**Česká zemědělská univerzita v Praze**

**Provozně ekonomická fakulta**

**Katedra informačního inženýrství**


**Diplomová práce**

# Využití microservices architektury pro automatizaci firemních procesů

**Bc. Miloslav Lejček**

# ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

# ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Miloslav Lejček

Systémové inženýrství a informatika
Informatika

Název práce

**Využití microservices architektury pro automatizaci firemních procesů**

Název anglicky

**Use of microservices architecture for automation of business processes**

---

**Cíle práce**

Diplomová práce je tematicky zaměřena na technologie určené k automatizaci firemních procesů. Hlavním cílem práce je návrh systému pro automatizaci firemních procesů s využitím microservices návrhového vzoru.
Dílčí cíle diplomové práce jsou:

- Vytvoření literární rešerše týkající se problematiky firemních procesů a microservices návrhového vzoru.
- Analýza vybraných technologií dostupných pro automatizaci firemních procesů.
- Návrh systému určeného k automatizaci firemních procesů s využitím microservices návrhového vzoru.
- Ověření návrhu na vybraném firemním procesu.
- Syntetizace výsledků práce, formulace přínosů a závěry práce.

**Metodika**

Metodika řešené problematiky diplomové práce vychází ze studia a analýzy odborných informačních zdrojů, stejně tak jako ze zkušeností převzatých z praxe. Praktická část je zaměřena na vypracování případové studie, pro kterou bude vyhotoven návrh řešení automatizace procesů. Na základě syntézy teoretických poznatků a výsledků praktické části práce budou formulovány závěry diplomové práce.

**Doporučený rozsah práce**

60-80

**Klíčová slova**

BPMN, microservices architektura, firemní procesy, Java

---

**Doporučené zdroje informací**

LACKO, Ľ. SQL : hotová řešení : pro SQL Server, Oracle a MySQL. Brno: Computer Press, 2003. ISBN 80-7226-975-5

PECINOVSKÝ, R. Myslíme objektově v jazyku Java : kompletní učebnice pro začátečníky. Praha: Grada, 2009

SHARMA, S. Mastering Microservices with Java: Build Enterprise Microservices with Spring Boot 2.0, Spring Cloud, and Angular. Packt Publishing Ltd, 2019.

SVOZILOVÁ, A. *Zlepšování podnikových procesů.* Praha: Grada, 2011. ISBN 978-80-247-3938-0.

VOŘÍŠEK, J. *Strategické řízení informačního systému a systémová integrace.* Praha: Management Press, 1997. ISBN 80-85943-40-9.

---

**Předběžný termín obhajoby**

2020/21 LS – PEF

**Vedoucí práce**

Ing. Jan Tyrychtr, Ph.D.

**Garantující pracoviště**

Katedra informačního inženýrství

Elektronicky schváleno dne 19. 11. 2020

**Ing. Martin Pelikán, Ph.D.**

Vedoucí katedry

Elektronicky schváleno dne 19. 11. 2020

**Ing. Martin Pelikán, Ph.D.**

Děkan

V Praze dne 24. 03. 2021

**Čestné prohlášení**

Prohlašuji, že svou diplomovou práci "Využití microservices architektury pro automatizaci firemních procesů" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 30.3.2021 _____

**Poděkování**

Rád bych touto cestou poděkoval Ing. Janu Tyrychtrovi, Ph.D. za ochotu, věcné připomínky a odborné vedení práce.

# Využití microservices architektury pro automatizaci firemních procesů

**Abstrakt**

Tato diplomová práce se zabývá využitím návrhového vzoru mikroslužeb za účelem automatizace firemních procesů. V teoretické části je popsaná problematika firemních procesů včetně technologií určených k jejich implementaci.

V praktické části se autor zabývá analýzou, návrhem a implementací konkrétních firemních procesů dle potřeb firmy. Po představení společnosti, pro kterou je proces vytvářen, je popsán samotný průběh životního cyklu vývoje konkrétního procesu za využití architektury mikroslužeb.

**Klíčová slova:** BPMN, Firemní procesy, Automatizace, Mikroslužby, Java, Spring Boot

# Use of microservices architecture for automation of business processes

**Abstract**

Theme of this diploma thesis is how to use microservices architecture for Automation of business processes. In theoretical part of this thesis author describes the business process matters including technologies which can be used for implementation.

In practical part of this thesis author writes about analysis, design and implementation of specific business processes needed for specific company. Firstly, author introduces the company and after that he writes about life cycle of development of the specific business process where he used microservices architecture.

**Keywords**: BPMN, Business processes, Automation, Microservices, Java, Spring Boot

# Obsah

# Seznam obrázků

# Seznam tabulek

# 1 Úvod

Tato práce je zaměřená na velmi aktuální téma, které řeší většina firem a společností na celém světě. Jedná se o problematiku firemních procesů, přesněji řečeno o jejich optimalizaci a především automatizaci.

Pokud má být firma konkurenceschopná, musí stále zvyšovat svou výkonnost, kvalitu svých služeb a zároveň se vyvarovat navýšení nákladů. Z tohoto důvodu je pro firmy žádoucí se naučit správně používat jednotlivé procesy, které tvoří celou fungující infrastrukturu podniku. Bez této schopnosti firma nebude konkurenceschopná a tím pádem ani úspěšná. Pouze za využití správně vytvořené procesní firemní struktury, může zvýšit efektivitu a kvalitu svých služeb za prakticky stejné náklady.

Cílem této práce je návrh systému pro automatizaci firemních procesů s využitím návrhového vzoru mikroslužeb.

Praktická část se skládá především z výběru vhodných technologií v závislosti na požadavcích klienta a následného popsání vývoje firemního procesu, tj. analýzy, návrh a implementace.

Teoretická část se věnuje popisu problematiky firemních procesů včetně různých technologií, které lze využít při jejich vývoji.

Důvodem k výběru tohoto konkrétního tématu, je několikaletá zkušenost autora s různými BPMN nástroji, které jsou používány v různých nadnárodních společnostech.

# 2 Cíl práce a metodika

## 2.1 Cíl práce

Diplomová práce je tematicky zaměřena na technologie určené k automatizaci firemních procesů. Hlavním cílem práce je návrh systému pro automatizaci firemních procesů s využitím návrhového vzoru mikroslužeb.

Dílčí cíle diplomové práce jsou:

- Vytvoření literární rešerše týkající se problematiky firemních procesů a návrhového vzoru mikroslužeb.
- Analýza vybraných technologií dostupných pro automatizaci firemních procesů.
- Návrh systému určeného k automatizaci firemních procesů s využitím návrhového vzoru mikroslužeb.
- Ověření návrhu na vybraném firemním procesu.
- Syntetizace výsledků práce, formulace přínosů a závěry práce.

## 2.2 Metodika

Metodika řešené problematiky diplomové práce vychází ze studia a analýzy odborných informačních zdrojů, stejně tak jako ze zkušeností převzatých z praxe. Praktická část je zaměřena na vypracování případové studie zaměřené na jednu konkrétní společnost, pro kterou je vyhotoven návrh řešení automatizace procesů. Na základě syntézy teoretických poznatků a výsledků praktické části práce jsou formulovány závěry diplomové práce.

### 2.2.1 Notace BPMN

BPMN prvky se dají rozdělit do čtyř základních kategorií, kterými jsou: Událost (event), Aktivita (activity), Brána (gateway) a Tok (flow). Tyto kategorie lze dále rozdělit na další podkategorie. [5]

#### 2.2.1.1 Událost

Událost je úkon či okamžik, který nastane v průběhu procesu, čímž se ovlivní jeho celkový průběh, a to tak, že buď začíná, končí, nebo představuje tzv. průběžnou činnost. Události se obecně značí kroužkem, které dále dělíme na tři podkategorie:

**Tabulka 1 BPMN notace – Událost**

| Název | Popis | Značení |
|---|---|---|
| Počáteční událost | Označuje místo a případně i typ události (v konkrétní čas, zasláním konkrétního signálu…), čím daný proces začne |  |
| Průběžná událost | Má vliv na průběh procesu, ale přímo nezačíná či neukončuje proces |  |
| Konečná událost | Označuje místo a případně i typ události (zaslání zprávy, chybový konec procesu…), kterou daný proces skončí |  |

[5]

#### 2.2.1.2 Aktivita

Aktivita je činnost, která musí být vykonána. Jedná se o obecný pojem pro práci, která musí být provedena společností v rámci dosáhnutí cíle procesu. Tyto aktivity můžeme rozdělit na atomické činnosti a sub procesy neboli činnost složenou z atomických úkolů. Obecně se aktivity znázorňují jako obdélníky se zaoblenými rohy.

**Tabulka 2 BPMN notace – Aktivita (podle [5])**

| Název | Popis | Značení |
|---|---|---|
| Sub proces | Znázorňuje vykonání skupiny činností zahrnutých do tzv. sub procesu |  |
| Úkol | Znázorňuje atomickou činnost |  |

#### 2.2.1.3 Brána

Brány se využívají k větvení či naopak sloučení toků procesů v závislosti na definovaných podmínkách. Obecně se brány značí pomocí kosočtverce. Dělíme je na čtyři základní typy: exkluzivní, inkluzivní, komplexní a paralelní brány.

**Tabulka 3 BPMN notace – Brána (podle [5])**

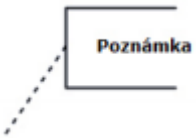
| **Název** | **Popis** | Označení |
|---|---|---|
| Exkluzivní brána závisející na datech | Exkluzivní brány závisející na datech se použijí v případě, kdy je žádoucí, aby tok pokračoval dále pouze jednou cestou |  |

| | | |
|---|---|---|
| | v závislosti na stanovených podmínkách. | |
| Exkluzivní brána závisející na událostech | Stejně jako typ exkluzivní brány závisející na datech, tak i typ závisející na událostech se využije v případě, kdy má tok pokračovat pouze jednou cestou. Ovšem v tomto případě se o směru toku rozhodne na základě výstupu události. |  |
| Inkluzivní brána | Inkluzivní brána je obdobou exkluzivní brány. Hlavním rozdílem je, že tok může pokračovat vícero cestami, v závislosti na stanovených podmínkách. |  |
| Paralelní brána | Paralelní brány se užívají v případě, kdy má tok pokračovat vícero cestami ve stejný okamžik. |  |
| Komplexní brána | Komplexní brány se používají na místech, kde nelze použít žádný z předchozích typů bran a kde dochází k dělení cest ve více branách. |  |

#### 2.2.1.4 Artefakty

Jedná se o grafický prvek, který poskytuje informace o celém, či o části procesu. Artefakty dělíme na tři skupiny: datové objekty, seskupení a poznámka.

**Tabulka 4 BPMN notace – Artefakty (podle [5])**

| Název | Popis | Označení |
|---|---|---|
| Datový objekt | Popisují data, která jsou vyžadována na vstupu aktivity či jsou výstupem dané aktivity. |  |
| Seskupení | Seskupení označuje skupinu prvků procesu bez vlivu na funkcionalitu procesu. Užívá se pro analytické účely a také dokumentaci. |  |
| Poznámka | Poskytuje informace o konkrétním prvku procesu za účelem zpřehlednění a snazším pochopení významu takového prvku v procesu. |  |

#### 2.2.1.5 Spojovací objekty

Jak již název napovídá, jedná se o objekty, které propojují jednotlivé části procesu do uceleného celku. Obecně se značí jako šipka. Dělíme je na tři základní typy: sekvenční tok, tok zprávy a asociační tok.

**Tabulka 5 BPMN notace – Spojovací objekty (podle [5])**

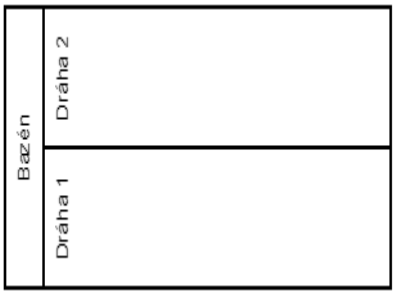
| Název | Popis | Značení |
|---|---|---|
| Sekvenční tok | Zobrazuje sekvenci činností v procesu. |  |
| Tok zprávy | Zobrazuje přenos zprávy |  |

| | | |
|---|---|---|
| | mezi účastníky procesu. | |
| Asociační tok | Zobrazuje propojení prvků s artefakty či vysvětlivkami. |  |

2.2.1.6 Plavecké dráhy

Plavecké dráhy se užívají ke kategorizaci jednotlivých účastníků a činností v procesu. Plavecké dráhy dělíme na dva základní typy: bazén a dráha.

