobální proměnné, *cookie poisoning*, vkládání příkazů do HTTP hlaviček nebo manipulaci s formuláři ovými daty. Dalším problémem může být tzv. *Cross-site scripting (XSS)*, neboli zasílání skriptů ze serveru na klienta. Pokud neprobíhá na serveru dostatečná kontrola, může dojít k spuštění javascriptových příkazů, které jsou schopny poškodit klienta.

Velmi známým útokem je také *SQL injection*, kde se útočník snaží do dat odesílaných na server do databáze vložit specifické textové řetězce schopné infikovat či vrátit utajovaná data. Takový textový řetězec může mít například tvar `OR '1'='1'`, který když se vloží na konec SQL dotazu způsobí kladné vyhodnocení dotazové podmínky.

Systém musí provádět nejen autentizaci při přihlášení, ale musí také docházet k autorizaci při přístupu do nejrůznějších částí systému. Tento mechanismus se často implementuje pomocí serverových proměnných – *sessions*.

Nahrávání souborů na server může být velmi nebezpečné z důvodu používání vícenásobných přípon. Například pokud by se útočník snažil dostat na server spustitelný exe soubor, ale připojil na konec příponu .jpg, nemuselo by dojít k jeho kontrole, jelikož by si server myslel, že se jedná o obrázek. Což by samozřejmě mohlo vést k velkým potenciálním problémům.

Eskalace privilegií je typ útoku, díky kterému lze získat neoprávněný přístup do zabezpečených částí systému. Útočník šplhá po pomyslném žebříku od nejméně oprávněného úrovně po nejvíce, dokud nedojde do požadované pozice. Útok přímou na nejvyšší pozici by byl pravděpodobně neúspěšný, tak se využívá chyb v systému a postupně se propracovává nahoru.

Útočník se může také pokusit zhostit serveru, aby mu sloužil jako uzel pro posílání SPAM mailů. Nejedná se nezbytně o útok poškozující data nebo infrastrukturu samotného serveru, ale i tak je třeba se takovýmto útokem bránit. Nejčastěji bývá implementována ochrana *CAPTCHA*, která je schopna rozpoznat, zda se jedná o člověka nebo robota [34].

## 3.5 Současně open-source systémy

Je složitý proces vývoje CMS, zůstává v současnosti většina velkých platform open source. Výhodou open source projektu je potřeba kapitál nutný k jejich rozvoji, možnost vytvoření vlastních rozšíření, případně instalace již existujících a plná kontrola nad zdrojovými kódy. Pro potřeby průzkumu modularizace a user groups byly použity výhradně open-source projekty, jelikož se dá prozkoumat jejich vnitřní implementace daných funkcí. Podle průzkumu portálu W3techs patří mezi nejpoužívanější open source systémy Wordpress<sup>1</sup>, Joomla<sup>2</sup> a Drupal<sup>3</sup> [43]. Pro porovnání byl také vybrán systém Dokuwiki. Není sice tak oblíbený jako zbylé CMS, ale při průzkumu byl využit pro alternativní pohled na správu dat – ukládající data do souborů namísto do databáze. Vybraná sada analyzovaných systémů je tedy: Wordpress, Drupal, Joomla a Dokuwiki<sup>4</sup>.

### 3.5.1 Wordpress

Wordpress je v současnosti nejznámější a nejpoužívanější open-source CMS. Vznikl v roce 2003, kdy byla podle jeho autorů nezbytná potřeba elegantního a dobře strukturovaného systému pro správu obsahu. Wordpress je postavený na skriptovacím jazyce PHP a z po-

<sup>1</sup>[www.wordpress.org](http://www.wordpress.org)

<sup>2</sup>[www.joomla.org](http://www.joomla.org)

<sup>3</sup>[www.drupal.org](http://www.drupal.org)

<sup>4</sup>[www.dokuwiki.org](http://www.dokuwiki.org)

hledu databáze využívá MySQL. V listopadu 2022 byl využíván na 43 % všech veřejných webových stránek [44].

Velkou výhodou Wordpressu je jeho rozšířenost. Díky tomu existuje mnoho, a již place-ných, i nezaplatných, rozšíření, které velmi urychlují vývoj webových stránek a aplikací a usnadní jejich údržbu. Není tedy problém poměrně rychle přeměnit statickou stránku například na e-shop, nebo blog. Další nespornou výhodou je rozsáhlá komunita, která se snaží poradit a poradá také různé vzdělávací události.

Jeho rozšířenost přitahuje větší množství útočníků snažících se najít skulinu v zabezpečení. Když se útočníkovi podaří najít chybu v zabezpečení systému, je pak schopný zneužít danou nedokonalost na velké množství stránek. Komunita vývojářů je si tohoto ovšem vědoma a neustále se pracuje na aktualizacích, které opravují nejrizičnější potenciální problémy.

### 3.5.2 Joomla!

Joomla! je bezplatný open-source redakční systém vyvinutý v roce 2005 pro publikování webového obsahu. Systém je založený na MVC<sup>5</sup> architektuře a je postavený na klasické kombinaci PHP a MySQL. Pravidelně se umísťuje v žebříčcích CMS Critic People's Choice Awards jako nejlepší bezplatný CMS [27].

Jako výhoda systému Joomla! by se dala vzít její míra flexibility. Vhodná je spíše pro větší a komplexnější projekty, ale přesto nevyžaduje znalost programování. Na rozdíl od Wordpressu nabízí detailnější správu uživatelských rolí a možnost použití více šablon současně (zná šablona pro určitý obsah) [25].

Jelikož není Joomla tak rozšířená, jak Wordpress, nabízí více než šestinásobně méně rozšíření zalistovaných v oficiálních katalogích. Co se týče zabezpečení, pak se dá za výhodu brát zabudované SSL v jádru systému. Nicméně i skrze tuto vlastnost je procentuální napadnutelnost stránek postavených na Joomla cca o 132 % větší, než její podíl na trhu [25]. Dalším problémem se může jevit sestup v oblíbenosti, který v posledních letech Joomla zažívá. To do budoucna může znamenat nedostatečný vývoj a údržbu jak samotného systému, tak i rozšíření z těchto stran, což může omezit její konkurenceschopnost.

### 3.5.3 Drupal

Drupal je v současnosti jedním z nejpoužívanějších bezplatných open-source CMS. Byl vytvořen v roce 2001 bývalým studentem Antverpské univerzity Driesem Buytaertem, tím pádem se jedná o nejstarší projekt z této kategorie porovnávaných. Jak tomu je u velkého množství CMS, tak je také i Drupal postavený na kombinaci PHP a MySQL. [17]

Kromě zabudované možnosti modularizace a detailní správy uživatelských rolí a jejich pravomocí, se dá za přednost Drupalu považovat jeho bezpečnost. Průzkum z roku 2016 tvrdí, že pouze 2 % ze zkoumaných webových stránek postavených na Drupalu bylo infikováno [26]. Díky této vlastnosti je Drupal oblíbenou variantou pro velké a vládní instituce (například česká televize, Prima nebo vládní stránky Rhode Islandu v USA).

Na první pohled by se mohlo zdát, že má Drupal velké množství modulů pro rozšíření funkcionality, existuje zde ale bohužel problém s jejich aktuálností. Z celkového množství cca 39 000 oficiálních modulů je jen lehce přes 10 % z nich kompatibilních s verzí 8 a vyššími. Když se k tomu přidá uživatelé často zmiňovaná strmá křivka uhlavování, může z toho vyjít poměrně složitá situace pro neprogramující uživatele [26].

---

<sup>5</sup>Model-view-controller

### 3.5.4 Dokuwiki

Dokuwiki je univerzální odleh ený open-source wiki software, který vznikl v roce 2004. Nejedná se primárn ě od CMS, ale díky možnosti správy uživatelských rolí a schopnosti rozší ení o moduly jej lze tak nastavit. Dokuwiki je postavena na PHP, ovšem co je oproti výše zmín ěným rozdílné, je to, že ke svému běhu nevyžaduje databázi. Pot ebné konfigurace a data jsou ukládány do stromové struktury ve form ě soubor ě. Díky tomu je levn ější na hosting a jednodušší na po ěáte ní nastavení. To z ní d ělá oblíbené light-weight ešení [21].

Bohužel tím, že se nejedná primárn ě o CMS, tak na Dokuwiki existuje v oficiálním seznamu pouze p ěs 1 340 balí k ě s rozší eními, což je násobn ě mén ě, než u výše zmín ěných systém ě [5].

## Kapitola 4

# Analýza současného stavu

Před samotným vývojem došlo k provedení úvodní analýzy současného stavu a na jejím základě jsou následně v další kapitole navrženy změny a zlepšení. V této kapitole bude rozebráno současné fungování platformy SmartCity od firmy Logimic<sup>1</sup> a bude kladen velký důraz na zpětnou vazbu obdrženu od editore firmy Michala Valného a firemního vývojáře Jiřího Hynka, který je zároveň vedoucím této práce.

### 4.1 Logimic a SmartCity

Firma Logimic, s.r.o. je česká společnost poskytující cloudová IoT řešení pro města a průmysl. Zabývá se také bezdrátovým přenosem dat prostřednictvím IQRF technologie, propojení průmyslových zařízení do cloudu a vývojem software a hardware řešení. Firma má zákazníky převážně z Evropy, ale spolupracuje také i s americkými městy, tudíž je kladen důraz na multijazyčné řešení. Ke dni psaní této zprávy má přes 30 zákazníků a přes 1 000 zapojených koncových zařízení.

Hlavním produktem firmy je platforma pro správu chytrých měst pojmenovaná Smart-City. Jedná se o cloudovou platformu (informační systém) umožňující připojení a monitoring koncových zařízení bez nebo s minimem programátorských dovedností. Platforma je vybavena webovým rozhraním a mobilní aplikací. Jejím cílem je umožnit rychlé vytvoření IoT aplikace nad jakýmkoliv koncovými zařízeními [31]. V platformě se nachází klíčové metriky a ukazatele jednotlivých připojených senzorů, možnosti nastavení upozornění, plánování servisních požadavků, výpis výkazů a další funkce nezbytné pro monitoring a správu IoT zařízení.

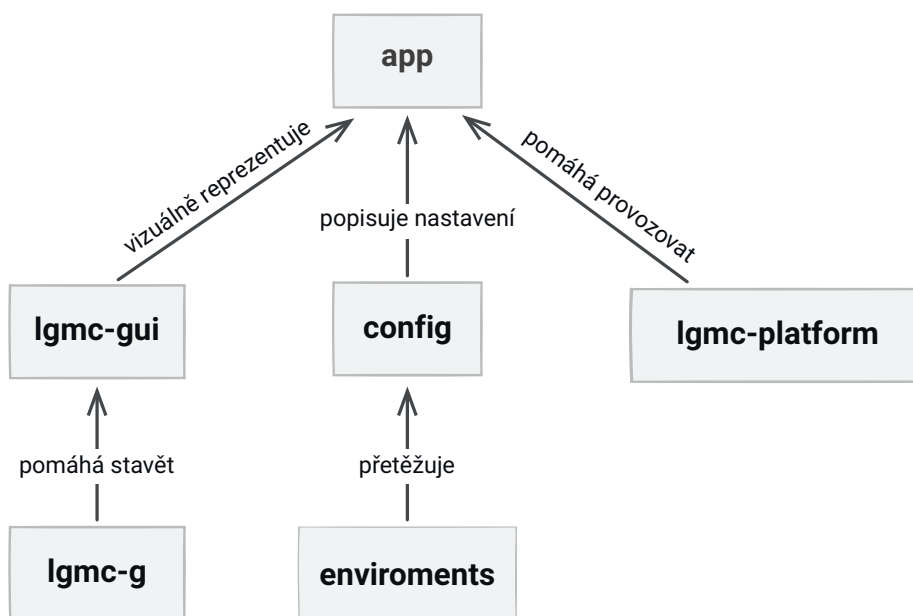
Platforma SmartCity je aplikací dodávanou zákazníkům firmy. Pro každého zákazníka je zřízena vlastní databáze a hosting. Hlavním úkolem aplikace je připojení různých IoT zařízení a jejich následná správa. Aby byla správa jednodušší, existuje zde možnost shlukovat zařízení do skupin a následně je zobrazovat v dashboardu s klíčovými ukazateli. Pokud by zákazník vyžadoval ještě další úroveň shlukování pro efektivnější správu, existují uvnitř SmartCity entity *enterprise* a *system*. Shlukování je efektivním způsobem řízení pravomocí a delegace práce. Například zákazník v domácnosti, kde se typicky vyskytuje méně zařízení a není třeba řešit uživatelské role bude mít ve SmartCity pouze 1 systém bez *enterprise*. Na druhou stranu nějaká vnitřní korporace s velkým množstvím zařízení bude moci rozdělit na několik systémů a množiny systémů přidat pod jednotlivá *enterprise*. Díky tomu bude korporace schopna přidat jednotlivým oddělením role (například systémový

<sup>1</sup>[www.logimic.com](http://www.logimic.com)

administrátor i *enterprise* administrátor) a takto rozdelit správu zařízením na povolené osoby.