**Tabulka 6 BPMN notace – Plavecké dráhy (podle [5])**

| Název | Popis | Zobrazení |
|---|---|---|
| Bazén | Bazén označuje účastníka, který je uveden v záhlaví bazénu. Bazén je vždy tvořen alespoň jednou drahou. V každém bazénu je vždy pouze jeden proces. Jednotlivé bazény mezi sebou mohou komunikovat prostřednictvím toku zpráv. |  |
| Dráha | Dráha slouží ke kategorizaci účastníků v rámci konkrétního bazénu. Jednotlivé dráhy mezi sebou mohou komunikovat za pomoci sekvenčního toku dat. |  |

### 2.2.2 Technologie

Níže autor uvádí jednotlivé technologie, které byly užity v praktické části této diplomové práce.

#### 2.2.2.1 Camunda

Autor vybral jako stěžejní technologii pro automatizaci podnikových procesů nástroj Camunda.

Na rozdíl od jiných nástrojů, jejichž použití zvažoval, např. IBM BPM či Tibco BW je zde výhoda, že se jedná o open source řešení.

Camunda je open source framework založený na jazyce Java, který slouží k modelování podnikových procesů a jejich následé automatizaci za pomoci BPMN, CMMN či DMN notace. V rámci této diplomové práce se však využije pouze BPMN notace. [6]

##### 2.2.2.1.1 Camunda Modeler

Camunda modeler je nástroj přímo určený k modelování či úpravě podnikových procesů. Jelikož výstup z tohoto nástroje má podobu BPMN či DMN diagramu, vyhovuje jak vývojářům, tak i managerům, kteří se v něm taktéž snadno zorientují. Značnou výhodou tohoto stylu vývoje nové funkcionality je, pouhé „přetahování“ již vytvořených prvků, které se následně mezi sebou propojují. Díky tomu může jisté drobné změny provést i člověk, který neumí programovat. V případě potřeby Camunda Modeler nabízí i možnost exportu procesu do XML formátu. [6]

#### 2.2.2.2 Java

Java je objektově orientovaný programovací jazyk vyvinutý společností Sun Microsystems. Jehož základy byly definovány již v roce 1991 jejími zaměstnanci Patrickem Naughtonem, Edem Frankem, Christem Warthem, Mike Sheridanem a Jamesem Golsingem.

Tento programovací jazyk se mezi vývojáři poměrně rychle „uchytil“ a stal se jedním z nejpoužívanějších programovacích jazyků pro své všestranné využití, jednoduchost a mnoho dalších výhod, mezi které patří především:

- Jedná se o vyšší, obecně použitelný programovací jazyk s vysokým stupněm zabezpečení
- Je objektově orientovaný, ale umožňuje i klasické procedurální programování
- Jeho pomocí vytvořené programy jsou zcela portabilní (program vytvořený pod MS Windows bez problémů funguje pod Unixem a naopak)
- Syntaxe výrazů a příkazů vychází z jazyka C; přičemž přechod z C nebo C++ je tedy jednodušší, než z jiných produktů
- Základní implementaci jazyka (JDK – Java Development Kit) firmy Sun lze pro prostředí MS Windows i Unix zdarma stáhnout ze stránek firmy Oracle (před akvizicí Sun) [7]

Tento programovací jazyk se dělí do tří skupin:

- Java SE (Standart Edition) je základní edicí, využitelnou především při vytváření softwaru určeného pro desktop či server.
- Java ME (Micro Edition) je podmnožinou Java SE, verze je užívána pro vývoj softwaru pro malá zařízení s omezenými prostředky.
- Java EE (Enterprise Edition), tato verze se používá především při vývoji informačních systémů, či jiných podnikových aplikací. Obsahuje již velkou škálu knihoven. [9]

2.2.2.2.1 Způsob zpracování programu v“ Javě“

Standardní postup zpracování programu v programovacím jazyce Java prochází pěti fázemi – editováním, překladem (kompilací), zavedením (load), ověřováním (verifikací) a prováděním. Tyto čtyři fáze jsou běžné i v jiných programovacích jazycích, avšak fáze ověřování je nová. Umožňuje dosáhnout velmi vysoké bezpečnosti spuštěného programu, především pro uživatele, jež daný program spouští. [7]

Další „novinkou“, díky které se Java liší od ostatních programovacích jazyků je, že překlad neprobíhá do jazyka relativních adres, což je v podstatě strojový kód, ale do tzv. byte-code. Tento způsob překladu je možné psát na různých počítačích a programátor tak nemusí řešit, kde se jeho program bude využívat.

Tento kód je uložen v souboru s vyhrazenou příponou class, posléze je z disku zaváděn do paměti počítače, kde současně probíhá ověření byte-codu, což je možné jen díky jeho nezávislosti na platformě. Po ověření je program spouštěn pomocí interpretu. [9]

##### 2.2.2.2.2 Typy programů

Programy v Javě se dle cílového použití dělí na dvě velké skupiny. První jsou aplikace, které se dají chápat jako běžné programy známé z jiných programovacích jazyků a druhou skupinou jsou aplety (applets), které se používají na WWW stránkách. Pro obě skupiny platí stejná pravidla, která se týkají editování a překladu. Fáze zavedení se ale liší, jelikož aplety na WWW stránce jsou zaváděny do paměti počítače pomocí standardních prohlížečů typu Netscape nebo MSIE a fáze ověření je odlišná, jelikož pro aplety platí přísnější pravidla než pro aplikace. Provádění aplikací je u obou skupin stejné, program je interpretován. [7]

#### 2.2.2.3 Framework

Framework je označení pro softwarovou strukturu, která slouží jako podpora při programování. Může v sobě zahrnovat různé podpůrné programy, knihovny či podporu pro návrhové vzory. Účelem frameworků je převzetí běžně se vyskytujících problémů v dané oblasti, a tím usnadnit práci vývojáři, který se posléze může soustředit čistě na svůj pracovní úkon. Skládá se ze dvou částí:

- Frozen spots – definují architekturu softwarové struktury, základní komponenty a vztahy mezi nimi. Tyto části jsou neměnné.
- Hot spots – jsou komponenty přímo navazující na kód programátora, a tudíž jsou v každé aplikaci naprosto jedinečné. [9]

##### 2.2.2.3.1 Hibernate

Hibernate je Framework napsaný v jazyce Java umožňující tzv. objektově – relační mapování (ORM). Pomocí jeho využití je možné zachovat stejný stav mezi dvěma spuštěnými aplikacemi. Jinými slovy, přemapujeme objekty v jazyce Java, na entity v relační databázi. K dosažení cíle používá tzv. mapovací soubory, jejichž obsah udává, jakým způsobem se budou data transformovat do databáze a naopak. Tyto soubory jsou formátu XML, kdy pro jednu třídu máme jeden soubor, obyčejně je umístěn ve stejném adresáři jako samotná třída. Ovšem můžeme použít i jiný způsob, např. využít anotace místo mapovacích souborů, přičemž anotace lze použít dvěma způsoby, a to u atributů nebo u getterů. Díky tomuto frameworku si programátor usnadní práci, jelikož se již nemusí zabývat transformováním objektů do relací ručně. [10]

#### 2.2.2.3.2 Spring

Spring je open-source Framework, a je určený pro vývoj J2EE aplikací. Vznikl v říjnu roku 2002 a velmi rychle si našel u vývojářů oblibu. Tento framework byl vytvořen za cílem zjednodušení tvorby softwaru při užívání návrhových vzorů, kvalitní dekompozice kódu, či užívání rozhraní místo konkrétních tříd.

Jádro Springu bylo vytvořeno za užití návrhového vzoru Inversion of Control (IoC) kontejner. IoC kontejner funguje na principu, přesouvání zodpovědnosti za vytvoření a provázání objektů z aplikace na Framework, přičemž objekty lze získat vsazováním závislostí. Objekty vytvořené pomocí kontejneru jsou nazývány Beans a jsou běžně vytvářeny po načtení XML souboru obsahujícího definice pro jednotlivé Beans. Spring sám o sobě neřeší žádné problémy, místo toho využívá dobře fungující open-source nástroje, které jsou v něm integrovány. [11]

Spring se během své existence rozrostl do velkých rozměrů, přičemž se dělí na jednotlivé na sobě nezávislé moduly. Díky tomu, že se jedná o modulární Framework, postačí využít jen několik potřebných Spring modulů, které potřebujeme k vytvoření dané aplikace.

Mezi moduly patří například:

- JDBC
- ORM
- AOP
- OXM
- JMS [11]

### 2.2.2.4 Amazon Web Services


Obrázek 1 AWS logo [27]

Amazon Web Services neboli zkráceně AWS je komplexní cloudová platforma od firmy Amazon, založena Jeffem Bezosem v roce 1994.

Počátek služby AWS se datuje do roku 2000, kdy firma Amazon rozšířila svou oblast působnosti i na poskytování online nakupování. V této době soustavně narůstaly požadavky na celý elektronický komplexní systém.

Po jeho několikaletém zdokonalování se firma v roce 2006 rozhodla poskytovat navržené řešení cloudových služeb. Tento produkt nabízí jak výpočetní výkon, tak i databázové či analytické nástroje a mnoho dalšího. [25] [26]

#### 2.2.2.4.1 Výpočetní systémy

Do kategorie výpočetních systémů lze zařadit primárně nástroje EC2 a Lambda. Nástroj EC2 neboli Elastic Compute Cloud zajišťuje škálovatelnost výpočetní kapacity v cloudovém prostředí. Jedná se o rozhraní, pomocí kterého může uživatel spravovat kompletní výpočetní kapacitu. Například může vytvářet / odstraňovat jednotlivé instance virtuálních serverů, případně je spouštět / zastavovat, nebo jim i přiřazovat zdroje. [26]

Nástroj Lambda slouží ke spouštění naprogramovaného kódu, díky čemuž odpadá starost o dostatek zdrojů, nasazení, dostupnost atd. Spouštění tohoto kódu lze provést pomocí RESTového rozhraní. Tato služba je určená především pro spouštění rychle vykonavatelného kódu, kvůli čemuž má i přidělený omezený čas, po který se může program vykonávat. [26]

#### 2.2.2.4.2 Datová úložiště

Amazon EBS neboli Amazon Elastic Block Store poskytuje uživateli trvalé blokové úložiště dat. Primárně tento nástroj slouží jako místo pro instalaci jednotlivých virtuálních počítačů služby Amazon EC2. Velkou výhodou tohoto úložiště je možnost automatické replikace dat, díky čemuž je velmi odolná vůči výpadkům či ztrátám dat.

Amazon S3 neboli Amazon Simple Storage Service je cloudové úložiště dat. Jedná se o zabezpečené škálovatelné úložiště, které nabízí mnoho možností, jak s přístupovými datovými oprávněními pracovat. Je to ideální místo pro uložení všech souborů od datasetů až po firemní dokumenty. Jeho výhoda spočívá v neomezeném prostoru pro ukládání dat, přístupnosti, ale i zabezpečení, a díky možnosti automatizace RESTových endpointů přístupu do samotného úložiště. [26]

##### 2.2.2.4.3 Databázová úložiště

Amazon RDS neboli Amazon Relational Database Service je služba poskytující rozhraní pro správu relačních databázových systémů. Pomocí tohoto nástroje je může uživatel snadno upravovat a zároveň je zajištěna i vysoká rychlost a dostupnost. Uživatel má dále dostupné i běžné databázové systémy, mezi které patří například Microsoft SQL server, Oracle, PostgreSQL či MySQL. [26]

Amazon DynamoDB je NoSQL databázové úložiště, které se nejčastěji užívá v situacích, kdy je potřeba co nejrychlejší přístupová doba ke čtení či zápisu dat do databázového systému. [26]

Amazon Aurora je specifický relační databázový nástroj, který vychází z MySQL databází. Jeho výhodou je rychlost a spolehlivost, při zachování minimálních nákladů. Uživatel k tomuto produktu může přistupovat pomocí Amazon RDS nástroje. [26]

##### 2.2.2.4.4 Síťové služby a doručování obsahu

Amazon CloudFront je služba sloužící k doručování obsahu mezi zdrojem, a cílem požadavku. Tato služba využívá geografického umístění mezi lokalitami AWS, aby byla co možná nejefektivnější. [26]

Amazon API Gateway je nástroj určený k orchestraci kompletního rozhraní pro programování aplikací (REST API). Kromě základních funkcí k vytváření a provozování těchto rozhraní nabízí i jejich monitoring a zabezpečení. [26]

##### 2.2.2.4.5 Zabezpečení a správa identit

AWS IAM neboli Identity & Access Management je služba poskytující pokročilé rozhraní určené pro správu identit a přístupových oprávnění k jednotlivým funkcím či celým produktům AWS. Díky tomu, že tato služba do jisté míry orchestruje všechny ostatní produkty, se jedná o jednu ze stěžejních služeb systému AWS. [26]

#### 2.2.2.4.6 Správa AWS

Díky komplexnosti systému AWS se Amazon rozhodl vytvořit i několik různých způsobů, jak nastavení celého systému spravovat. Nejjednodušší je užití nástroje AWS Management Console.

##### 2.2.2.4.6.1 AWS Management Console

Jak již bylo uvedeno výše, AWS Management Console je nejjednodušším způsobem, jak spravovat nastavení celého AWS. Jedná se o webový portál, kdy po autorizaci pomocí IAM služby má uživatel přístup ke všem službám, které AWS nabízí. Navigace v tomto nástroji je velmi snadná díky vyhledavači konkrétních produktů, u kterých chce příslušný uživatel upravit nastavení. Jeho samotná úprava probíhá za pomoci již předpřipravených formulářů.

Bohužel vzhledem k tomu, že administrátor musí vše procházet a nastavovat ručně v rámci formulářů, je zde jistá časová zátěž. [26]

##### 2.2.2.4.6.2 AWS Command Line Interface

Efektivnější možností pro správu nastavení systému AWS je AWS Command Line Interface neboli CLI, který uživateli nabízí možnost úprav nastavení pomocí příkazové řádky. Jedná se o uživatelské rozhraní určené především pro zkušenější uživatele. Jednotlivé příkazy jde mimo jiné i spojovat do ucelených souborů, díky čemuž lze vytvářet i automatizované skripty.[26]

##### 2.2.2.4.6.3 SDK a ostatní

Nastavení služeb AWS lze spravovat i za pomoci programovacího REST API rozhraní. Díky velkému množství endpointů lze zasílat GET a POST požadavky, pomocí kterých uživatel nakonfiguruje veškeré služby systému AWS.

Tento způsob nastavování je vhodný především k integraci s externími systémy či automatizaci některých firemních procesů. [26]

# 3 Teoretická východiska

## 3.1 Procesní řízení

Procesní řízení je činnost, kterou denně provádí ať již vědomě či nevědomě každý firemní manažer. V praxi se jedná o řízení podniku takovým způsobem, v němž jednotlivé procesy hrají klíčovou roli.

Jde o poměrně nový způsob, který si klade za cíl především zpřehlednění chování jednotlivých částí firmy na elementární úrovni, a vytvoření podkladů pro její další rozvoj a s tím spjatý růst.

Tento přístup má mnoho různých výhod, kterými jsou např. [1]:

- Detailní popis pracovních postupů: které vedou ke snazšímu pochopení fungování daného procesu, a tudíž jej lze i snáze zmodifikovat
- Transparentnost organizace: jedná se o zpřehlednění nejen interních procesů firmy, ale i o vztahu k externímu okolí, což jsou především dodavatelé, zákazníci či různí obchodní partneři. Toto je pro podnik také velice důležité, jelikož se běžně stává, že jednotlivé firmy spolu spolupracují. K zajištění dlouhodobě příznivých a efektivních vztahů je nutné, aby tyto vztahy byly vzájemně transparentní při současném akceptování potřeb všech zainteresovaných stran. Namodelované procesy ve vztahu k ostatním organizacím je umožňují i lépe definovat.
- Rychlá reakce na změny: pokud má podnik detailně a přehledně popsané své jednotlivé procesy, nečiní mu takové problémy přizpůsobení se stále se měnícím podmínkám trhu a být stále konkurenceschopný.
- Dokumentace know – how: síla a prosperita každého podniku tkví v informacích, respektive v postupu, jak provádět pracovní činnosti co nejefektivněji. V případě že má firma detailně popsané své jednotlivé procesy, má i vhodně uloženou dokumentaci svého know – how.
- Přesná definice povinností: díky důslednému procesnímu řízení je ihned zřejmé, kdo za proces, jako celek zodpovídá. Kromě toho je díky procesní mapě i jasné, kdo má na starost, jakou dílčí část procesu, např. naskladnění zboží, expedici objednávek atd. Toto vše přispívá ke striktnímu dodržování pracovních povinností, jelikož je to snadno zpětně dohledatelné.

Aby procesní řízení přinášelo žádaný efekt, je nutné dodržet několik základních pravidel. V žádném případě by nemělo docházet ke konfliktům mezi jednotlivými procesy a všechny by měly být, pokud možno co nejjednodušší. Neměly by se zde vyskytovat žádné složitosti ani duplicitní části logiky či duplicitní procesy. Firma by neměla užívat procesy, které pro ni nemají žádný přínos, tj. nadbytečné procesy. Každý podnik by si měl zavést i nějaký svůj vnitřní standart pro vytváření jednotlivých procesů a zároveň by měl i vždy uvést zaměstnanec, který bude za konkrétní proces zodpovídat.

Hlavním cílem procesního řízení je řízení a s tím spjatý monitoring jednotlivých firemních procesů, což by mělo vést k růstu produktivity a zvýšení kvality nabízených výrobků / služeb, díky čemuž by měla firma prosperovat a dále se rozvíjet.

Procesní přístup řízení podniku přináší i možnost plné či alespoň částečné automatizace mnoha firemních procesů např. předávání dat, zálohování atd. Procesy, které lze alespoň částečně zautomatizovat můžeme označit jako tzv. workflow.

Při zautomatizování procesů se užívá tzv. workflow software, pomocí kterého si lze nadefinovat veškeré datové toky procesu včetně jednotlivých podmínek. [1]

## 3.2 Podnikové procesy

Každá firma jako celek je tvořena souborem jednotlivých činností, které na sebe navzájem různě navazují.

Tyto činnosti označujeme jako tzv. Business procesy. Jedná se o řadu logicky na sebe navazujících elementárních úkonů, které mají vést k předem stanovenému cíli. Tyto cíle se mohou lišit v závislosti na různých vstupech či výstupech z jednotlivých částí procesu.

Každá věc i každý zaměstnanec ve firmě je součástí mnoha různých procesů, i když si to mnohdy nemusí ani uvědomovat. Aby firma mohla prosperovat, a být konkurenceschopná je důležité, aby si jednotlivé procesy důkladně zmapovala, díky čemuž je může i snáze měnit za účelem zefektivnění.

### 3.2.1 Typy procesů

Procesy lze dle jejich využití roztřídit do tří základních kategorií.

- Hlavní
- Řídící
- Podpůrné

Všechny tyto typy mají svůj nezastupitelný význam a firma nemůže existovat bez zástupců procesů v jednotlivých kategoriích. [2]

#### 3.2.1.1 Hlavní

Mezi hlavní procesy řadíme ty, které jsou pro existenci firmy klíčové. Každý podnik, klade důraz především na tento typ procesů, neboť vytváří firmě zisk. Jako příklad můžeme uvést nacenění nabídky klientovi či prodej produktu. Procesy lze rozpoznat podle těchto klíčových vlastností [2]:

- Vytváří zisk
- Přijde s nimi do styku zákazník
- Bývají komplikované
- Management společnosti je snadno identifikuje

#### 3.2.1.2 Řídící

Tyto procesy jsou nutné k řízení chodu společnosti i přesto, že nevytváření žádný zisk. Jsou nesmírně důležité, jelikož hlavní procesy by bez řídících nemohly fungovat. Mezi řídící procesy patří například vytváření strategie. [2]

#### 3.2.1.3 Podpůrné

Podpůrné procesy jsou hned po hlavních nejdůležitější. Jako příklad můžeme uvést nákup materiálu, vyplácení mzdy zaměstnancům atd. Ty častokrát bývají společné pro celou firmu. [2]

## 3.3 Návrh procesu

### 3.3.1 Analýza procesů

Analýza procesů také někdy nazývána procesní analýza slouží k pochopení řízení, a především ke zlepšení fungování jednotlivých procesů v rámci firmy. Proces se analyzuje na elementární úrovni, tj. jednotlivé dílčí části, ze kterých je jako celek tvořen. V rámci této analýzy jsou mimo jiné i přesně definované vstupy a výstupy jednotlivých částí, včetně vhodné reakce na různé výstupy / vstupy. Jinými slovy je to detailně popsaný postup.

Analýza procesů se provádí primárně ze tří hlavních důvodů:

- Dokumentace fungování procesů
- Příprava pro automatizaci procesů tzv. workflow
- Z důvodu optimalizace [3]

### 3.3.2 Optimalizace procesů

Data získaná za pomoci analýzy procesů poslouží při jejich optimalizaci. V rámci ní se provádí různé návrhy za účelem co možná nejvyššího zefektivnění procesu a tím i zefektivnění fungování firmy. Díky takovéto optimalizaci může podnik kupříkladu snížit náklady, a naopak zvýšit svou efektivitu. [4]

### 3.3.3 Modelování procesů

Pro lepší představu fungování jednotlivých částí konkrétního procesu se kromě slovního popisu využívají i různé vizuální nástroje, kterými jsou například UML, DFD, DMN, BPMN atd.

V praxi se řadí mezi nejpopulárnější především UML a BPMN diagramy.

## 3.4 Business Process Model and Notation (BPMN)

Business Process Model and Notation bylo vyvinuto společností Object Management Group (OMG). Jedná se o poměrně nový typ modelu, jelikož byl navržen teprve v roce 2005 a jako standart se začal používat v následujícím roce, 2006. V té době ovšem tato zkratka měla trochu jiný význam, znamenalo to Business Process Modelling Notation. Stávající název tento typ notace získal až v lednu 2011, kdy vzniklo BPMN 2.0.

Tento typ notace vznikl především sloučením a úpravou již existujících notací, například UML.

BPMN si primárně klade za cíl poskytnout notaci, které snadno porozumí všichni podnikoví uživatelé, kterými jsou například business analytici, IT architekti, vývojáři či lidé, kteří přímo nepracují v IT odvětví, jako jsou různí manageři či obchodníci.