Platforma je postavena na skriptovacím jazyce Typescript<sup>2</sup>, který je založený na jazyku JavaScript. Přináší do něj ovšem vylepšení o statickou typovou kontrolu, aby došlo k minimalizaci chyb. Hodí se jak pro vývoj klientských, tak i serverových řešení. Pro samotnou implementaci klientské části platformy byl použit rámec Angular<sup>3</sup> vyvinut společností Google. Jedná se o multiplatformní rámec založený na komponentách a službách, využívající architekturu MVC, obousměrnou datovou vazbu a injekce závislostí. Na pozadí platformy běží open-source objektová relační databáze PostgreSQL<sup>4</sup>, do které jsou ukládána data vyžadující persistenci. Jelikož je platforma využívána z různých částí světa, běží na cloudovém řešení AWS<sup>5</sup> od firmy Amazon. Díky tomu je systém schopný reagovat s nízkou latencí, vysokou bezpečností a dá se lehce škálovat.

Architektura platformy SmartCity je vyobrazena na obrázku 4.1. Platforma je sestavena z množiny modulů, jež má každý svůj definovaný význam a reprezentuje určitou funkcionalitu (např. notifikace, zobrazení dashboardu i administrátorský modul pro správu nastavení). Tyto moduly, obsažené v části `lgmc-gui`, slouží pro vizuální reprezentaci aplikací a pro interakci uživatele s aplikací. Jednotlivé části modulů jsou, mimo jiné, postaveny pomocí univerzálních grafických komponent nacházejících se v `lgmc-g`. Pod grafickými komponentami si lze představit například dialogová okna, ukazatele na číselníky i tlačítka.



Obrázek 4.1: Architektura platformy SmartCity

Nastavení aplikace se provádí programově v části zvané `config`. Lze tam najít konfigurace jak pro jednotlivé moduly platformy, tak i například nastavení vizuálních stylů, jazyka

<sup>2</sup>[www.typescriptlang.org](http://www.typescriptlang.org)

<sup>3</sup>[www.angular.io](http://www.angular.io)

<sup>4</sup>[www.postgresql.org](http://www.postgresql.org)

<sup>5</sup>[www.aws.amazon.com](http://www.aws.amazon.com)

nebo cest k obrázkům. Množina těchto konfigurací bude dále v textu referována jako statická aplikační konfigurace. Statická z toho důvodu, že pro každé její změny musí dojít k optimálnímu překladu zdrojových kódů a nelze tudíž měnit za běhu. Nicméně každý zákazník může mít vlastní požadavky na drobné úpravy vzhledu i funkcionality. Z toho důvodu dochází k přetížení statické aplikační konfigurace konfigurací obsaženou v rámci prostředí. Tomuto přetížení nastavení se dále v práci říká statická konfigurace prostředí.

Další nastavení týkající se platformy (přístupy, jednotky, autentizace, API a další) jsou definovány v `lgmc-platform`. Tato část obsahuje také funkce obsluhující aplikační jádro, čímž pomáhá řídit chod celé aplikace. App pak obsahuje jen, dle standardní Angular architektury, základní aplikační modul, ze kterého je odkazováno na ostatní moduly a konfigurace tvořící aplikaci.

## 4.2 Analýza uživatelů a definice problémů

Po konzultaci s vývojáři platformy bylo zjištěno, že s platformou pracují zpravidla 3 typy lidí:

- Vývojář – má na starosti korektní chod aplikace a kromě vytváření rozšíření, řešení chyb a plánování změn, nastavuje statickou konfiguraci aplikace i prostředí. Vývojář zná jednotlivé souvislosti a je schopen měnit prakticky celý chod platformy. Problémem u vývojáře je ten, že jakákoliv změna je pomalá a drahá, jelikož musí ručně přepisovat kód a mít znalost programových závislostí. V souasných chvílích je vývojář naplňovat své povinnosti v plném rozsahu, avšak v rámci analýzy platformy došlo k zjištění určitých nedostatků. Hlavním problémem byla nejednotná struktura konfigurací (převážně pak části s moduly), a tudíž by bylo velmi obtížné opravit funkční generické UI. Ze strany firmy Logimic došlo během psaní práce k ústřední změně architektury týkající se modulů a také k nasazení rozhraní na moduly, čímž se sjednotila jejich základní struktura.
- Správce IoT systému – jedná se o uživatelskou roli v rámci platformy (nazývanou také jako administrátor). V souasných chvílích nemá možnost nijak uživatelsky zasahovat do nastavení konfigurací, a tedy je tato práce ponechána na vývojáři. Jeho motivací do budoucna je přepisování vývojáři staticky definovaných konfigurací tak, aby přizpůsobil aplikaci zákaznickým požadavkům. Mohlo by se jednat například o nastavení stavů různých modulů, změny jejich obsahu, připojení nových zařízení i o správu uživatelů. Výstupem administrátorských změn bude stav nazývaný dynamická aplikační konfigurace. Dynamická z toho důvodu, že bude měnit za běhu aplikace a bez zásahů do zdrojových kódů programu. Tyto změny budou ovlivňovat všechny uživatele v rámci dané aplikace. V praxi roli administrátora může zastávat například zaměstnanec IT oddělení nebo přední správce z firmy Logimic.
- Uživatel IoT systému – uživatelská role platformy, která má nejnižší práva. Jeho motivací je si v nastavení změnit konfiguraci tak, aby mu aplikace více vyhovovala a lépe se používala. Za jeho interakci s nastavením by se převážně daly brát vizuální změny jako třeba změna ikon, barev nebo rozložení. Tím, že je vše řešeno staticky uvnitř kódu, nemůže ani uživatel nijak zasahovat do konfigurace aplikace. Zbývá mu tím pádem více pasivní role pozorovatele indikátorů a dashboardů. Do budoucna je žádoucí dovolit uživateli měnit určité nastavení. Tyto uživatelské změny budou dále referovány jako dynamická uživatelská konfigurace a budou ovlivňovat pouze

konkrétního uživatele. V domácích podmínkách jsou role administrátora a uživatele často spojeny.

Z analýzy uživatelů vyplynulo, že v platformě chybí nástroj pro správu nastavení modulů a zbylých aplikačních konfigurací. Problémem současně je pomalá změna nastavení, jelikož musí dojít k zásahu do programu a takovouto změnu lze provést pouze vývojář. Po zanesení změny musí dojít k opětovnému přeložení programu a kompilaci, čímž se platforma stává na určitou dobu nedostupnou a nelze dojít i ke ztrátě rozpracovaných dat. Problémem je také absence možnosti změny dat přímo z aplikace a za jejího běhu. Kvůli této skutečnosti nemůže naplňovat ani administrátor ani uživatel své požadované akce, jelikož nemají přístup ke kódu aplikace. Kdyby existoval v rámci platformy nástroj umožňující změny konfigurace uživatelsky přístupivým způsobem, mohlo by dojít k delegování vývojářské práce na zbytek uživatelů a tím by se vývojáři zbavili nekvalifikované práce a mohli se jás lépe soustředit na další vývoj.

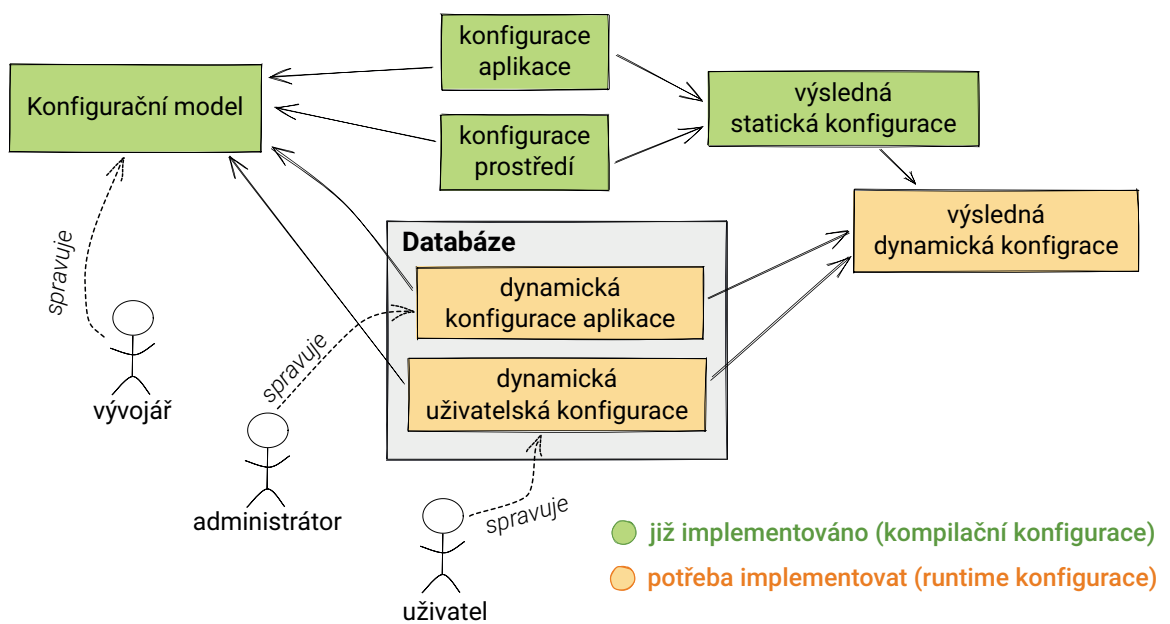
### 4.3 Požadavky na změnu

Firmě by pomohlo, kdyby v rámci platformy došlo ke vzniku nástroje, který by umožnil administrátorům a uživatelům zasahovat do nastavení modulů a zbytku aplikace, a to za jejího běhu. Tím, že je funkcionality platformy rozdělena na několik modulů, je žádoucí, aby i tento nástroj byl implementován formou rozšiřujícího modulu. Tento modul by měl poskytovat možnost změny nastavení uživatelsky přístupivým způsobem – tedy grafická komponenta, která by se dala v aplikaci zobrazit a interagovat s ní. Dále je potřeba, aby se změna nastavení projevila do chodu platformy. Je nezbytné rozlišovat mezi dynamicky vytvořenou aplikační a uživatelskou konfigurací. Aplikační konfiguraci by měla mít na starost administrátor a měla by ovlivnit všechny uživatele v rámci dané aplikace. Uživatelská konfigurace se má vztahovat pouze k jednomu konkrétnímu uživateli. Dále je potřeba zajistit určitou kontrolu, jaká data je povoleno měnit, aby nedošlo k porušení chodu celé aplikace (například nesmí dojít ke změně API, jelikož by pak nemusela fungovat správná komunikace se serverovou částí aplikace).

Dynamicky vytvořené změny je třeba uchovávat v již nasazené databázi, aby došlo k perzistentnímu uchování změných dat. Při vývoji by měla být kladen důraz na požadavky vývojáře i administrátora tak, aby došlo k co možná nejvyššímu zefektivnění práce. Firma dále chce pomocí dynamické konfigurace co nejvíce eliminovat nutnost vytváření různých produktů v tví pro jednotlivé zákazníky, ve chvíli, kdy chtějí měnit určitá nastavení. Rozdílnost struktur jednotlivých konfigurací klade velký důraz na vytvoření generického řešení schopného zpracovávat nejen atomická data (text, čísla a pravdivostní hodnoty) ale i zanořené objekty a pole.

A v neposlední řadě je preferováno dodržení vizuální identity celé platformy, která prošla minulý rok kompletní změnou popsanou v bakalářské práci [10]. I přes to, že se platforma nejpohodlněji využívá na počítači, díky zvoleným technologiím a responzivnímu designu je přenositelná i na mobilní zařízení s menší obrazovkou. Přenositelnost a responzivitu je třeba zachovat i v nově vznikajícím rozšiřujícím modulu.

Na obrázku 4.2 je ilustrována současná situace konfigurací a požadovaná změna. Zeleň zvýrazněné části vykreslují současný stav aplikace – statickou konfiguraci vytvořenou vývojářem přímo v kódu. Dynamicky vytvořenou konfiguraci, kterou znázorňuje oranžová část obrázku, bude třeba doimplementovat. Nakonec bude ještě potřeba, před samotnou integrací, provést testování, zda změny nijak nenarušují chod a funkčnost platformy.



Obrázek 4.2: Vysokourovňové schéma požadované architektury modularizace a dynamické konfigurace. Při překladu a kompilaci aplikace dochází nejprve k nastavení statické konfigurace aplikace a jejího následného načtení pomocí konfigurace prostředí obsahující specifické detailní nastavení pro daného klienta. V rámci rozšíření je třeba načíst načtení dynamickou aplikací konfigurací a následně dynamickou uživatelskou konfigurací, čímž vznikne výsledná dynamická konfigurace.