BPMN procesy se skládají z jednotlivých prvků a vazeb mezi nimi, které představují firemní aktivity či položky práce vykonávané zaměstnanci či stroji. [5]

## 3.5 Mikroslužby návrhový vzor

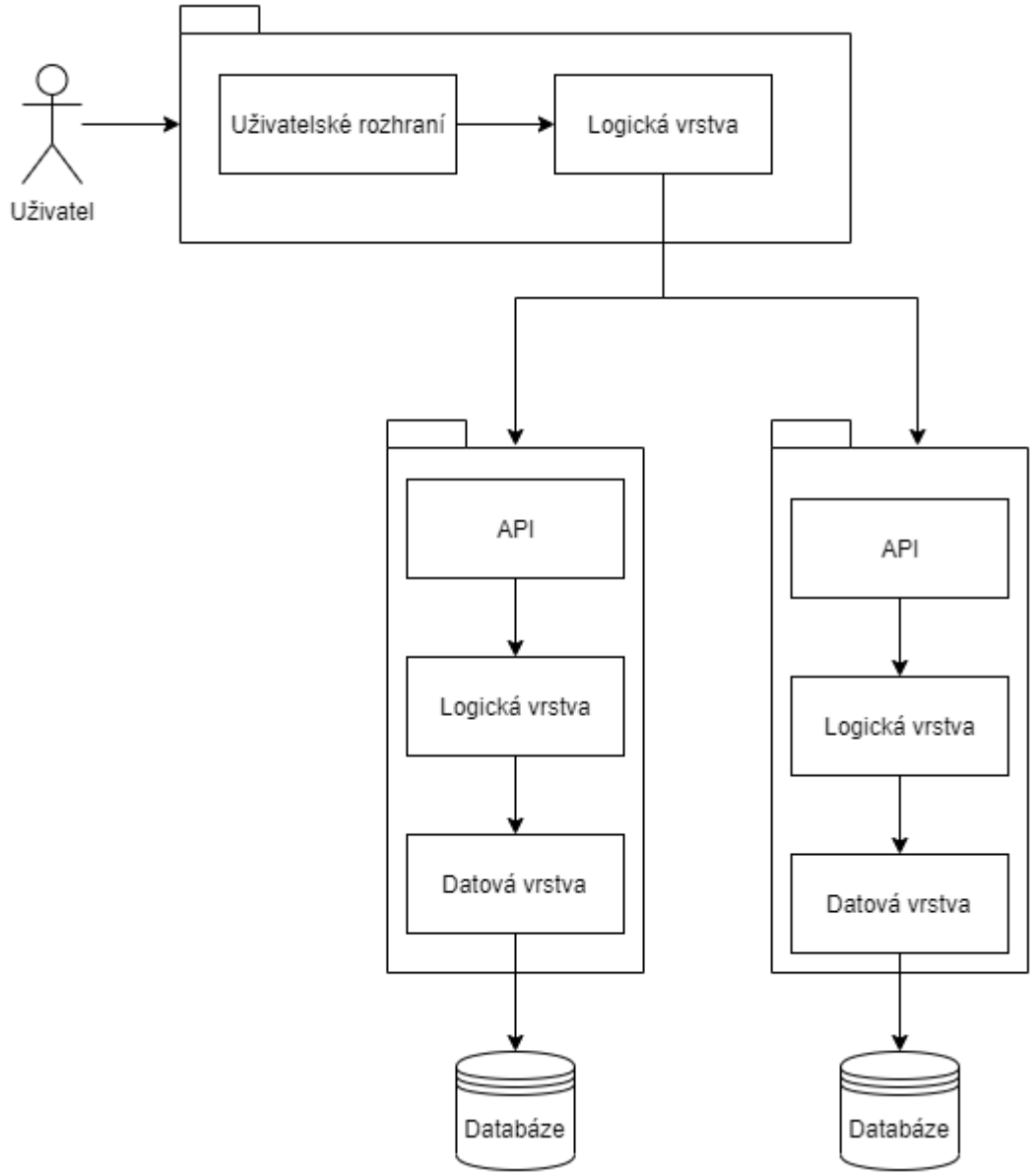
Mikroslužby jsou jedním z mnoha návrhových vzorů užívaných při vývoji softwaru.

Jak již název napovídá, hlavní vlastností tohoto návrhového vzoru je rozdělení vyvíjeného softwaru na jednotlivé funkční komponenty tzv. mikroslužby. Tyto dílčí části jsou absolutně izolované od ostatních, včetně datové vrstvy. Díky tomu se nemůže stát, že by dvě různé mikroslužby přistupovaly ke stejným datům uloženým v databázi.

Principiálně lze do mikroslužby přistoupit pouze prostřednictvím veřejného API. Ovšem může nastat případ, kdy spolu musí spolupracovat vícero mikroslužeb. Často se tak děje z důvodu získání potřebných dat z databáze, kterou spravuje jiná mikroslužba. Tato komunikace napříč mikroslužbami se primárně řeší vystavením interního zabezpečeného API volání, které není přístupné zvenčí, tudíž ho uživatel nemůže využít.

Tímto se razantně liší typické monolitické architektury aplikací, kde je celá funkcionalita tvořena jedním velkým celkem. [13] [14] [15]


Obrázek 2 Typická architektura monolitické aplikace




Obrázek 3 Typická architektura aplikace složené z mikroslužeb

### 3.5.1 API rozhraní

API rozhraní je součástí prakticky všech aplikací založených na principu mikroslužeb. API rozhraní slouží jako vstupní bod do systému mikroslužeb. Jedná se o velmi důležitou komponentu, která zastává několik rolí. Kromě toho, že přes toto rozhraní prochází jednotlivé volání do systému, lze tímto způsobem i snáze zabezpečit vstup do mikroslužeb. Mezi další jeho výhody patří například možnost routování, tj. vyvolání operace mikroslužby s odlišným názvem, než je vytvořen v rámci mikroslužby. Jako jedna z největších výhod se však jeví možnost sdružení jednotlivých volání mikroslužeb do ucelené flow. Zde stačí vyslat pouze jeden příkaz směrem na backend, kde se posléze vytvoří i několik různých volání napříč mikroslužbami, než se vrátí nazpět odpověď. [13]

#### 3.5.1.1 REST API

REST neboli Representational State Transfer je architektura rozhraní navržená pro distribuované prostředí, které využívá ke své komunikaci zejména protokol HTTP. První návrh tohoto typu architektury vznikl již v roce 2000 v rámci disertační práce Ray T. Fieldinga.

Rozhraní REST je použitelné pro jednotný a snadný přístup ke zdrojům (resources), které jsou identifikovány za pomocí URI neboli Uniform Resource Identifier. Na rozdíl od jiných typů služeb využívá REST širší spektrum metod protokolu HTTP při práci se zdroji.

- GET – Metoda sloužící k získání dat z určitého zdroje.
- POST – Metoda sloužící k vložení nového záznamu do cílové kolekce.
- PUT – Tato metoda je velmi podobná metodě POST, ovšem liší se v použití. Užívá se především k úpravě již existujícího záznamu v kolekci.
- DELETE – Jak je již z názvu zřejmé tato metoda slouží ke smazání záznamu z kolekce.

Pro výměnu dat se můžou v případě REST architektury použít různé datové formáty. Volba konkrétního formátu je většinou dána především užitím programovacího jazyka. Ovšem nejčastějším formátem dat v případě REST architektury je JSON. [31]

#### 3.5.1.1.1 JSON

JSON neboli JavaScript Object Notation je standardizovaný jednoduchý textový formát určený pro výměnu dat.

Tento datový formát vznikl na podmnožině skriptovacího jazyka Javascript a jedná se o strukturu klíč – hodnota. Hodnoty mohou být několika typů, tj. textový řetězec (string), číslo, objekt, hodnoty true / false (boolean), prázdná hodnota (null) a pole hodnot.

Tento textový formát má několik pravidel, které musí být splněny pro to, aby se jednalo o validní soubor. [30]

- Všechny klíče musí být zapsány ve dvojitých uvozovkách.
- Hodnota datového typu string musí být uvedená ve dvojitých uvozovkách.
- Objekt musí být umístěn mezi složené závorky {}, jednotlivé atributy objektu jsou odděleny čárkou.
- Pole hodnot musí být umístěno mezi hranaté závorky []. Jednotlivé hodnoty v rámci pole jsou odděleny čárkou, ale za poslední hodnotou v poli čárka být nesmí.
- Číslo musí být zapsáno v desítkovém formátu a pro oddělení desetinné části musí být užita tečka. [30]

### 3.5.1.2 Dokumentace API

Jednotlivé mikroslužby vystavují přístupové body, tzv. endpointy, které lze provolat a tím spustit některou z operací, jež daná mikroslužba nabízí.

Ovšem v případě objemnějších aplikací můžeme mít i několik desítek různých mikroslužeb, přičemž každá z nich může mít vystavenou celou řadu různých end pointů. Z toho důvodu je nutné mít veškeré end pointy důkladně zdokumentované. Taková dokumentace by kromě přístupového bodu měla obsahovat i veškeré hodnoty, které jsou požadované na vstupu a zároveň i veškeré možné výstupy z mikroslužby.

V případě SOAP endpointů se takováto dokumentace řešila formou WSDL souborů, REST rozhraní ho zas pro změnu řešilo formou WADL souboru. V dnešní době se však tato dokumentace řeší formou, kterou lze kdykoliv vygenerovat přímo v rámci aplikace.

Příkladem takového strukturovaného popisu je například Swagger, který využívá formát YAML. Tento typ popisu je v praxi poměrně oblíbený, a tak byl vytvořen plugin napříč programovacími jazyky, kterého ho lze využít např. jak v jazyce Java, tak i v jazyce Javascript. [13]

### 3.5.2 Monitorování a notifikace

V případě, že dojde z jakéhokoliv důvodu k selhání mikroslužby a tím pádem i výpadku části celkové funkcionality, je nezbytné tuto skutečnost oznámit odpovědným pracovníkům, kteří za chod těchto mikroslužeb zodpovídají.

Z tohoto důvodu je nutné při běhu mikroslužeb zaznamenávat jejich aktivitu, a to především v produkčním prostředí. Na rozdíl od decentralizace tohoto návrhového vzoru je vhodné mít veškeré záznamy o aktivitě sjednocené do centrálního místa.

Vzhledem k tomu, že takovýchto záznamů může vzniknout velké množství, je vhodné využít některý z již existujících systémů určených ke správě logů. Mezi nejpoužívanější patří kupříkladu Logstash, ElasticSearch či Kibana.

Tyto nástroje mívají často vestavěné rozhraní REST API, pomocí kterého může uživatel zadávat jednoduché dotazy, pomocí kterých získá ze systému jen ty záznamy, které potřebuje.

Díky takto uchovaným záznamům o aktivitě každé z mikroslužeb je může snáze odpovědný pracovník zanalyzovat, snadno opravit, případně si i dané chybné chování znovu nasimulovat. [13]

### 3.5.3 Typy komunikace

Komunikace dělíme dle metody synchronizace přenosu na dva základní typy, na synchronní a asynchronní.

Synchronní komunikace je založena na zaslání požadavku a očekávání odpovědi. Během doby, kdy odesílatel vyčkává na vrácení odpovědi, musí být mezi odesílatelem a příjemcem požadavku aktivní spojení. Tento typ komunikace je vhodný, pokud očekáváme rychlou odpověď, např. získání dat z databáze.

Asynchronní komunikace je založena na pouhém zaslání požadavku bez čekání na odpověď. Díky tomu nemusí odesílatel udržovat aktivní komunikační spojení mezi ním a příjemcem požadavku. V případě, že by byla nějaká odpověď posléze potřeba, je nutné její zaslání realizovat přes jiný komunikační systém. K tomuto účelu se užívají různé

nástroje, které asynchronní zprávy ukládají do tzv. fronty, ze které je lze postupně zpracovávat. Takovým nástrojem je např. RabbitMQ. Asynchronní typ komunikace je vhodný zejména pro typy operací, kdy zpracování požadavku trvá delší dobu, např. zpracování videa, fotografie atd. [13] [14]

### 3.5.4 Klíčové výhody

Z předešlých kapitol lze vydedukovat několik hlavních výhod, které nabízí architektura mikroslužeb.

Díky tomu, že je aplikace jako celek rozdělena na dílčí logické části, je snazší na udržitelnost a lze i jednotlivé mikroslužby instalovat do produkčního prostředí. Zároveň se jedná i o „bezpečnější“ systém, jakákoliv případná chyba v kódu ovlivní pouze velmi omezenou část aplikace jako celku. V případě nutnosti lze jednotlivé mikroslužby napsat i v různých programovacích jazycích.

Vzhledem k tomu, že každá z nich spravuje pouze vlastí část databáze, je zde možnost přizpůsobit databázovou vrstvu potřebám konkrétní mikroslužby.

Jejich architektura nabízí poměrně velkou možnost agilního vývoje, jelikož každý tým či dokonce každý jednotlivec může jednu mikroslužbu spravovat. S tím je spjatá další z výhod tohoto návrhového vzoru a tím je rozdělení odpovědností, kdy je přesně dáno, který tým či jednotlivec za danou funkcionalitu zodpovídá.

Velkou výhodou je i možnost přiřazení rozdílných zdrojů, kdy např. mikroslužbě, která slouží pro složité matematické výpočty, může mít přiřazeno vícero potřebných zdrojů než té, která v podstatě jen získává data z databáze. [13]

### 3.5.5 Nevýhody

Vzhledem k tomu, že tento styl vývoje je postaven na rozdělení aplikace jako celku na dílčí části, může to sebou přinášet i některé nevýhody. Jednou z nich může být například složitost pochopení fungování celé aplikace.

Další problém může nastat na databázové vrstvě. Vzhledem k tomu, že každá mikroslužba si „obhospodařuje“ pouze svou část, může dojít k nekonzistenci dat. Například, v případě, kdy mikroslužba spravující uživatelské profily odstraní uživatele ze své databáze. Tato informace se následně nedostane k mikroslužbě, jež spravuje uživatelské příspěvky o uživateli (již smazaném), který je autorem jednoho z nich, uložených v její databázi.

Další nevýhodou může být i spojování jednotlivých meziprocesních volání do ucelené flow. Například když mikroslužba upraví strukturu odpovědi, která se používá při zaslání následujícího požadavku v jiné mikroslužbě, může dojít k chybovému stavu. Z toho důvodu je vhodné vytvářet jednotlivé verze endpointů tak, aby se jednotlivé mikroslužby mohly připravit na případnou změnu struktury příkazů či odpovědí a nemohlo tak docházet ke kolizím.

S meziprocesní komunikací souvisí i další z nevýhod tohoto návrhového vzoru a tím je rychlost. Aby došlo ke komunikaci mezi jednotlivými mikroslužbami, musí dojít ke kódování zprávy, jejím odeslání, přijetí, dekódování a zpracování. V případě synchronního typu komunikace se poté celý proces zopakuje při zasílání odpovědi. Z tohoto důvodu je jasné, že celková doba trvání operace využívající funkcionalitu umístěnou v různých mikroslužbách je o poznání delší než v případě monolitické aplikace.

Vzhledem k možnosti propojení jednotlivých mikroslužeb může být i složitější jejich jednotlivé nasazování, jelikož může dojít k předefinování pořadí. [13]

# Vlastní práce

## 3.6 Představení společnosti LutherOne a.s.

Předmětem praktické části této diplomové práce je návrh automatizace procesu aktivace nového zákazníka pro společnost LutherOne a.s.

Jedná se o mladou firmu tzv. start-up, která byla založena 1. srpna 2018 Jiřím Báčou. [18] Tato společnost se zabývá vývojem platformy, která si klade za cíl lépe využít potenciál zaměstnanců a tím zvýšit celkovou efektivitu. Tato platforma je cílena především na střední a velké podniky. Mimo jiné obsahuje celou řadu pro firmy zajímavých funkcí, kterými jsou například: sledování spokojenosti pracovníků v reálném čase, stálá zpětná vazba mezi pracovníky a vedením, kontinuální měření výkonu, sledování plnění stanovených cílů a v neposlední řadě i vnitropodniková sociální síť. Za pomoci těchto instrumentů by měla být firma schopna „podchytit“ zhoršující se vzájemné vztahy na jednotlivých odděleních, díky čemuž může předejít například odchodům zaměstnanců, či výkyvům ve výkonnosti. Kromě toho je platforma schopna i navrhnout na jaké pozice se jednotliví pracovníci nejvíce hodí a které jim budou lépe vyhovovat v závislosti na jejich vlastnostech.

Platforma jako taková se skládá z několika produktů, které lze libovolně kombinovat dle potřeb konkrétního zákazníka. Díky tomu si může firma zakoupit pouze produkty a jejich funkce, o kterých ví, že je využije, např. hodnocení vlastností zaměstnanců, kontrolování plnění stanovených cílů ap., namísto toho, aby „zbytečně vynakládala finanční prostředky “ za funkcionalitu, o které ví, že ji nevyužije.

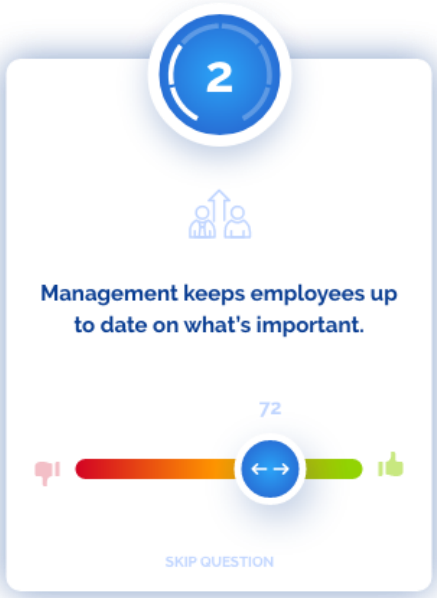
## 3.7 Popis platformy LutherOne

### 3.7.1 Charakteristika produktu Friday 6

Název tohoto produktu je odvozen od jeho funkcionality. Každý pátek se vygeneruje pro jednotlivé zaměstnance dotazník obsahující 6 nahodilých otázek.

Cílem těchto pravidelných dotazování je získání poznatků, kterými jsou např. firemní atmosféra, spokojenost zaměstnanců, různé indexy, skóre, priority, příčiny, včasná varování, upozornění a další hodnoty v určitém časovém horizontu.

Díky těmto dotazníkům mohou manažeři monitorovat spokojenost podřízených a celkovou náladu na pracovišti, včas na to reagovat, aby například zabránili případnému odchodu zaměstnanců.




Obrázek 4 Friday 6 [23]

### 3.7.2 Charakteristika produktu My spectrum and bubbles

My spectrum je produkt, který slouží ke sběru hodnocení týkající se zaměstnanců, jejich výkonu, talentu i rozvoje.

Tento produkt v sobě soustřeďuje výhody distribuované důvěry a poznatků z psychometrie a psychodiagnostiky k vytváření unikátních profilů zaměstnanců, které dokonale vystihují jejich jedinečnou profesní identitu.

Zkombinováním velkého počtu vstupů, získaných v průběhu času od velkého počtu lidí, nám LutherOne pomáhá porozumět každému jednotlivci a převzít tak odpovědnost za jeho profesní rozvoj, a zároveň mu umožnit, aby i on ovlivnil své pracovní prostředí.

Tyto hodnoty se posléze promítnou do tzv. kytky, kde převádí odpovědi od jednotlivých uživatelů do formy souhrnného grafu.


Obrázek 5 Spectrum – kytka [22]

Bubbles, stejně jako Spectrum slouží k hodnocení vlastností jednotlivých zaměstnanců. Hlavním rozdílem mezi těmito produkty je znázornění získaných hodnot. Ty jsou vyobrazené ve formě tzv. bublin.


Obrázek 6 Bubbles [22]

### 3.7.3 Charakteristika produktu Celebr8

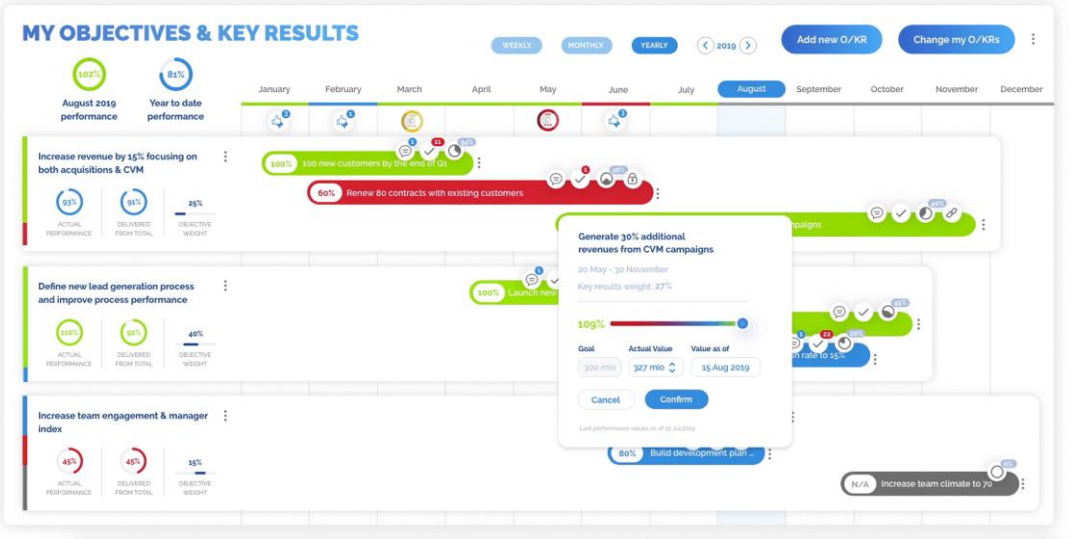
Tento produkt slouží ke správě odměn, které mohou zaměstnanci firmy získat za svůj výkon. Tato ocenění jsou trojího typu, odznaky, ocenění a certifikáty.


Obrázek 7 Certifikát [19], Ocenění [20], Odznak [21]

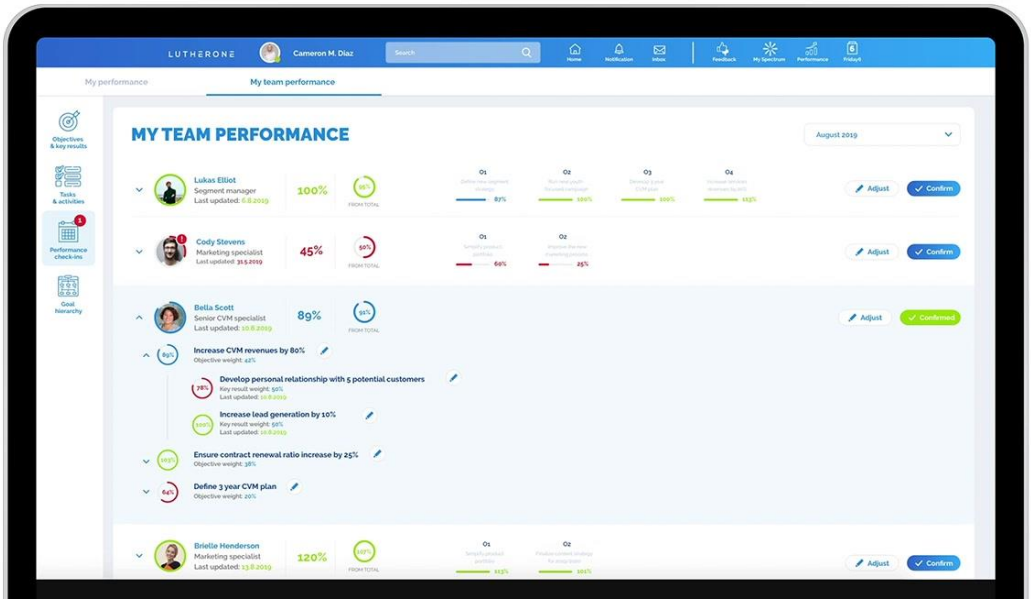
### 3.7.4 Charakteristika produktu Performance management and tasks

Jak již název napovídá, tento produkt slouží k řízení výkonu zaměstnanců. Jeho úkolem je pomoci firmě ve sledování stanovených cílů a správu jednotlivých úkolů.

Pomocí tohoto produktu může uživatel spravovat a kontrolovat plnění cílů v reálném čase. Stejně tak může i kontrolovat jakých výsledků dosahují jak jednotlivci, tak i celé týmy. Díky tomu uživatel snadno zjistí, zdali plnění cílů pokračuje dle stanoveného harmonogramu či jaký přístup k práci mají jednotlivý zaměstnanci.