# Kapitola 5

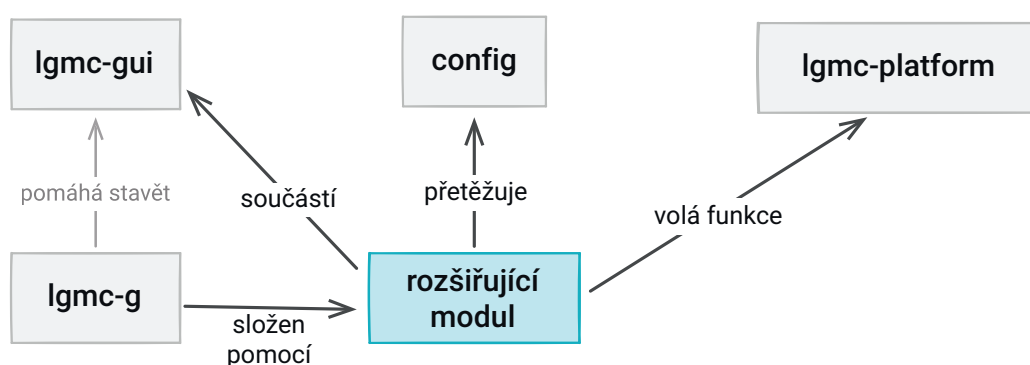
## Návrh

Po provedené analýze a před samotnou implementací došlo k nastavení návrhu řešení, které je popsáno v této kapitole. Jednotlivé podkapitoly popisují a prakticky demonstrují nezbytné návrhové části jako jsou: architektura, koncepce datového modelu a návrh uživatelského rozhraní.

### 5.1 Architektura rozšíření

Z požadavků na změny, zmíněných v rámci analýzy, vyplývá, že je nezbytné vytvořit nástroj pro správu konfigurací a nastavení modulů. V rámci zachování modularizace platformy bude tento nástroj ve formě rozšiřujícího modulu umístěn mezi ostatními moduly tak, aby zapadl do celkové architektury platformy zobrazené na obrázku 4.1. Tento modul má umožňovat změny konfigurací přímo z aplikace a má být uživatelsky přívětivý. Pro zobrazení nastavení konfiguračních částí budou v modulu grafické komponenty. Tyto komponenty budou vycházet z návrhu obrazovek uživatelského rozhraní detailně rozebraném v kapitole 5.3. Aby se dala zobrazovat data genericky bez předěšlé znalosti jejich struktury, bude vizuální stránka reprezentovaná dynamickým formulářovým zobrazením a každá položka bude obsahovat vstup pro změnu dat.

Na obrázku 5.1 je demonstrován vztah rozšiřujícího modulu se zbytkem platformy. Toto schéma vychází z architektury platformy znázorněné na obrázku 4.1. Rozšiřující modul se



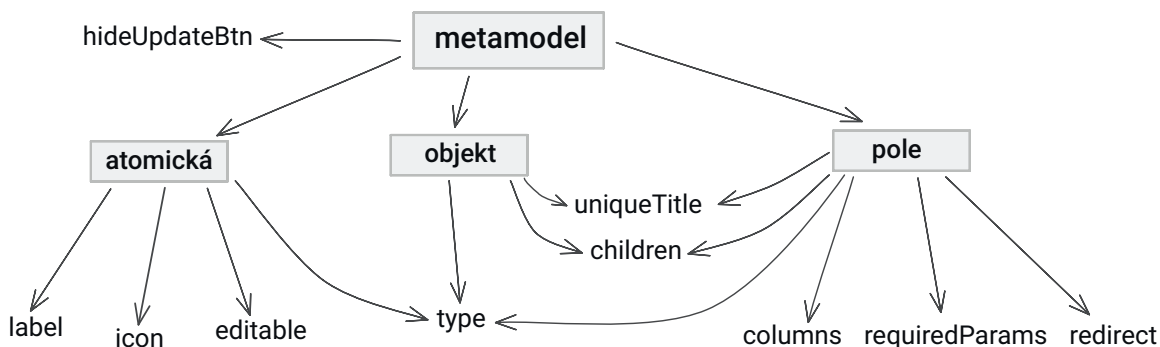
Obrázek 5.1: Architektura rozšíření v kontextu architektury SmartCity



metamodel. Metamodely budou rozdílné komplexnosti a složitosti a budou se skládat z kombinací následujících vlastností:

- `type` – typ zobrazení UI. Existují 4 typy: `form`, `table`, `detailtable` a `mixed`. Zobrazení typu `form` slouží pro formuláře na nejnižší úrovni, nebo pro objekty obsahující pouze atomické hodnoty. Jeho použití reprezentuje obrázek 5.6. `Table` slouží pro vykreslení tabulky obsahující zanořené objekty (viz obrázek 5.5). `Detailtable` je tabulka obsahující, oproti `table`, informace navíc a slouží pro vykreslení hodnot pole. V rámci nastavení `columns` lze definovat jaké sloupce budou u každého řádku detailní tabulky zobrazeny. Její návrh lze vidět na obrázku 5.7. Poslední typ `mixed` vznikl kvůli objektům skládajícím se jak z atomických dat, tak i zanořených a jedná se o hybridní zobrazení mezi `form` a `table` (dojde k vykreslení obou).
- `editable` – číselná hodnota značící, jaká role má danou vlastnost mít. V aktuální fázi byly zavedeny 4 hodnoty (0 pro vývojáře, 10 pro administrátora, 100 pro uživatele a 1 000 bez práv změny). S rostoucí hodnotou pravomoce na změnu klesají.
- `label` a `icon` – `label` slouží jako štítek popisující aktuální hodnoty zobrazené ve formulářích. Je používán ve formátu jazykových souborů, aby mohlo docházet ke správným překladům na základě aktuálně zvoleného jazyka. V určitých případech je žádoucí zobrazit místo `label` u ikonu, z toho důvodu je u některých atributů zadefinována. Pokud je v metamodelu uvedena ikona, je vždy přidávána navíc k `label` u, aby existoval *fallback* v případě, že se ikona správně nenajde.
- `hideUpdateBtn` – pravdivostní příznak, zda má v detailním zobrazení být viditelné tlačítko pro aktualizaci dat. V rámci konfigurací existují části, které je žádoucí zobrazovat, nicméně však mít jejich hodnoty (např. API nebo autentizace). Do formuláře nelze zapisovat, nemá-li aktuální uživatel dostatečná práva, ale nevykreslení tlačítka je dalším opatřením k zamezení změny dat a zamezí také zmatení uživatelů.
- `children` – Obsahuje-li objekt v rámci své struktury další zanořený objekt, pak jsou tato data v metamodelu obalena do `children`. Hlavním efektem je rozlišení atomických a vrstvených hodnot, aby mohlo dojít k jejich odlišení a následnému správnému vykreslení v rámci UI.
- `uniqueTitle` – ne vždy jsou názvy objektů v tabulce, nebo v záložkách zřejmé, a tak byla potřeba, aby vznikla položka, která bude řešit jejich korektní pojmenování. Není například velmi vhodné používat v rámci tabulky označení řádků jako indexy, nebo názvy programových proměnných. Díky `uniqueTitle` se tento problém vyřeší, a navíc lze i provést jazykovou mutaci. Tento parametr používají pouze tabulková zobrazení.
- `columns` – spolupracuje s `type: detailtable` a slouží pro definici, jaké sloupce mají být zobrazeny v rámci tabulky. Jedná se o pole textových řetězců.
- `requiredParams` – také úzce spolupracuje s `type: detailtable` a jeho význam je pro definici, které vlastnosti musí nově přidávaná položka do pole obsahovat. Opět se jedná o pole textových řetězců na jejichž základě je vykreslený *pop-up* formulář při přidávání nového řádku.
- `redirect` – slouží pro dodatečné přesměrování na správnou cestu k datům objektu. Jeho využití je spíše vzácné a používá se pouze u polí jejichž vyobrazení je řešeno pomocí `type: detailtable`.

Obrázek 5.3 slouží k ilustraci, pro jaké datové typy se používají jaké parametry metamodelu. Na obrázku jsou znázorněny 3 typy dat: atomická, objekty a pole. Tyto typy jsou dostupné pro zobrazení všech konfigurací nastavení. Ke změně metamodelu nemusí docházet dynamicky za běhu aplikace.



Obrázek 5.3: Rozdělení, kterou vlastnost metamodelu má, že jaký typ dat využívat

### 5.3 Uživatelské rozhraní

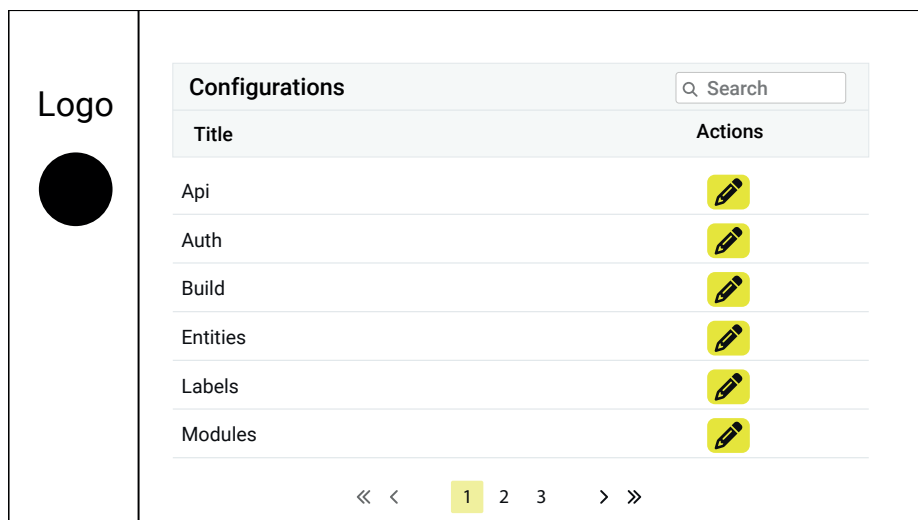
Při navrhování uživatelského rozhraní (zkráceně UI) byl kladen důraz na dodržení vizuální identity celé platformy tak, aby rozšíření nijak nevybočovalo a nemátló uživatele. Návrh uživatelského rozhraní se skládá ze 4 částí. Jelikož se dopředu nedá určit, kolik úrovní vnořených dat požadují jednotlivé konfigurace zobrazit, je UI vytvořeno tak, aby se pomocí něj dala vykreslit všechna potřebná dynamická data. Hlavními prvky jsou tabulka a formulář. Tabulka slouží pro výpis výčtu jednotlivých nastavení, které obsahují strukturovaná data (objekty). Formulář je následně určen pro atomická data jako jsou například textové a číselné a pravdivostní hodnoty.

Na obrázku 5.4 lze vidět vyobrazení úvodní obrazovky po rozkliknutí navrhovaného modulu pro nastavení konfigurací. Tato část vyzývá uživatele k výběru konfigurace, jejíž nastavení má zájem upravovat. Kvůli velkému množství existujících konfigurací částí, ale také vývojem nově vznikajícím, bylo zvoleno zobrazení ve formě tabulky. Velkou roli hrálo i to, že se jedná o objekty.

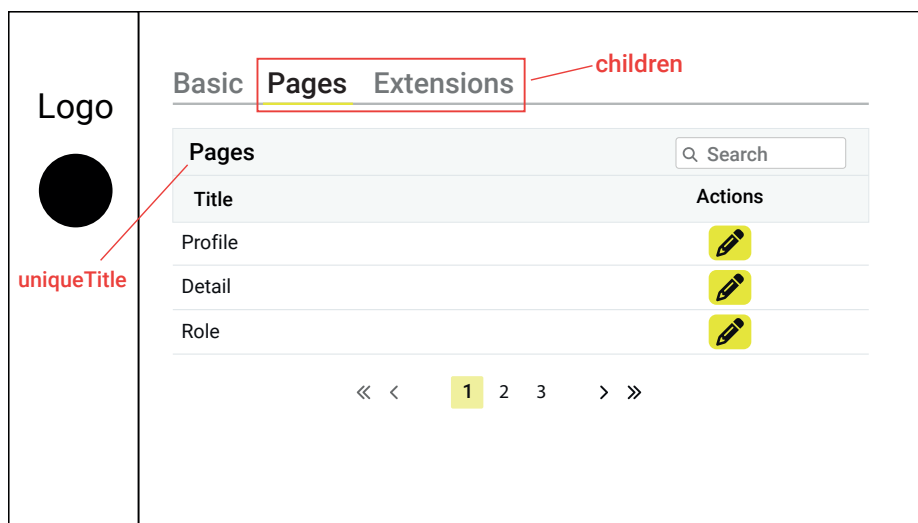
Po rozkliknutí detailu jedné z konfigurací dojde k zobrazení jejího nastavení, které je navrženo dle obrázku 5.5. Kartu *Basic* obsahuje každá konfigurací část. Jsou v ní obsažena všechna atomická data na aktuální úrovni zanoření (například, aktivní příznak a název). Ostatní záložky – existují-li – jsou generovány dynamicky ze strukturovaných objektů na aktuální úrovni daného modelu konfigurace. Na obrázku 5.5 je zmíněna konfigurace modulu *admin* obsahující dynamické karty *pages* a *extensions*. Jednotlivé karty budou zobrazovány pomocí grafické komponenty, která zajišťuje responzivní zobrazení. Na mobilních zařízeních dojde ke změně rozložení. Karty nebudou vedle sebe, ale pod sebou ve formě tzv. *accordion* zobrazení.