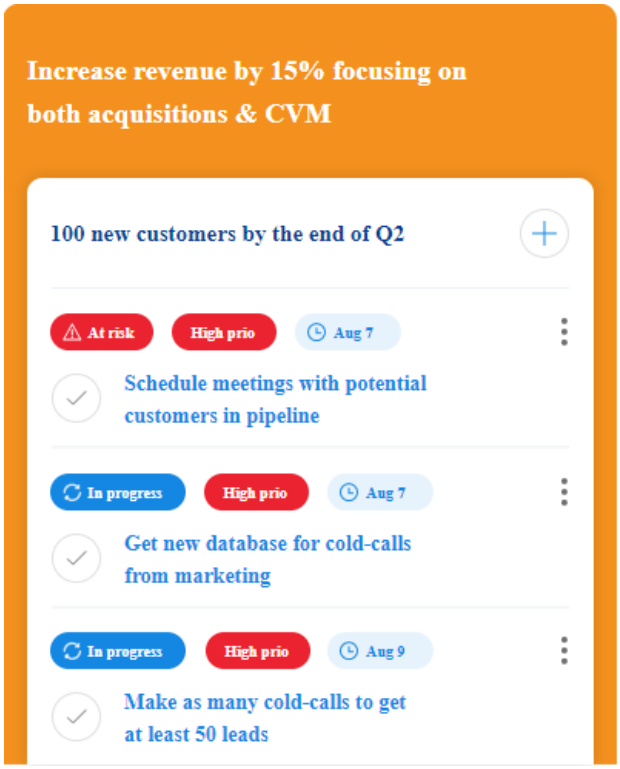



Obrázek 8 PerformanceManagement [15]




Obrázek 9 Team performance management [16]

Součástí tohoto produktu je i správa úkolů. Pomocí tohoto nástroje si je firma může roztřídit dle různých kritérií, kterými jsou například důležitost, priorita atd. Posléze může uživatel jednotlivé úkoly přiřadit konkrétním pracovníkům a monitorovat jejich úsilí a úspěšnost při jejich řešení.




Obrázek 10 PerformanceManagement [17]

### 3.7.5 Charakteristika produktu Feeds

Dalším z produktů platformy LutherOne je Feeds. Tento produkt si klade za cíl usnadnit a urychlit spolupráci a komunikaci mezi jednotlivými zaměstnanci. Díky tomu se zefektivní práce a zároveň vše bude probíhat v uzavřeném a bezpečném prostředí, kde nehrozí žádný únik informací.

Kromě toho tento produkt dále nabízí i možnost eliminovat nepotřebné schůzky či zbytečné e-maily.


Obrázek 11 Mobilní verze produktu Feeds [28] [29]

### 3.7.6 Charakteristika produktu Feedback

Jak již název napovídá, tento produkt slouží k získání několika typů zpětných uživatelských vazeb. Jejich pomocí lze udělit hodnocení jednotlivcům, týmům či si např. ověřit zpětnou vazbu na proběhlou schůzku. V případě, že má uživatel zájem o zpětné odezvy na konkrétní událost, může vytvořit i svůj vlastní dotazník, který by mu měl napomoci k získání potřebných informací.


Obrázek 12 Feedback+ [24]

## 3.8 Stávající situace IS/IT

Pro běh platformy společnost využívá infrastrukturu, postavenou na produktech od společnosti Amazon, tj. Amazon Web Services (AWS). Společnost se rozhodla pro užití těchto produktů z důvodu jejich předpřipravených komplexních řešení. Dalším důvodem pro zvolení byla jejich podpora služeb postavených na architektuře mikroslužeb, což je případ i platformy LutherOne.

Platforma jako aplikace je tvořena několika desítkami mikroslužeb, které jsou primárně napsány ve skriptovacím jazyce Typescript, za pomoci framework Node.js, ale některé funkce byly postupně napsány i v jazyce Python. K ukládání dat se využívají dva typy databáze, PostgreSQL a MongoDB a pro ukládání médií je využito cloudové úložiště zakomponované v AWS, tj. S3, neboli Simple Storage Service.

Jak již bylo zmíněno, jedná se o začínající firmu, která jako většina jiných IT start-upů vyvíjejících vlastní produkty v prvé řadě řešila vybudování stabilní infrastruktury a následně samotný vývoj nabízené aplikace. V současné době se firma dostala do fáze, kdy je na místě řešit automatizaci procesu aktivace nového zákazníka.

## 3.9 Proces aktivace nového zákazníka

Platforma je provozována na sdílené nebo dedikované infrastruktuře, záleží na velikosti a preferencích zákazníka. Pokud požaduje provoz platformy v dedikované infrastruktuře, potom samotné konfiguraci zákazníka předchází krok přípravy infrastruktury:

- vytvoření virtual private cloud
- konfigurace network security
- konfigurace load balancing
- konfigurace CDN
- nastavení přístupu pro S3 storage
- alokace EC2 virtuálních serverů
- instalace a konfigurace unmanaged services (MongoDB, RabbitMQ)
- alokace a konfigurace managed services (PostreSQL)

Pro takto připravenou infrastrukturu je zapotřebí následně provést instalaci platformy. Tento krok se skládá z řady komplexních úkolů, instalací mikroslužeb, jejich konfigurací a vzájemného propojení.

Následující kroky jsou již společné jak pro provoz ve sdílené, tak i dedikované infrastruktuře.

Dalším krokem přípravy platformy je její konfigurace. Tento krok vyžaduje konfiguraci zákazníka jako tenanta a dále systémových administrátorů, jejich přístupů a aktivaci zvolených produktů.

Posledním krokem pro spuštění platformy a aktivaci zákazníka je nahrání všech jeho organizačních struktur (důležité především pro komplexní holdingové struktury a globální společnosti).

## 3.10 Aktuální stav řízení procesu aktivace nového zákazníka

Postup aktivace nového zákazníka je značně komplexní proces skládající se z řady kroků. Tento proces jako celek je složen z mnoha podprocesů, které jsou již zautomatizované či je jejich realizace poloautomatická.

### 3.10.1 Vytvoření infrastruktury

Samotná příprava produkčního prostředí na aktivaci nového zákazníka je poměrně komplexní. Jedná se o sérii kroků, která si klade za cíl nastavit veškeré platformou LutherOne vyžadované produkty od firmy Amazon (AWS).

V prvé řadě musí odborný pracovník posoudit nakolik je stávající produkční prostředí vytížené. V případě, že je již přetížené, musí být vytvořeno další prostředí, aby se mohl zaktivovat další zákazník. Vytvoření nového prostředí je již zautomatizované pomocí produktu AWS CloudFormation, který je součástí platformy AWS. Jedná se o soubor napsaný ve formátu JSON či YAML, který slouží jako šablona definující vše co je nutné nastavit v rámci AWS platformy.

### 3.10.2 Instalace platformy

Po nastavení prostředí započne fáze instalace jednotlivých mikroslužeb. Tento krok je již zautomatizován za pomoci skriptů napsaných v jazyce Python. Tento skript vybere požadované mikroslužby, které jsou již nainstalovány na před produkčním prostředí a nainstaluje je na nově vzniklé produkční prostředí.

Tato operace je řízena pomocí dvou vstupních parametrů určujících které mikroslužby chce uživatel nainstalovat a názvem jejich cílového prostředí. Verze mikroslužeb, které se instalují v novém produkční prostředí, jsou vybírány z před produkčního prostředí pomocí nainstalovaných stabilních mikroslužeb, tzn. verze, které by již neměly obsahovat žádné vážnější chyby.

Po jejich úspěšném nainstalování ještě musí odborný pracovník manuálně provést u každé ze služeb konfiguraci. V rámci tohoto kroku se jim alokují potřebné zdroje, kterými jsou především výpočetní výkon (CPU) a paměť. Toto individuální nastavení se provádí především z důvodu úspory financí, jelikož z používání služeb platformy AWS se odvádí poplatky. Z tohoto důvodu není vhodné alokovat nadměrné množství zdrojů, které mikroslužba pro svou činnost nepotřebuje. Kromě alokace zdrojů musí odborný pracovník nastavit každé z mikroslužeb správné hodnoty enviromentálních proměnných. Tyto proměnné obsahují hodnoty vztahující se převážně ke konkrétnímu prostředí, tj. např. url adresa ostatních mikroslužeb, pro zákazníka vyhraněná lokalita cloudového úložiště určené k nahrávání jeho osobních souborů (obrázky, videa, dokumenty…) atd.

### 3.10.3 Aktivace platformy

Po dokončení instalace započne fáze aktivace, zákazníkem zakoupených produktů. Odborný pracovník musí manuálně zaktivovat každou z mikroslužeb obhospodařujících zákazníkem zakoupené produkty. Tato aktivace se převážně skládá z manuálního provolání několika interních endpointů vystavených na REST API a nahrání zákaznických dat do platformy.

K fungování tohoto systému je nutné, aby klient dodal soubory obsahující potřebné údaje týkající se jeho společnosti, tj. základní údaje o firmě, struktura společnosti a organizační struktura. Před nahráním těchto dat do systému musí odborný pracovník zkontrolovat, zdali jsou dodané soubory validně vyplněné. Tato činnost se provádí za pomoci skriptů napsaných v jazyce Python. U souboru, který se týká základních informací o firmě je validace poměrně snadná, jelikož obsahuje pouze několik hodnot, mezi které patří např. název společnosti, IČ, DIČ atd.

Zbývající dva soubory obsahují o poznání více informací, z tohoto důvodu je zde validace mnohonásobně složitější. Soubor zabývající se informacemi o struktuře firmy musí obsahovat data o všech jednotlivých pobočkách daného podniku.

Zbylá zákazníkem dodávána data jsou umístěna v souboru týkajícím se organizační struktury podniku. Vzhledem k tomu, že jsou v něm umístěny veškeré informace o každém ze zaměstnanců, včetně jejich umístění v hierarchii podniku, jedná se o velmi složitý soubor.

V případě, že kontrola všech zákazníkem dodaných souborů proběhne v pořádku, započne odborný pracovník proces, nahrávání získaných dat do databáze. K tomuto slouží další z interních endpointů vystavených na REST API.

## 3.11 Cílový stav řízení procesu aktivace nového zákazníka

Jak je již z předchozí kapitoly patrné, postup aktivace nového zákazníka je komplexní proces složený z mnoha kroků. Každý z nich je samostatným podprocesem, který je již dnes plně zautomatizován, nebo je jeho realizace poloautomatická. Celý proces aktivace je ale řízen manuálně. Cílem je plná automatizace, jak jednotlivých kroků, tak i celého procesu.

V rámci tohoto procesu je tedy potřeba zvolit takové technologie, aby odpovídaly potřebám firmy a následně byly vhodné i pro samotný úkon aktivace zákazníka. Jelikož se dá předpokládat, že firma bude v budoucnu automatizovat více podnikových procesů, bylo by vhodné vytvořit novou interní mikroslužbu, která bude veškerou automatizaci společnosti spravovat.

Díky zautomatizování procesu aktivace nového zákazníka firma očekává, že dojde k jejímu zefektivnění a zároveň i urychlení dokončení tohoto procesu.

## 3.12 Výběr technologií

V prvotní fázi automatizace je nutné vhodně zvolit jednotlivé technologie, které se následně při vývoji jednotlivých částí této mikroslužby užijí.

### 3.12.1 Výběr automatizačního softwaru

Na trhu existuje celá řada softwarových nástrojů, které se na automatizaci firemních procesů specializují. V návaznosti na předchozí osobní zkušenosti se autor rozhodoval mezi třemi různými řešeními, a to TIBCO BusinessWorks verzi 5.8 společně s Tibco iProcess, Camunda BPMN a IBM Business Process Manager. Všechny tyto technologie jsou zdánlivě pro řešení dané problematiky vhodné.

Nástroj, který nabízí firma IBM je poměrně jednoduchý a vývoj jednotlivých procesů v něm je poměrně rychlý, ale přesto není pro potřeby této firmy nejvhodnější. Jedná se o komplexní platformu obsahující velké množství funkcionalit. Ovšem v tomto případě by většina z těchto funkcí nebyla využita. Cílovou skupinou zákazníků pro využití této platformy jsou primárně velké, mnohdy až nadnárodní podniky, které nabízené funkcionality využijí k řešení svých mnohonásobně komplexnějších firemních procesů.

Dalším problémem je i cena takovéto licence. Vzhledem k tomu, že tento software nabízí velkou škálu funkcionalit, odpovídá tomu i cena licencí na jejich využití, která rozhodně není přijatelná pro začínající firmu.

Podobné nevýhody jsou i v případě nástrojů, které nabízí společnost Tibco Software. I v tomto případě se jedná o softwarové řešení určené spíše pro velké firmy, ovšem v tomto případě je zde i další nevýhoda a tou je stáří této vývojářské platformy.

Z těchto důvodů se firma nakonec rozhodla pro užití nástroje Camunda BPMN, který sice nenabízí tolik funkcí jako předešlé a ani nemá takovou klientskou podporu, ale zato poskytuje mnoho funkcí zcela zdarma. Další z výhod tohoto IDE je i možnost vytváření vlastních skriptů napsaných v několika různých programovacích jazycích. Jedním z těchto jazyků je i Javascript, což velmi usnadní práci při manipulaci s daty, které se získávají či zasílají za využití rozhraní REST API. Vše potřebné k dosažení stanovených cílů lze v rámci tohoto nástroje využít bez jakýchkoliv poplatků.

### 3.12.2 Výběr technologie pro vývoj logické vrstvy

V závislosti na vybraném nástroji pro automatizaci firemních procesů připadají v úvahu dva jazyky, pomocí kterých by šlo požadovanou mikro službu vytvořit. Autor se rozhodoval mezi užitím programovacího jazyka Java, či skriptovacího jazyka Javascript, konkrétněji framework Node .js.

Obě tyto technologie by bylo možné k realizaci mikroslužby použít. V případě Node .js je možnost doinstalovat do projektu za využití NPM (Node .js package manager) balíčky, pomocí kterých by se dala propojit logická vrstva s procesy navrženými v programu Camunda BPMN. Ovšem po prostudování dokumentace se zdá být jako vhodnější programovací Jazyk Java za použití Spring Boot frameworku, jelikož tvůrci zvoleného automatizačního softwaru uvádějí přímo v rámci dokumentace podrobný návod, jak propojit Camundu se Spring Boot frameworkem.

### 3.12.3 Výběr technologie pro vývoj databázové vrstvy

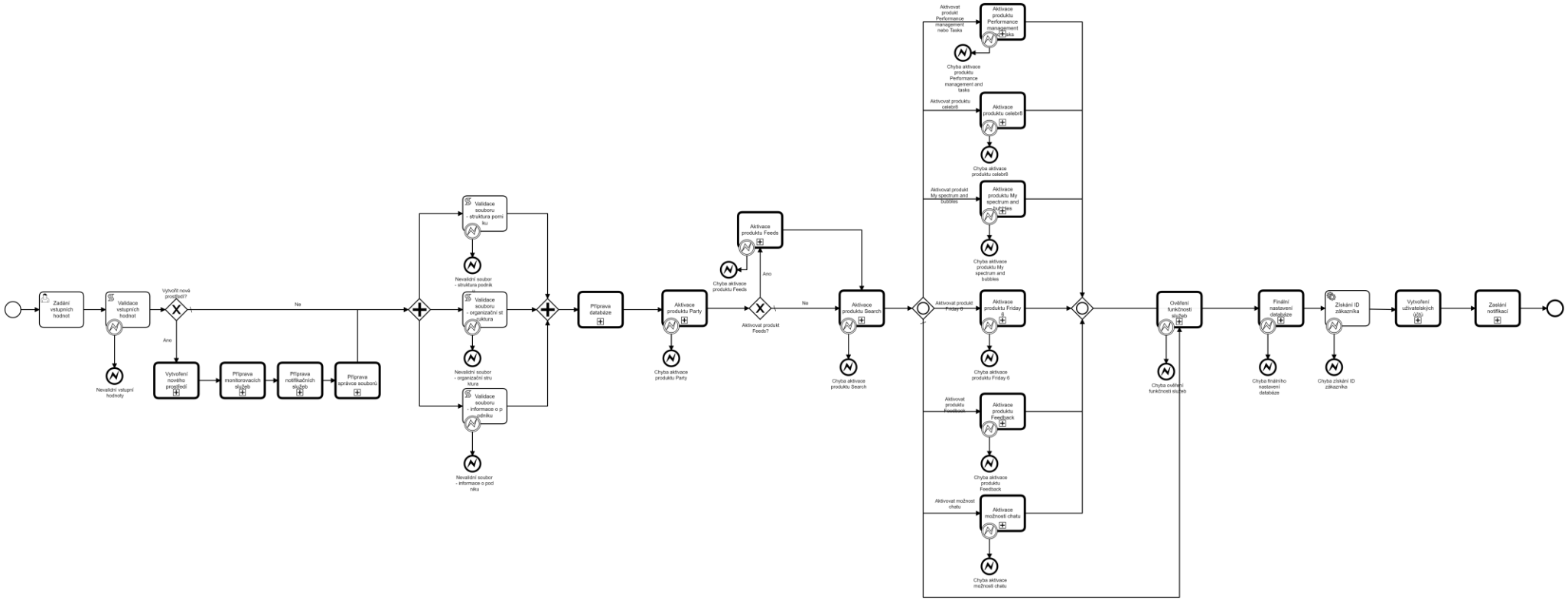
Pro uložení navržených procesů v prostředí Camunda bylo také třeba vybrat vhodné technologie. Databáze NoSQL typu MongoDB aj. se k tomuto účelu nehodí, a proto bylo potřeba vybrat pouze jeden konkrétní typ SQL databázového řešení. Tento výběr byl poměrně snadný, jelikož v rámci tohoto podniku je již zavedena PostgreSQL databáze. Nemělo by tedy smysl zavádět další typ SQL databázového řešení, ale použít již zavedené technologie i do budoucna.

## 3.13 Návrh procesu aktivace nového zákazníka

Vzhledem ke komplexnosti procesu aktivace nového zákazníka se autor rozhodl využít možnosti, rozdělit je na několik dílčích podprocesů, které budou mít za cíl nakonfigurovat a zaktivovat jednotlivé části nabízené platformy.

### 3.13.1 Hlavní proces

Návrh hlavního procesu se skládá z několika podprocesů, validačních skriptů a jednoho uživatelského úkolu.



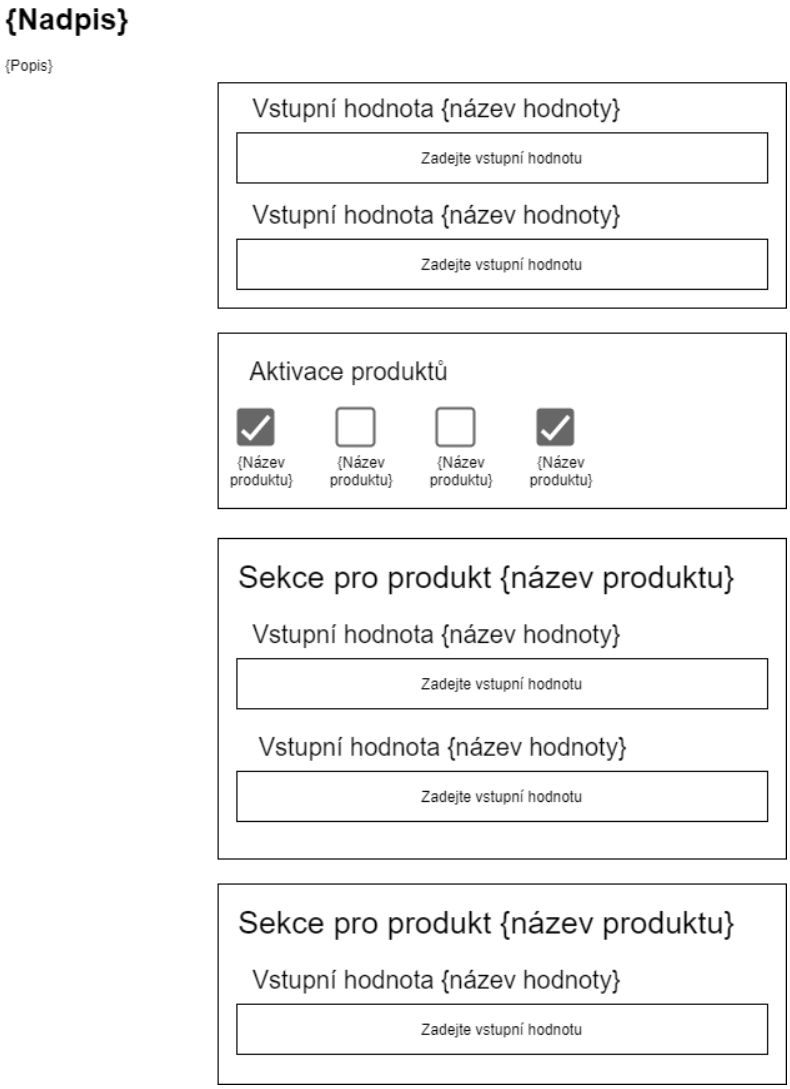

Obrázek 13 Hlavní proces

#### 3.13.1.1 Nastavení vstupních hodnot

Celý proces aktivace nového zákazníka začíná uživatelským úkolem. V rámci tohoto postupu se uživateli zobrazí v prohlížeči, po zavolání konkrétního REST API endpointu, který se bude nacházet na nově vzniklé mikroslužbě, formulář, jenž bude obsahovat několik sekcí, které je třeba vyplnit. Tento formulář bude obsahovat sekci pro povinné hodnoty, které bude třeba vždy zadat. Poté následuje sekce, kdy uživatel vybere pomocí checkBox prvků jednotlivé produkty, které chce aktivovat.

V závislosti na jeho výběru se mu zobrazí sekce, které budou obsahovat povinné vstupní hodnoty, které jsou potřebné pro aktivaci daného produktu.

Uživatelem vyplněné údaje následně poslouží jako vstupní hodnoty pro jednotlivé skripty / REST API endpointy v následujících krocích procesu.




Obrázek 14 Návrh uživatelského formuláře

#### 3.13.1.2 Validace vstupních hodnot

Jedná se o úkol typu skript. V rámci tohoto kroku dojde ke kontrole, zdali uživatelem zadaná data jsou validní. Může se jednat i o kontrolu, zdali uživatel nezadal do formuláře chybné údaje, např. nesprávný datum, prázdný textový řetězec ap.

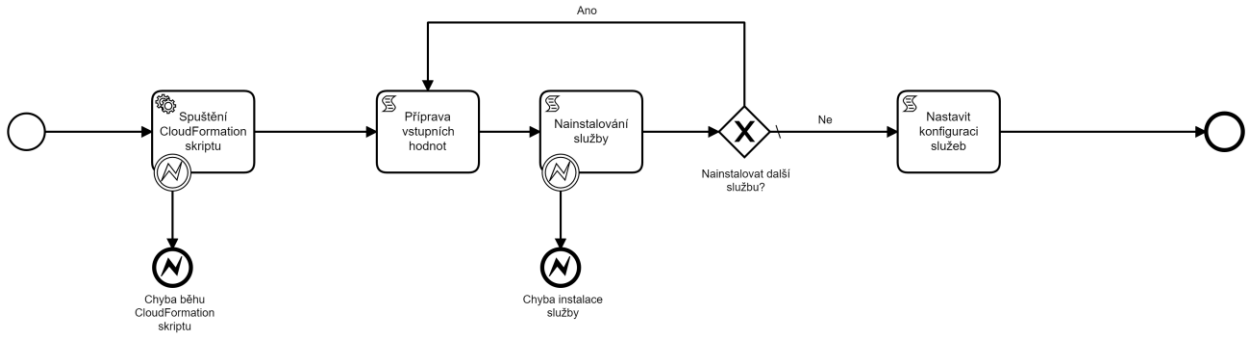
Pokud uživatel zadal jakoukoliv nesprávnou hodnou, celý proces předčasně skončí v chybovém stavu.

#### 3.13.1.3 Vytváření nového prostředí

Pokud je uživatelem vyplněný formulář validní, následuje exkluzivní typ brány, kde se na základě uživatelských hodnot rozhoduje, zdali se vytvoří nové produkční prostředí, či aktivace nového zákazníka proběhne na již existujícím prostředí.

V případě, že dojde ke splnění stanovené podmínky, pokračuje datový tok do procesu sloužícího k předpřipravení prostředí k aktivaci produktů. Pokud nedojde ke splnění podmínky, pokračuje datový tok do paralelní brány a následné validaci souborů.