Po rozkliknutí jednotlivého detailu z tabulky nastavení dojde k vykreslení obrazovky na detail konkrétního strukturovaného nastavení. Po rozkliknutí detailu může dojít k vykreslení další tabulky, detailní tabulky (viz obrázek 5.7), formuláře (viz obrázek 5.6) a jejich mixu. Typ vykreslení je dán metamodelem dané konfigurace. Neobsahuje-li konfigurace

metamodel, neví program, jak daná data zobrazit, a tudíž dojde k mixovanému vykreslení. Formulář se využívá zpravidla pro zobrazení dat na nejnižších úrovních zanoření a slouží ke změně dat. Dalo by se tedy říct, že se jedná o klíčové zobrazení a aby k němu byl přístup, je občas třeba využít tabulky.



Obrázek 5.4: Mockup seznamu konfigurací



Obrázek 5.5: Mockup detailu konfigurace modulu. Červené popisky ilustrují jednotlivé vlastnosti metamodelu v praxi. Tento mockup zobrazuje type: table

**Logo**

**kpiCard**

Enabled

Description: lgmc-gui-kpi.dashboard-card.summary.description

Icon: pi pi-bell

Icon color: #bada55

Title: lgmc-gui-kpi.dashboard-card.summary.title

Card showing key indicator

Annotations: label, icon, editable, hideUpdateBtn

Obrázek 5.6: Mockup konkrétního nastavení konfigurace na nejnižší vrstvě zanoření. červené popisky ilustrují jednotlivé vlastnosti metamodelu v praxi. Tento mockup zobrazuje type: form. Parametr icon je pro ilustraci zakázaný mnit, z toho důvodu je vizuálně odlišen

**Logo**

**Basic**

Pages   **columns**

Title	path	icon	Actions
Profile	admin/profile	pi pi-briefcase	
Detail	detail	pi pi-info	
Role	admin/role	pi pi-user	

1 2 3 > >>

Annotation: redirect

Obrázek 5.7: Mockup detailní tabulky konfigurace modulu obsahující pole hodnot. Ikony umožňující smazání řádku budou obsahovat pouze dynamicky přidávané řádky pole. červené popisky ilustrují jednotlivé vlastnosti metamodelu v praxi. Tento mockup zobrazuje type: detail-table

## Kapitola 6

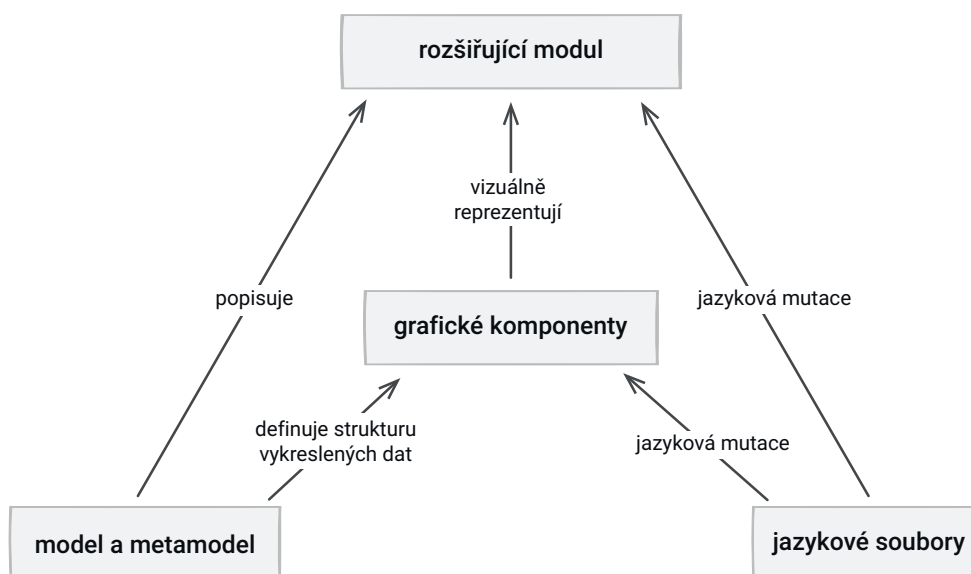
# Implementace

Tato část se zaměřuje na praktickou stránku projektu a podrobně popisuje, jak bylo řešení implementováno pomocí různých nástrojů a technologií. Tato kapitola se pokusí nastínit problémy, které se v rámci procesu vývoje vyskytly a ukázat jejich řešení.

### 6.1 Architektura rozšiřujícího modulu

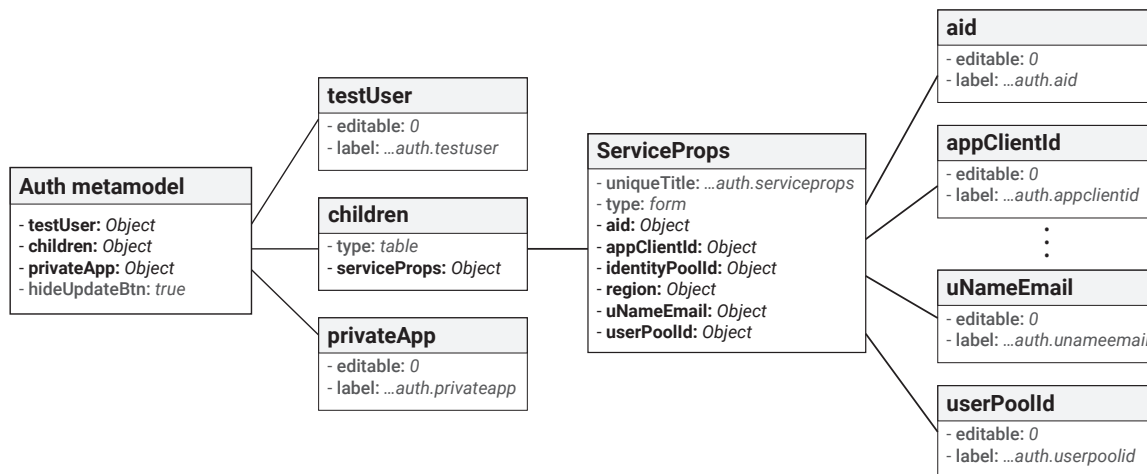
Nízkoúrovňová struktura architektury rozšiřujícího modulu je vyobrazena na obrázku 6.1. Jelikož se tato práce zabývá tvorbou rozšíření klientské části aplikace, která je implementována pomocí rámce Angular, využívá tuto technologii i nově vzniklý rozšiřující modul.

Modul je graficky reprezentován množinou komponent. Díky těmto komponentám lze zajistit interakci uživatele s modulem formou grafického uživatelského rozhraní. Vizualní stránka komponent vychází z kapitoly 5.3 a detailní principy funkcionality komponent budou detailně popsány v kapitole 6.2.



Obrázek 6.1: Nízkoúrovňová architektura rozšiřujícího modulu

Aby mohl modul správně interagovat se zbytkem platformy, došlo ke zřízení modelu, který jej popisuje. Tento model vychází z napří platformou používaného rozhraní LgmcModul eConfig a obsahuje jedinečný identifikátor modulu, jeho aktivní stav a stránky zahrnuté pod modulem. V rámci grafických komponent je dle kapitoly 5.2 třeba využívat pomocný metamodel, který pomáhá definovat strukturu vykreslovaných dat, čímž dochází ke korektnímu vykreslení UI. Konkrétní příklad vlastností metamodelu lze vidět na obrázku 6.2.



Obrázek 6.2: Schéma metamodelu autentizační konfigurace. Celá cesta je u položek label a uniqueTitle pro jednoduchost schématu zjednodušena. Znak výpustky reprezentuje název součásti modulu. Pár vlastností ServiceProps bylo pro jednoduchost schématu vypuštěno.

Jelikož se v platformě dbá na multijazyčnost, obsahuje rozšířující modul i jazykové soubory sloužící k překladu všech nezbytných názvů. Příkladem obsahu jazykového souboru je níže zmíněný kód ve formátu JSON:

```

"lgmc-gui-configs": {
  "images": {
    "defaultprofile": "Default profile",
    "favicon": "Favicon",
    "emptyplaceholder": "Empty placeholder"
  }
}
  
```

V současné chvíli podporuje platforma český a anglický jazyk a pro překlady se používá Angular translate pipe.

Pro komunikaci mezi klientskou a serverovou částí aplikace bylo použito aplikační rozhraní REST, které umožňuje jednoduchou a nenáročnou komunikaci mezi systémy. Jednou z klíčových výhod použití rozhraní REST je, že umožňuje oddělit problémy mezi klientskou a serverovou částí aplikace. To znamená, že změny v jedné části systému nemusí nutně ovlivnit ostatní části, což usnadňuje jeho údržbu a aktualizaci v průběhu času. API, pro správnou komunikaci mezi vytvářeným rozšířením a databází, bylo vytvořeno ze strany firmy Logimic a následně došlo k automatickému vygenerování služby pro její obsluhu v rámci kódu.

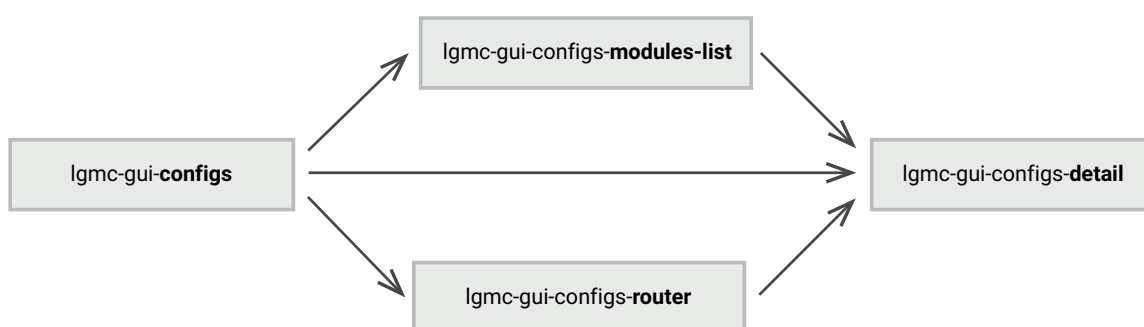
Aby docházelo k persistentnímu uchování dat, je v rámci platformy vybraná databázová technologie PostgreSQL. Jedná se o výkonnou open-source rela ní databázi, která poskytuje velkou flexibilitu a škálovatelnost. Je také známá svou spolehlivostí a výkonem. V databázi vznikly, na základ návrhu datového modelu z kapitoly 5.2, tři tabulky: `mn_config`, `rel_app_config`, `rel_app_user_config`.

## 6.2 Funkcionalita komponent

Komponenty jsou pro modul velmi důležité, protože díky nim může uživatel komunikovat s modulem a může díky tomu docházet ke změně konfigurací. Komponenty neslouží pouze pro grafickou reprezentaci dat, ale na jejich pozadí jsou provozovány i funkce umožňující korektní změnu dat a nahrání změny do databáze. Pro správnou funkcionalitu modulu vznikly následující 4 komponenty, které jsou detailně popsány v dalších sekcích této kapitoly:

- `lgmc-gui-configs` – úvodní komponenta.
- `lgmc-gui-configs-detail` – detailní komponenta, ve které probíhají změny dat.
- `lgmc-gui-configs-modules-list` – komponenta zobrazující seznam modulů.
- `lgmc-gui-configs-router` – komponenta pro nastavení routeru.

Tyto komponenty spolupracují, jak je znázorněno na obrázku 6.3. Z úvodní UI komponenty `lgmc-gui-configs` se lze prokliknout na zbylé 3. Pro dříve v této kapitole zmíněnou konfiguraci částí dojde, po prokliku, rovnou k vykreslení `lgmc-gui-configs-detail` a následně může docházet k případné změně dat. Aby se předešlo velké rozsáhlosti kódu detailní komponenty a dodržení principu dělení aplikace na více částí, došlo k vytvoření ještě dvou doplňujících komponent – pro správu modulů a routeru. `lgmc-gui-configs-modules-list` je složena z tabulky umožňující proklik do detailu jednotlivých modulů, čímž dojde k vykreslení `lgmc-gui-configs-detail`. V rámci komponenty `lgmc-gui-configs-router` je vykreslena detailní tabulka se všemi cestami a opatřena možnostmi prokliku do detailu cest a vykreslení detailní komponenty.



Obrázek 6.3: Spolupráce komponent rozšiřujícího modulu. Šipky znázorňují postupné možnosti vykreslení

## 6.2.1 Úvodní komponenta

První komponenta `lgmc-gui-configs` je úvodní komponentou rozšíření. Obsahuje tabulku všech konfigurací a možnost prokliku k jejich detailům a případně změně jejich nastavení. Tato komponenta odpovídá návrhu UI zobrazeném na obrázku 5.4.

V rámci dodržení vizuální konzistence platformy byla, nejen v úvodní komponentě, ale i ve zbylých, využita knihovna PrimeNG<sup>1</sup>. Její výhodou je, že nabízí mnoho funkcí, které jsou pro tvorbu uživatelského rozhraní potřebné (např. datové tabulky a formuláře). Díky použití PrimeNG jsem se mohl v rámci práce soustředit spíše na funkcionalitu, než na detaily návrhu uživatelského rozhraní, a zároveň navázat na výše zmíněný ložský redesign vzhledu [10]. Díky souboru CSS v rámci jednotlivých komponent pak následně jen došlo k drobným úpravám a doladění vzhledu.

Komponenta je z aplikace přístupná ze dvou cest – z administrátorské části platformy a z profilu přihlášeného uživatele. Na základě situace, ze které je komponenta vyvolána a vykreslena, se rozlišuje, zda bude zobrazený model detailu sloužit pro zobrazení dynamických uživatelských, či dynamických aplikačních nastavení.

Přepínání mezi těmito dvěma stavy řeší zabudovaný dekorátor `@Input`. V Angularu se jedná o prvek, který umožňuje komponentě přijímat vstupní data od své nadřazené komponenty skrze HTML šablonu. Hodnota dekorátoru je výchoze nastavena na úpravu dynamické aplikační konfigurace. Za předpokladu, že je požadována změna konfigurací uživatelských, lze se ke komponentě dostat přes uživatelský profil. Ten má v rámci své HTML šablony nastavenou hodnotu dekorátoru na takovou, která reprezentuje změnu dynamické uživatelské konfigurace.

Úvodní `lgmc-gui-configs` komponenta, po kliknutí na detail položky tabulky, provede přesměrování pomocí zabudovaného Angular routeru na detailní grafickou komponentu. Přes `NavigationExtras` volitelný parametr zasílá detailní komponentě data z dekorátoru, aby docházelo k rozlišení, zda se má editovat dynamická uživatelská nebo dynamická aplikační konfigurace. V platformě existují také konfigurace, které jsou čistě systémové (např. `language-extensions`) a které nedává smysl zabývat se dynamicky měnit. Tudíž ještě před samotným zobrazením tabulky dojde k jejich vyfiltrování.

## 6.2.2 Detailní komponenta

Další komponenta `lgmc-gui-configs-detail` slouží k zobrazení dat jednotlivých konfigurací a je pro rozšíření zcela klíčová – odehrává se v ní změna konfiguračních dat. Data lze měnit pouze ve chvíli, kdy jsou zobrazeny atomické hodnoty a k tomu slouží formulářový pohled (viz obrázek 5.6). Nicméně, jak již bylo párkrát zmíněno, každá konfigurace je velmi odlišná, a ne vždy se stane, že hned na nejvyšší úrovni jsou pouze atomická data. Občas je třeba se k těmto datům postupně skrývaně objekty proklikat. Z tohoto důvodu se v této komponentě nezobrazuje pouze formulář s daty, ale jsou implementovány i pohledy tabulky, dle obrázku 5.5, (pro objekty) a detailní tabulky (pro pole) viz obrázek 5.7.

Tím, že se v rámci komponenty lze proklikávat do různých hloubek zanoření, musí docházet ke správnému získávání momentálně zobrazovaných dat. Data, která v danou chvíli komponenta zobrazuje, budou dále referovaná jako model. Modely jednotlivých konfigurací jsou často velmi zavrstvené, a tudíž je žádoucí, aby se k detailním datům dalo, pro zlepšení uživatelskéhožitku, dostat i skrze změnu URL, jelikož proklikávání může být zdlouhavé.

<sup>1</sup>[www.primeng.org](http://www.primeng.org)

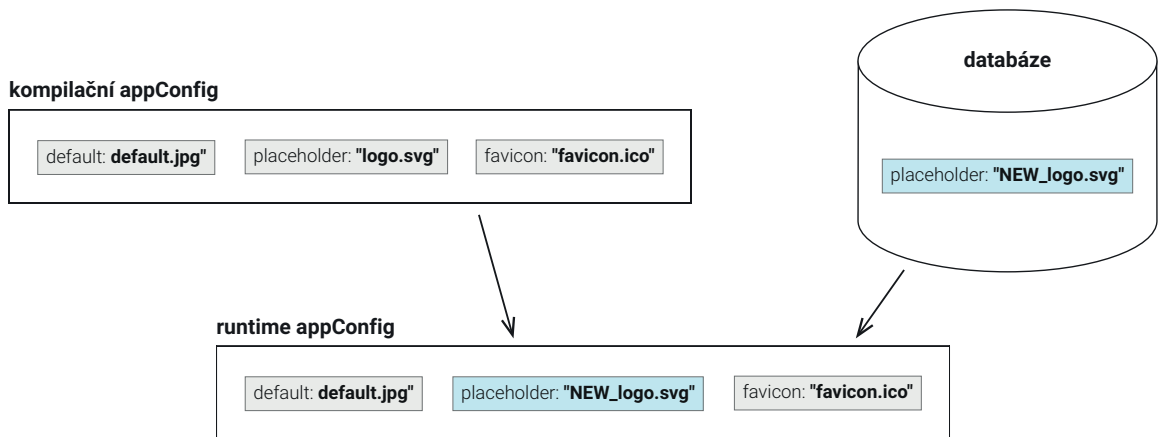
Z toho dle vodu funguje získávání aktuálního modelu na základě URL, která reprezentuje cestu k dané části modelu (viz obrázek 6.5). Níže lze vidět pseudokód získávání modelu:

```

p esm rování na novou URL
na tení konfigurace z databáze
spojení databázových dat do momentální aplika ní konfigurace
získání sou asného modelu
získání sou asného metamodelu
na tení dynamických karet

```

Na itání nového modelu se provádí pokaždé, když dojde k prokliku na detail řádku momentálně zobrazené tabulky. První ze všeho musí dojít k p esm rování na novou URL, aby se dala z aplika ní konfigurace vytáhnout data na správné cestě. Po p esm rování dojde k na tení konfigurace z databáze a ke spojení této p et žující konfigurace s momentální aplika ní konfigurací (viz obrázek 6.4). Podle momentálně upravované konfigurace dojde



Obrázek 6.4: Schéma kombinace spojení konfigurace z databáze a statické aplika ní konfigurace

k na tení dat bu z aplika ní nebo uživatelské tabulky. Tento krok probíhá z dle vodu, aby se v rámci UI zobrazoval skutečný stav výsledné dynamické konfigurace.

Následně se vezme spojená aplika ní konfigurace a na základě cesty URL dojde k iteraci na její správnou část obsahující model. Pro tuto iteraci se používá rekurzivní funkce `objectRecursi veLoop(obj: any, urls: string[])`, která na vstupu očekává objekt, ze kterého se má vrátit jeho část a jako druhý parametr očekává pole textových řetězců, díky kterému si najde cestu k danému objektu. Polem textových řetězců se rozumí URL segmenty. Návratovou hodnotou funkce je, dle aktuální URL, model a její kód je následovný:

```

let item = obj; //p i azení parametru do pomocné proměnné
for(let url of urls) { //smy ka skrze objekt
  if(item[url] == undefined) {
    return undefined; //ošet ení nedefinované vlastnosti
  }
  item = item[url]; //rekurzivní pr chod objektem
}
return item; //návrat části konfigurace na dané cest

```

Stejná funkce se používá i na získání správné části metamodelu. S cestou ke korektní části metamodelu je to bohužel složitější, jelikož nekopíruje úplně přesnou strukturu modelu. Obsah jsou v metamodelu v cestě ke správné části vloženy položky children, které se používají pro zavrstvené objekty. Aby mohla funkce korektně najít správnou část metamodelu, dochází k vložení textu „children“ mezi každé 2 URL segmenty (viz obrázek 6.5). Díky této změně se opraví cesta a může dojít ke korektní iteraci na správnou část.

```
URL
/lang/data/0/localeData/6/0

Cesta k modelu
lang | data 0 localeData 6 0

Cesta k metamodelu
lang | data 0 children localeData 6 children 0
```

Obrázek 6.5: Ukázka změny cesty pro správné nalezení části metamodelu korespondující s konkrétním modelem. Červené části jsou dynamicky vloženy do cesty pro získání dat

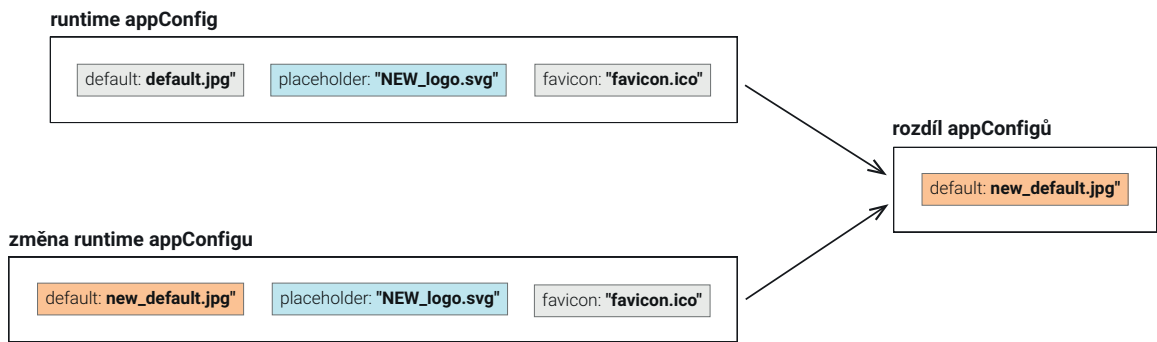
Ve chvíli, kdy je na ten korektní metamodel a je zobrazen model, jež obsahuje atomická data zobrazená v rámci formuláře, nastává čas, kdy lze data změnit. V rámci metamodelu je implementována kontrola pravomocí na úpravu jednotlivých položek. Tato kontrola se provádí porovnáním hodnoty editace, uložené v metamodelu, s hodnotou dodanou dekorátorem @Input (tedy buď úprava administrátorská nebo uživatelská). Funkce řešící uložení změných dat má následovný pseudokód:

```
identifikace změných hodnot
na tení databázové konfigurace
spojení změných hodnot s databázovými daty
nahrání nových hodnot do databáze
zplněná vazba o úspěšnosti
```

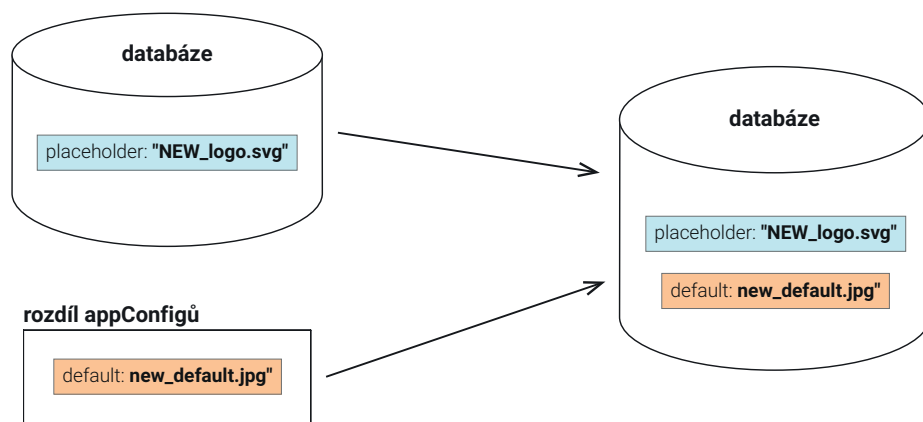
Na počátku je třeba zjistit, která data byla v rámci modelu změněna. K tomuto zjištění byla vytvořena funkce `getChangedValues(oldObj: any, newObj: any)`, která vyžaduje starou a novou konfiguraci a vrátí objekt obsahující rozdílnosti (viz obrázek 6.6). Tato funkce dokáže rozeznat rozdíly nejen atomických hodnot, ale i polí i objektů.

Poté dojde k načtení dat z databáze (buď z aplikační nebo uživatelské tabulky) a následnému spojení databázových dat se změněnými hodnotami. Po spojení je nová konfigurace nahrána do databáze (viz obrázek 6.7). Na konec dojde k zobrazení hlášky o úspěchu aktualizace.

V rámci detailní komponenty je třeba občas zobrazit i hodnoty pole. Uvnitř těchto polí je požadováno, aby bylo možné položky nejen upravovat, ale také přidávat a odebírat. Z toho důvodu jsou v detailní komponentě vytvořeny také funkce, které tyto operace umožní. V případě přidávání položky dojde k vygenerování formuláře s požadovanými hodnotami. Tyto hodnoty jsou definovány v metamodelu pod položkou `requiredParams`. Pro určení, které hodnoty objekt pole mají být v detailní tabulce zobrazeny, se využívá z metamodelu parametr `columns`. Dynamicky přidávané položky pole jsou označeny příznakem `dynamicAdd` a díky tomu jsou od nezbytných, staticky definovaných, aplikačních položek odlišeny. Tím pádem se nemůže stát, že by byl odstraněn řádek, který nebyl přidán dynamicky. Ikona na



Obrázek 6.6: Vstupy funkce `getChangedValues` a její výstup v podobě nového objektu obsahující pouze změněná data



Obrázek 6.7: Poslední fáze aktualizace dat. Dochází ke spojení dat z databáze a nově změněných dat v konfiguraci a výsledek této operace je nahrán jako nový stav databáze

odstranění položky se zobrazuje pouze u dynamicky přidávaných položek, a tudíž odstranění programů definovaných řádkem není díky tomuto mechanismu umožněno. Před samotným smazáním dynamického řádku je ještě vykreslena potvrzovací tabulka pro minimalizaci chyb z prokliknutí.

### 6.2.3 Ostatní komponenty

Aby se předešlo velké rozsáhlosti detailní komponenty a dodržení principu dělení aplikace na více částí, došlo k vytvoření ještě dvou doplujících komponent. První z nich je `l-gmc-gui-configs-modules-list`. Jak již název napovídá, jedná se o komponentu zobrazující seznam modulů. Moduly jsou jednou z nejrozsáhlejších a nejkomplexnějších částí aplikační konfigurace a v rámci jejich vykreslení v tabulku jsou žádoucí funkce, které by v klasickém detailním zobrazení byly navíc. Grafické rozhraní komponenty se skládá z jednoduché tabulky sloužící pro přehledné zobrazování na detail jednotlivého modulu (podobně jako na obrázku 5.4). Řádky jsou nicméně v rámci zlepšení uživatelské přívětivosti obohaceny o příznak aktivity daného modulu. Je-li modul aktivní, svítí vedle tlačítka na prokliknutí detailu modulových nastavení zelený indikátor aktivity. Není-li ovšem aktivní, je tento indikátor

nahrazen tlačítkem pro aktivaci – tím se uživateli umožní zapnout modul bez nutnosti návštěvy jeho detailu. Samotné moduly mají své ID ve tvaru `lgmc-gui-nazev`, tudíž aby bylo zobrazení názvu modulu v jednotlivých řádcích přehlednější, existuje v komponentě funkce separující z celého ID pouze část s názvem, která je následně v tabulce zobrazena. Význam této komponenty je spíše jako prostředník mezi úvodní a detailní komponentou.

Podobnou funkci má i poslední komponenta `lgmc-gui-configs-router`, jež slouží k nastavení konfigurací routeru. Dominantní částí routeru je část se samotnými uloženými cestami a jelikož se jedná o pole, je tato část řešena zobrazením ve formě detailní tabulky (viz obrázek 5.7). Po diskuzi se zástupci firmy Logimic se došlo k závěru, že konfiguraci routeru není žádoucí mít, a tudíž byly všechny změny, skrze metamodel, zakázány. Nicméně, na rozdíl od například jazykové konfigurace, je žádoucí konfiguraci routeru alespoň zobrazovat, a tak byla komponenta ponechána.

#### 6.2.4 Princip vykreslení UI

Celé rozšíření používá již hotové komponenty z knihovny PrimeNG a univerzální grafické komponenty vytvořené firmou Logimic (část `lgmc-g` z obrázku 4.1). Jádrem UI je tabulka `p-table` a formulář `spidlenoutinputgroup`. Jak již bylo výše zmíněno, tabulka slouží pro vypsání objektů a polí v rámci aktuálního modelu a možnosti prokliku k jejich úpravám. Formulář je pro zobrazení a úpravu atomických dat, jako jsou například číselné, textové, nebo pravdivostní hodnoty.

Komponenty jsou stavěny tak, aby využívaly metamodel ke správnému vykreslení hodnot. Na jeho základě se volí typ vykreslení, zda se použije ikona nebo popisek, unikátní názvy jednotlivých položek či zobrazení aktualizace tlačítka. Ovšem jedním z požadavků při návrhu bylo, aby šlo data zobrazovat i za stavu, kdy bude metamodel chybět. Z toho důvodu dochází v rámci HTML šablony ke kontrole jeho existence a pokud dojde k zjištění, že neexistuje, dojde i tak k vykreslení dat. Princip vykreslení v takovém případě funguje na principu zanedbání typu vykreslení a jelikož dopředu nelze vědět, jaká data model obsahuje, vykreslí se jak tabulka, tak formulář. Aby nicméně nedošlo k zobrazení objektu ve formuláři nebo atomických dat v tabulce a byla zachována funkčnost, před samotným vygenerováním položky dojde k ověření datového typu konkrétní vlastnosti modelu. Dalo by se tedy říct, že za absence metamodelu je zachována funkčnost a jejím důsledkem je pouze estetická stránka.

Velkou roli v rámci vykreslování uživatelského rozhraní hraje také grafická komponenta `lgmc-g-tabs-to-accordion`, která vytváří, v rámci UI, záložky (viz horní část obrázku 5.5). Tyto záložky jsou dynamicky generovány na základě pole `dynamicTabsArr`, které obsahuje objektové struktury na aktuální úrovni modelu. Díky těmto záložkám odpadá nutnost zahrnovat všechny položky do tabulky a dochází ke zmenšení již velkého počtu kliknutí před samotnou editací dat. Tato grafická komponenta také velmi pomáhá celkovému responzivnímu zobrazení detailní komponenty modulu například různými velikostmi obrazovek. Díky zalamovacímu bodu se záložky na menších obrazovkách změní v *accordion* zobrazení a jsou místo vedle sebe zobrazeny pod sebou.

V rámci zobrazení dopředu neznámé struktury dat došlo vlastněmu využití `keyvaluepipe`, která zajišťuje přístup jak k hodnotě, tak ke klíči dané vlastnosti modelu. Díky této funkcionalitě lze snadno dynamicky vykreslit data modelu na daném klíči. Při použití `keyvalue` bohužel docházelo ke zmatkové uspořádání řádků a jejich seřazení dle klíče, což není ve všech případech žádoucí, a tudíž byla implementována funkce `originalOrder`, která zajišťuje zachování původního seřazení. Při editaci dat z formuláře docházelo bohužel ke

ztratění na momentální vstup z důvodu překreslování DOM. Aby se zamezilo tomuto problematickému chování, používá se u formulářových vstupů atribut `trackBy` a funkce `trackByFn`, která zamezuje ztratění na momentálně editovaný vstup.

### 6.3 Napojení na existující systém

Vytvářené rozšíření zapadá, dle architektury SmartCity zmíněné na obrázku 4.1, do složky `lgmc-gui` a konkrétně se jedná o modul s názvem `lgmc-gui-configs`. Model tohoto modulu vychází z napří platformou používané struktury `LgmcModuleConfigs`. Nicméně naprogramování samotného modulu by nestačilo. Aby došlo k propojení vytvořeného modulu a zbytku platformy, musely být v rámci implementace pozmeněny určité konfigurační soubory. Základní změnou byl import nově vzniklého modulu do aplikačního modulu – tedy modulu nejvyšší úrovně, nacházející se v složce `app`.

Poté, co byl modul propojen na nejvyšší úrovni, byl vytvořen konfigurační soubor v rámci složky `configs`. Tento konfigurační soubor, dle šablony, obsahuje id modulu, jeho aktivní stav a všechny komponenty zahrnuté pod stránkami. V rámci každé stránky je definována přístupová URL, komponenta k vykreslení, štítek, ikona a přístupový formulář. V platformě je implementována bezpečnostní kontrola, zda má uživatel právo zobrazit si danou komponentu, zvaná `accessForm` (neboli přístupový formulář). Jedná se o textový metazetec, který bylo potřeba přidat do databáze, aby nebylo zobrazení komponent blokováno.

Po vytvoření této konfigurace došlo k zahrnutí všech vytvořených komponent do konfigurace routeru. Díky této změně dochází k vykreslení požadovaných komponent na základě dané URL. Po provedení výše zmíněných změn a zadání správné URL, šlo komponenty zobrazit, nicméně nebylo ještě dořešeno, jak se dostat ke komponentám skrze odkazy v rámci aplikace. Z tohoto důvodu byla rozšířující úvodní komponenta (`lgmc-gui-configs`) zakomponována do extensí administrátorského modulu a díky tomu se objevila dlaždice s možností kliknutí z administrátorské části platformy. Aby se dalo ke změnám konfigurací dostat i přes uživatelský profil a tímto mít uživatelská nastavení, došlo k přidání `lgmc-gui-configs` komponenty do HTML šablony profilové komponenty.

V danou chvíli byla třeba doladit ještě jazykové předklady. Pro jejich správné fungování byla cesta k jazykovým mutacím modulu přidána do konfiguračního souboru `angular.json` a zanesena do konfigurace jazyku nacházející se v části `configs`.

Probléma také změna služby `app-state-service` umístěné v `lgmc-platform`. Při kompilaci aplikace je nezbytné, aby byly spojeny staticky nastavené konfigurace s konfiguracemi dynamickými. Z tohoto důvodu byly do inicializační funkce aplikace přidány funkce získávající data z databáze. Po získání dat dojde nejprve ke sloučení statické konfigurace s dynamickou aplikační konfigurací. Poté je nově vzniklá konfigurace ještě spojena s dynamickou uživatelskou konfigurací, dle aktuálně přihlášeného uživatele.

# Kapitola 7

## Testování

Testování je nezbytným aspektem vývoje software, protože pomáhá identifikovat potenciální chyby a nedostatky v softwaru ještě před jeho uvolněním pro koncové uživatele. V rámci této práce probíhala největší část testování již během vývoje, aby došlo k odchytení chyb v co možná nejranější fázi. V této kapitole bude postupně popsáno testovací prostředí, princip testování a jeho výsledky.

### 7.1 Testovací prostředí

Testovací prostředí je s vývojovým velmi podobné. Klientská část platformy běží na rámci Angular ve verzi 13 v kombinaci s Node.js verze 16. Příklad aplikace byl řešen pomocí WSL2 (*Windows Subsystem for Linux*<sup>1</sup>), díky kterému mohou vývojáři používající operační systém Windows pracovat v GNU/Linux prostředí a tím práce mohla simulovat linuxový server, na kterém je ostrý provoz platformy provozován. V rámci testování byla tedy platforma provozována lokálně. Jelikož se jedná o webovou aplikaci, byla funkčnost rozšíření zkušena na 3 typech rozšířených prohlížečů: Google Chrome, Microsoft Edge a Mozilla Firefox. Původní plán byl testovat i lokální databázi, ale kvůli složitosti realizování lokální API a irelevantnosti tohoto kroku vzhledem k tématu této práce, se podařilo domluvit vytvoření API komunikující se vzdálenou testovací databází – jinými slovy komunikace s těmi produkty verzí serverové části. Po ukončení vývoje došlo nakonec na poslední lokální testování. Dalším krokem bylo testování rozšíření ze strany firmy Logimic a jejich vývojáři.

### 7.2 Testovací scénáře

Pro demonstraci testování bylo vybráno pouze pár scénářů ukazujících klíčové kroky a potenciální problémy. Dané scénáře vždy zastupují určitou problematiku změn dat definovaných struktur. V každém scénáři je popsáno, jaká konkrétní data se budou měnit, očekávané projevení změn a jaký je účel daného testu. Pokud se provedená změna projeví a zároveň bude změněná hodnota obsažena v databázi, pak byl test úspěšný. Pokud selže test alespoň částečně, bude považován za neúspěšný a scénář označen za implementační nedostatek. Před každým testem byla vyžádána konfigurace z databáze a po ověření správnosti uložených dat došlo k oprotivnému uvedení do původního stavu.

<sup>1</sup>[www.learn.microsoft.com/en-us/windows/wsl/about](https://www.learn.microsoft.com/en-us/windows/wsl/about)

### 7.2.1 Test API

- ú el – zjištění, zda data zaslaná API jsou shodná s daty uloženými v databázi. Test všech 4 komunikačních metod (GET, POST, PUT, DELETE). Vyzkoušet jak pro aplikaci, tak pro uživatelskou konfiguraci.
- m n á data – vytvořen pokusný objekt obsahující atomické hodnoty, objekty a pole.
- o ekávaný výsledek – korektní vytvoření tabulek, aktualizace a získání jejich dat a následné smazání.

### 7.2.2 Jednoduchá změna

- ú el – pokus o základní a triviální změnu jedné atomické hodnoty. Mělo by se jednat o změnu s nejmenší složitostí, a tudíž optimální první test na jehož úspěšném zvládnutí budou budovány složitější.
- m n á data – změna boolean hodnoty hover m níci chování postranní navigace. Vyskytuje se pod konfigurací layouts, která slouží k nastavení rozložení prvků. Celá cesta k hodnotě je: layouts / sidebar / hover.
- o ekávaný výsledek – cílem testu je vypnout hover efekt na bočním panelu navigace. Po najetí myši na postranní navigaci by nemělo dojít k expanzi panelu a zůstane pouze ve formě ikon (nikoliv ikon a názvů, jako je tomu při aktivním stavu hover).

### 7.2.3 Změna pole

- ú el – vyzkoušení detailní tabulky a možnosti mít počet položek pole.
- m n á data – Přidání položky do postranního menu. Otevření detailu položky a následné odstranění řádku. Cesta k datům: menu.
- o ekávaný výsledek – Přidání řádku v tabulce a jeho následné korektní odstranění.

### 7.2.4 Změna zanořeného objektu

- ú el – vyzkoušení změny velmi zanořené vlastnosti objektu a test správného propojení modelu s metamodelem.
- m n á data – jako pokusná vlastnost byl vybrán štítek v rámci admin modulu na cestě: modules / admin / pages / role / label.
- o ekávaný výsledek – změna popisku stránky s rolemi napojené na admin modul.

### 7.2.5 Změna objektu uvnitř pole v rámci uživatelské konfigurace

- ú el – změna vlastnosti objektu, jehož nadřazená struktura je pole. Vyzkoušení algoritmu na nacházení změny a korektní nahrání do tabulky s uživatelskou konfigurací.
- m n á data – cesta k datům je definovaná jako: menu / dashboard / icon.
- o ekávaný výsledek – okamžitá změna ikony postranní navigace a korektní uložení do databázové tabulky s uživatelskými konfiguracemi.

### 7.2.6 Propsání do jádra

- ú el – n která nastavení se nemusí projevit ihned po jejich zm n , ale je pot eba vy ešit jejich zm nu v jádru aplikace i již p i p ekladu.
- m n ná data – l a b e l s ! t a b t i t l e.
- o ekávaný výsledek – zm na názvu záložky zobrazované naho e v prohlíže i. Tento atribut se nastavuje v jád e již p i inicializaci, takže o ekávaný výsledek testu je selhání.

## 7.3 Výsledky testování

Testy aplika ního rozhraní objevily problémy s metodou DELETE pro mazání ádk apli-ka ních konfigurací (uživatelské fungovali). Další problém byl v mazání záznam pouze z rela ní tabulky rel\_app\_conf i g a nikoliv z mn\_conf i g, na kterou je odkazováno. Na problémy jsem upozornil vývojá e firmy Logimic a problém byl brzy opraven, tudíž nyní funguje API korektn .

Výsledek testu 7.2.2 se ukázal jako správný – po vypnutí hover funkcionality opravdu došlo ke zm n chování. Pozitivní výsledky vyšly také z dalších test zabývajících se zm -nou dat. áste n problematický byl test 7.2.3. Zm ny tabulky se projevují v databázi i v ádcích tabulky korektn , nicmén nedošlo k p ídání položky do postranního menu. Tuto problematiku eší jádro aplikace.

Dalším problémem byl test 7.2.6. Tento test potvrdil o ekávání, že se ur itý typ zm n nepropíše ihned do chodu platformy, nýbrž je t eba m nit data v rámci jádra. O této sku-te nosti byla firma informována a bylo domluveno, že dojde k implementaci toto zajiš ující z jejich strany. Mým posledním poznatkem z testování je, že p i na ítání dat dochází k jis-tým prodlevám mezi na ítáním nového modelu a zm nou UI, která na ítání p edbíhá. Tento problém je dán pomalejším na ítáním dat z databáze a problém se ješt bude do budoucna ešit. Další forma testování probíhá v sou asnou chvíli intern vývojá i firmy Logimic a jejich zákazníky. Pokud se ukáže, že rozší ení bezchybn pracuje se zbytkem platformy a nedojde k objevení problematických ástí, bude rozší ení nasazeno do provozu.

## Kapitola 8

# Závěr

Cílem této bakalářské práce bylo zmapovat možnosti dynamických změn nastavení informačních systémů pro správu chytrých měst a navrhnout možné řešení pro platformu SmartCity od firmy Logimic. Výsledné řešení má zefektivnit práci vývojářů a povolit uživatelům a administrátorům přizpůsobit si platformu, bez nutnosti zásahu do programu.

Pro pochopení problematiky a vytvoření co možná nejlépe sloužícího řešení bylo před samotnou tvorbou potřeba nastudovat principy internetu v cíl a fungování systémů pro správu obsahu (CMS). Aby navrhované řešení zapadlo do koncepce platformy SmartCity, bylo nezbytné projít její dokumentaci a do detailu pochopit její architekturu.

Navržené řešení je ve formě rozšíření existujícího modulu, který umožňuje nastavení klientské části platformy dynamicky, za jejího běhu. Změny nastavení již nadále nemusí probíhat programově, nýbrž skrze grafické uživatelské rozhraní přímo z aplikace. Dynamicky měnit data je třeba persistentně uchovávat, tudíž došlo k rozšíření současně používané databáze a vytvoření API pro správnou komunikaci se serverovou částí aplikace.

Jelikož byl během tvorby práce rozšířen původní cíl z nástroje pro správu modulů na nástroj pro správu konfigurace celé platformy, vznikl problém s tvorbou uživatelského rozhraní. Každá konfigurační část má velmi rozdílnou strukturu dat, a tudíž byla vytvořena pomocná struktura pro popis dat – metamodel. Tato pomocná struktura obsahuje metadata o konfiguračních modelech a zajišťuje korektní práci s daty a jejich vizualizací. Metamodel také implementuje kontrolu pravomocí a dovoluje měnit pouze vybrané atributy, aby nedošlo k poškození běhu celé platformy.

Vytvořený nástroj umožňuje jednotlivá nastavení aplikace pomocí formulářového zobrazení přímo z aplikace, a to bez znalostí vnitřní programové logiky. Změny probíhají za běhu aplikace, takže dochází k požadované částečné delegaci práce vývojářů a na uživatele a administrátora a dochází tím také ke snížení počtu nezbytných zásahů do kódu. Díky tomuto rozšíření mohou nyní zasahovat uživatelé a administrátoři do nastavení platformy, což před touto prací nebylo možné.

Výsledek práce bude využit pro další nastavbu platformy ve formě plně přizpůsobitelných dashboardů, aby mohlo docházet k lepší personalizaci řešení pro jednotlivé zákazníky. Rozšíření existujícího modulu neumí v současné chvíli propisovat změny do jádra aplikace, a tak bude potřeba zásah ze strany vývojářů platformy. Na která data je totiž nutné definovat již před překlady a kompilací programu. Další potenciální rozšíření může být například obohacení o uživatelské styly, jež firma teprve plánuje doimplementovat, a díky této práci má předpřipravený nástroj.

# Literatura

- [1] *The "Only" Coke Machine on the Internet* [online]. 1998 [cit. 2022-10-02]. Dostupné z: [https://www.cs.cmu.edu/~coke/history\\_long.txt](https://www.cs.cmu.edu/~coke/history_long.txt).
- [2] Abdul Ghani, H. A., Konstantas, D. a Mahyoub, M. A comprehensive IoT attacks survey based on a building-blocked reference model. *International Journal of Advanced Computer Science and Applications*. Science and Information (SAI) Organization Limited. 2018, sv. 9, . 3.
- [3] Al Qaseemi, S. A., Al mul him, H. A., Al mul him, M. F. a Chaudhry, S. R. IoT architecture challenges and issues: Lack of standardization. In: IEEE. *2016 Future technologies conference (FTC)*. 2016, s. 731–738. ISBN 978-1-5090-4171-8.
- [4] Al Sarawi, S., Anbar, M., Al ieyan, K. a Al zubaidi, M. Internet of Things (IoT) communication protocols. In: IEEE. *2017 8th International conference on information technology (ICIT)*. 2017, s. 685–690. ISBN 978-1-5090-6332-1.
- [5] Aleksandr. *Plugins* [online]. [cit. 2019-11-20]. Dostupné z: [https://www.dokuwiki.org/plugin\\_s](https://www.dokuwiki.org/plugin_s).
- [6] Amazon Staff. *10 years of Amazon robotics: how robots help sort packages, move product, and improve safety* [online]. Amazon, erven 2022 [cit. 2022-12-05]. Dostupné z: <https://www.aboutamazon.com/news/operations/10-years-of-amazon-robotics-how-robots-help-sort-packages-move-product-and-improve-safety>.
- [7] Amsler, S. *Content management system (CMS)* [online]. Editoval Fred Churchville. TechTarget, únor 2021 [cit. 2022-10-23]. Dostupné z: <https://www.techtarget.com/searchcontentmanagement/definition/content-management-system-CMS>.
- [8] Babun, L., Denney, K., Cel ik, Z. B., McDaniel , P. a UI uagac, A. S. A survey on IoT platforms: Communication, security, and privacy perspectives. *Computer Networks*. Elsevier. 2021, sv. 192, s. 108040.
- [9] Barker, D. *Web content management: Systems, features, and best practices*. 1. vyd. "O'Reilly Media, Inc.", 2016. ISBN 1491908084.
- [10] Col níková, I. *Mobilní aplikace pro správu za ízení v IoT*. Brno, CZ, 2022. Bakalá ská práce. Vysoké u ení technické v Brn , Fakulta informa ních technologií. Dostupné z: [https://www.fit.vut.cz/study/thesi\\_s/24478/](https://www.fit.vut.cz/study/thesi_s/24478/).
- [11] Coskun, V., Ozdenizci, B. a Ok, K. A survey on near field communication (NFC) technology. *Wireless personal communications*. Springer. 2013, sv. 71, . 3, s. 2259–2294.

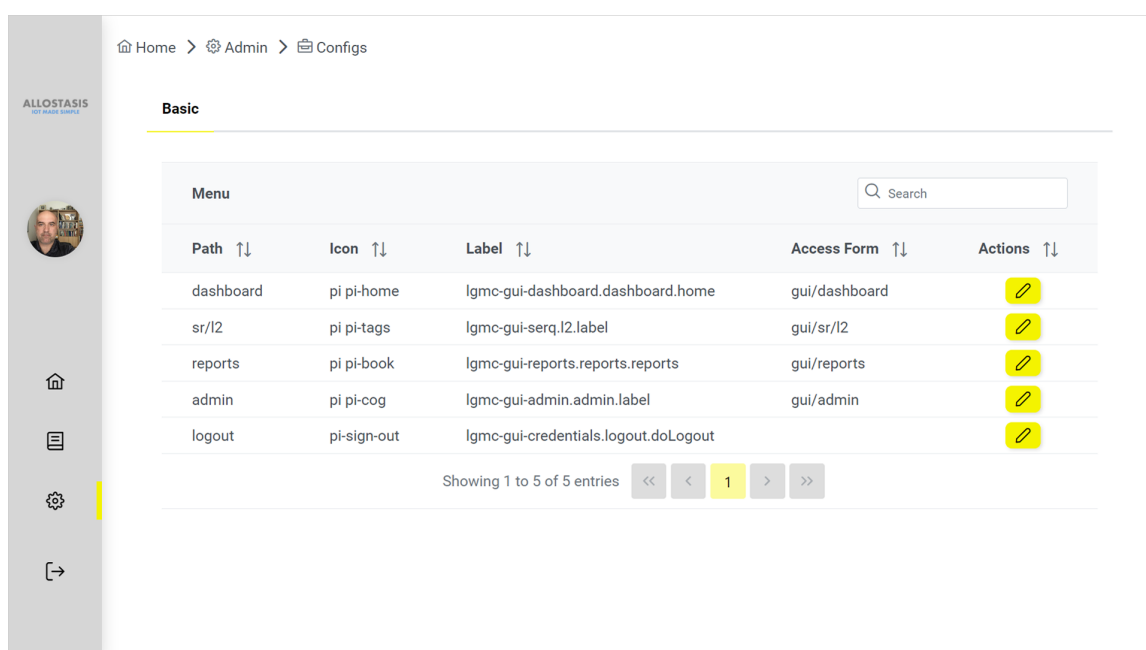
- [12] Crane, C. *What Is IoT Security? Insights & Tips from 11 IoT Experts* [online]. The SSL Store, říjen 2021 [cit. 2022-10-30]. Dostupné z: <https://www.thesslstore.com/blog/what-is-iot-security-insights-tips-from-iot-experts/>.
- [13] Datta, P. a Sharma, B. A survey on IoT architectures, protocols, security and smart city based applications. In: IEEE. *2017 8th International conference on computing, communication and networking technologies (ICCCNT)*. 2017, s. 1–5. ISBN 978-1-5090-3038-5.
- [14] Devalal, S. a Karthikeyan, A. LoRa technology-an overview. In: IEEE. *2018 second international conference on electronics, communication and aerospace technology (ICECA)*. 2018, s. 284–290. ISBN 978-1-5386-0965-1.
- [15] Dey, N., Hassanien, A. E., Bhatt, C., Ashour, A. a Satapathy, S. C. *Internet of things and big data analytics toward next-generation intelligence*. Springer, Switzerland, 2018. ISBN 9783319604350.
- [16] Dixit, A. *IoT Architecture – Detailed Explanation* [online]. červen 2022 [cit. 2022-12-27]. Dostupné z: <https://www.interviewbit.com/blog/iot-architecture/>.
- [17] Drupal.cz. *O Drupalu* [online]. Únor 2018 [cit. 2022-11-20]. Dostupné z: <https://www.drupal.cz/o-drupalu>.
- [18] Evans, D. *The Internet of Things: How the Next Evolution of the Internet Is Changing Everything* [online]. Duben 2011 [cit. 2022-10-02]. Dostupné z: [https://www.cisco.com/c/dam/en\\_us/about/ac79/docs/innov/IoT\\_IBSG\\_0411FINAL.pdf](https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf).
- [19] Fitzgerald, A. *14 Features Your CMS Absolutely Needs* [online]. Duben 2022 [cit. 2022-11-18]. Dostupné z: <https://blog.hubspot.com/web-site/cms-features>.
- [20] Gascó Hernandez, M. Building a smart city: Lessons from Barcelona. *Communications of the ACM*. ACM New York, NY, USA. 2018, sv. 61, . 4, s. 50–57.
- [21] Gohr, A. *DokuWiki* [online]. [cit. 2019-11-20]. Dostupné z: <https://www.dokuwiki.org/dokuwiki>.
- [22] Greengard, S. *The internet of things*. MIT press, 2021. ISBN 9780262542623.
- [23] Hejazi, H., Rajab, H., Cinkler, T. a Lengyel, L. Survey of platforms for massive IoT. In: IEEE. *2018 IEEE international conference on future IoT technologies (future IoT)*. 2018, s. 1–8. ISBN 978-1-5386-1208-8.
- [24] Heydon, R. a Hunn, N. Bluetooth low energy. *CSR Presentation, Bluetooth SIG*. 1. vyd. říjen 2012.
- [25] Jackson, B. *Joomla vs WordPress – Which One is Better? (Pros and Cons)* [online]. Kinsta, duben 2022 [cit. 2022-11-19]. Dostupné z: <https://kinsta.com/blog/joomla-vs-wordpress/>.
- [26] Jackson, B. *WordPress vs Drupal – Which One is Better in 2022? (Pros and Cons)* [online]. Kinsta, listopad 2022 [cit. 2022-11-20]. Dostupné z: <https://kinsta.com/blog/wordpress-vs-drupal/>.

- [27] Joomla!. *About Joomla!* [online]. [cit. 2022-11-19]. Dostupné z: <https://www.joomla.org/about-joomla.html>.
- [28] Keegan, S., O'Hare, G. M. a O'Grady, M. J. Retail in the digital city. *International Journal of E-Business Research (IJEER)*. 1. vyd. IGI Global. 2012, sv. 8, . 3, s. 18–32.
- [29] Kirimtat, A., Krejcar, O., Kertesz, A. a Tasgetiren, M. F. Future trends and current state of smart city concepts: A survey. *IEEE access*. IEEE. 2020, sv. 8, s. 86448–86467.
- [30] Lavric, A., Petrariu, A. I. a Popa, V. SigFox communication protocol: The new era of IoT? In: IEEE. *2019 international conference on sensing and instrumentation in IoT Era (ISSI)*. 2019, s. 1–4. ISBN 978-1-7281-1022-6.
- [31] Logimic. *Logimic IoT Platforma* [online]. 2022 [cit. 2022-12-09]. Dostupné z: <https://www.logimic.com/cs/platforma/>.
- [32] Mahmoud, R., Yousuf, T., Aloul, F. a Zualkernan, I. Internet of things (IoT) security: Current status, challenges and prospective measures. In: IEEE. *2015 10th international conference for internet technology and secured transactions (ICITST)*. 2015, s. 336–341. ISBN 978-1-9083-2052-0.
- [33] McKeever, S. Understanding Web content management systems: evolution, lifecycle and market. *Industrial management & data systems*. MCB UP Ltd. 2003, sv. 103, . 9, s. 686–692.
- [34] Meike, M., Sametinger, J. a Wiesauer, A. Security in open source web content management systems. *IEEE Security & Privacy*. IEEE. 2009, sv. 7, . 4, s. 44–51.
- [35] Mizrachi, A. *What Are User Permissions? Concepts, Examples, and Maintenance* [online]. červen 2022 [cit. 2023-01-23]. Dostupné z: <https://frontegg.com/blog/user-permissions/>.
- [36] Mone, G. The new smart cities. *Communications of the ACM*. ACM New York, NY, USA. 2015, sv. 58, . 7, s. 20–21.
- [37] Munro, K. *Internet of Things Security* [online]. TEDx Talks, září 2018 [cit. 2022-10-30]. Dostupné z: <https://www.youtube.com/watch?v=pGtnC1jKpMg>.
- [38] Palomo Navarro, A. a Navío Marco, J. Smart city networks' governance: The Spanish smart city network case study. *Telecommunications Policy*. Elsevier. 2018, sv. 42, . 10, s. 872–880.
- [39] Pierleoni, P., Concetti, R., Belli, A. a Palma, L. Amazon, Google and Microsoft solutions for IoT: Architectures and a performance comparison. *IEEE access*. IEEE. 2019, sv. 8, s. 5455–5470.
- [40] shekharsaxena316. *5 Layer Architecture of Internet of Things* [online]. Září 2020 [cit. 2022-11-10]. Dostupné z: <https://www.geeksforgeeks.org/5-layer-architecture-of-internet-of-things/>.






- [41] Sitecore. *Finding out which CMS features you need* [online]. [cit. 2022-11-18]. Dostupné z: <https://www.sitecore.com/knowledge-center/digital-marketing-resources/the-cms-features-you-need>.
- [42] Sterne, J. *Plug-in* [online]. Encyclopedia Britannica, 3. října 2019 [cit. 2022-10-23]. Dostupné z: <https://www.britannica.com/technology/plugin>.
- [43] W3techs. *Usage statistics of content management systems* [online]. říjen 2022 [cit. 2022-10-23]. Dostupné z: [https://w3techs.com/technologies/overview/content\\_management](https://w3techs.com/technologies/overview/content_management).
- [44] Wordpress.org. *Our Story* [online]. [cit. 2022-11-19]. Dostupné z: <https://wordpress.org/about/>.
- [45] World Health Organization. *Ambient (outdoor) air pollution* [online]. Září 2021 [cit. 2022-11-29]. Dostupné z: [https://www.who.int/news-room/fact-sheets/detail/ambient-\(outdoor\)-air-quality-and-health](https://www.who.int/news-room/fact-sheets/detail/ambient-(outdoor)-air-quality-and-health).
- [46] Xie, X.-F. *Smart and Scalable Urban Signal Networks* [online]. Srpen 2016 [cit. 2022-10-02]. Dostupné z: <http://www.wimax.com/team/xie/schic/>.

# Příloha A

## Snímky obrazovky

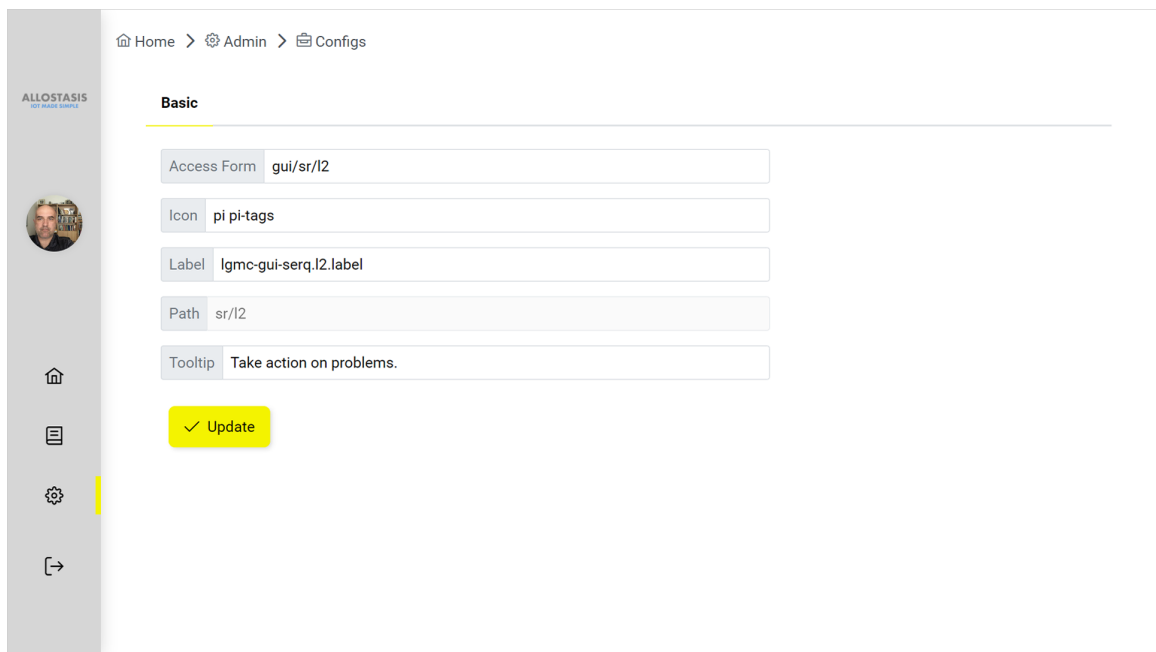


The screenshot shows the 'Basic' configuration page for the 'Menu' in the ALLOSTASIS system. The breadcrumb navigation is 'Home > Admin > Configs'. The page title is 'Basic'. Below the title is a search bar labeled 'Search'. The main content is a table with the following columns: Path, Icon, Label, Access Form, and Actions. The table contains five entries, each with a yellow edit icon in the Actions column. At the bottom of the table, there is a pagination control showing 'Showing 1 to 5 of 5 entries' and navigation buttons for first, previous, current (1), next, and last.

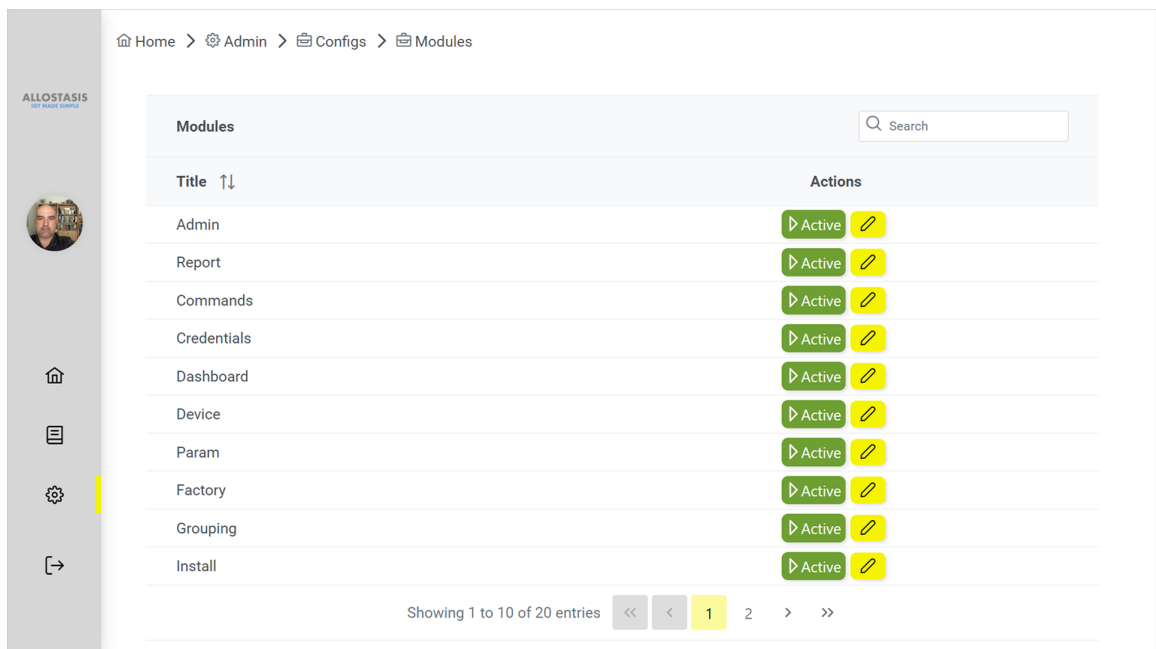
Path ↑↓	Icon ↑↓	Label ↑↓	Access Form ↑↓	Actions ↑↓
dashboard	pi pi-home	lgmc-gui-dashboard.dashboard.home	gui/dashboard	
sr/l2	pi pi-tags	lgmc-gui-serq.l2.label	gui/sr/l2	
reports	pi pi-book	lgmc-gui-reports.reports.reports	gui/reports	
admin	pi pi-cog	lgmc-gui-admin.admin.label	gui/admin	
logout	pi sign-out	lgmc-gui-credentials.logout.doLogout		

Showing 1 to 5 of 5 entries << < 1 > >>

Obrázek A.1: Detailní tabulka zobrazující nastavení menu



Obrázek A.2: Formulářový pohled položky menu



Obrázek A.3: Tabulka zobrazující seznam modulů

ALLOSTASIS  
IOT MADE SIMPLE

> Basic

Service Props

Aid SmartCity

App client ID 3rkhmaqha6i80i7rvjld29884l

Identity pool ID us-east-1:c5fe2897-9feb-4217-8dcf-5f447c824093

Region us-east-1

User pool ID us-east-1\_PRR0R4tSwb

Uname e-mail

ALLOSTASIS  
IOT MADE SIMPLE

Basic

Menu

Search

path	dashboard
icon	pi pi-home
label	Igmc-gui-dashboard.dashboard.home
accessForm	gui/dashboard
path	sr/l2
icon	pi pi-tags
label	Igmc-gui-serq.l2.label
accessForm	gui/sr/l2
path	reports
icon	pi pi-book
label	Igmc-gui-reports.reports.reports
accessForm	gui/reports

<< < 1 > >>

Obrázek A.4: Responzivní zobrazení formuláře a detailní tabulky. Záložky jsou zobrazeny ve form *accordion* rozložení