##### 3.13.1.3.1 Vytvoření nového prostředí




Obrázek 15 Vytvoření nového prostředí

Tento proces slouží k vytvoření nového produkčního prostředí a nainstalování jednotlivých mikroslužeb.

Prvním úkolem v tomto procesu je spuštění CloudFormation skriptu, který nastaví veškeré produkty AWS platformy, které jsou potřeba pro validní chod.

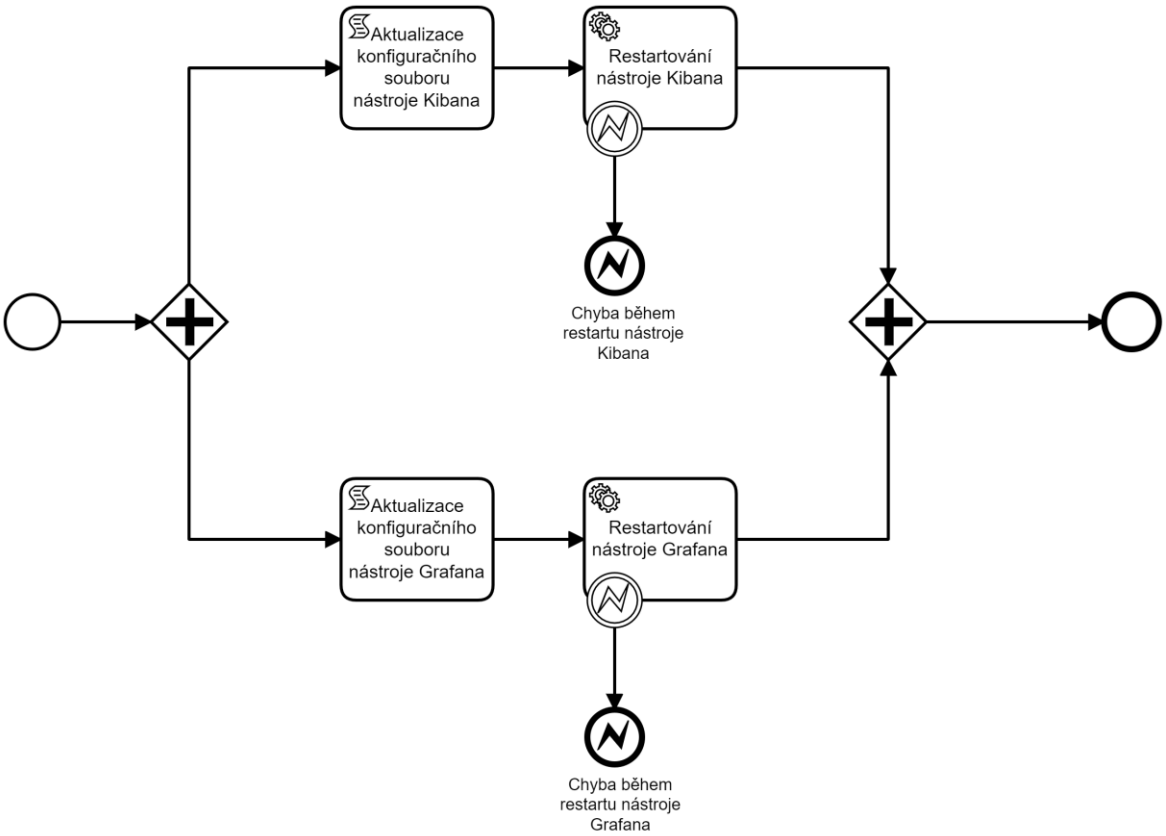
Jakmile je prostředí připravené k užívání, započne fáze instalace jednotlivých mikroslužeb. Tato fáze začíná úkonem Příprava vstupních hodnot, který je typu skript. V tomto úkolu se předpřipraví vstupní hodnoty, které se v následujícím kroku použijí pro instalaci mikroslužby.

Po instalačním skriptu následuje exkluzivní brána ověřující zdali již byly nainstalovány všechny mikroslužby. V případě, že některé ještě nebyly, vrací se datový tok

zpět do úkolu Příprava vstupních hodnot, kde se předpřipraví potřebné vstupní hodnoty pro instalaci další mikroslužby.

Po dokončení instalace následuje úkol Nastavit konfiguraci služeb, který je typu skript. Tento krok slouží k nastavení nainstalovaných mikroslužeb. Jedná se především o přiřazení zdrojů (CPU, paměť) a enviromentálních proměnných.

3.13.1.3.2 Příprava monitorovacích služeb



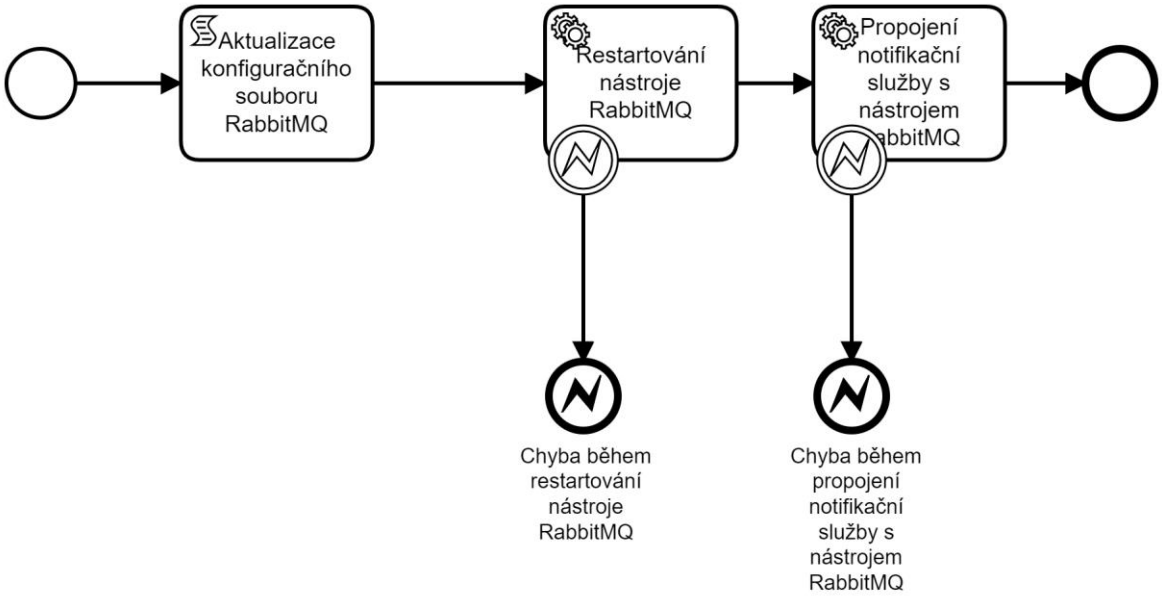

Obrázek 16 Příprava monitorovacích služeb

Aby odborní pracovníci mohli kontrolovat stav nabízené platformy, potřebují externí nástroje, jejichž pomocí mohou stav jednotlivých mikroslužeb monitorovat a v případě vyskytnutí se chybového stavu jim být nápomocny, proč k pochybení došlo. To je hlavním důvodem, proč je nutné neustálé připojení k těmto monitorovacím nástrojům.

Na začátku tohoto procesu je umístěna paralelní brána, která rozděluje vstupní datový tok do dvou větví. V obou je jako první úkon typu skript, kde se aktualizuje konfigurační soubor daného nástroje.

Následně dojde k restartování dané služby, díky čemuž začne používat upravenou verzi svého konfiguračního souboru.

3.13.1.3.3 Příprava notifikačních služeb




Obrázek 17 Příprava notifikačních služeb

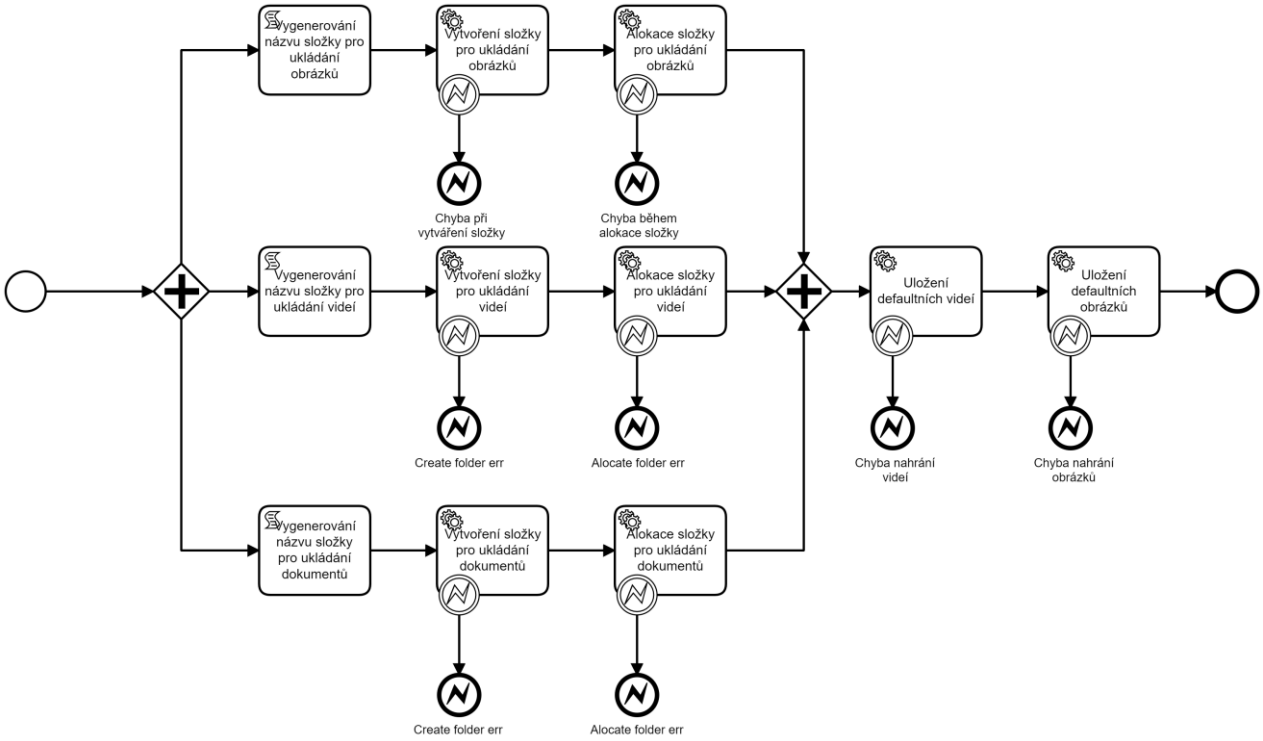
V rámci tohoto procesu se nakonfiguruje logika sloužící k nastavení a zasílání jakýchkoliv notifikací.

V rámci tohoto procesu se aktualizuje konfigurační soubor sloužící k nastavení externí služby zajišťující zprostředkování zasílání zpráv.

Následně dojde k restartu externího nástroje RabbitMQ, aby začal používat novou verzi svého konfiguračního souboru.

Poté již zbývá jen propojit mikroslužbu spravující jednotlivé typy notifikací s tímto nástrojem.

3.13.1.3.4 Příprava správce souborů




Obrázek 18 Příprava správce souborů

Na počátku procesu přípravy správce souborů se nachází paralelní brána rozdělující datový tok do tří. Tyto tři toky odpovídají rozdělení souborů na tři typy, tj. obrázky, videa a dokumenty.

V každé z těchto větví se nachází tři úkoly. První úkol je typu skript a slouží k vygenerování unikátního názvu složky, do kterého se budou příslušné soubory ukládat. V následujícím kroku se tento vygenerovaný název použije k vytvoření nové složky na tzv. cloudovém úložišti, kde posléze dojde v rámci databázového úložiště k její alokaci.

Po vytvoření složek pro jednotlivé typy souborů dojde nejdříve k nahrání defaultních videí, a následně i obrázků.

Pokud by během jednotlivých operací došlo k omylu, celý proces předčasně skončí v chybovém stavu.

3.13.1.4 Validace souborů

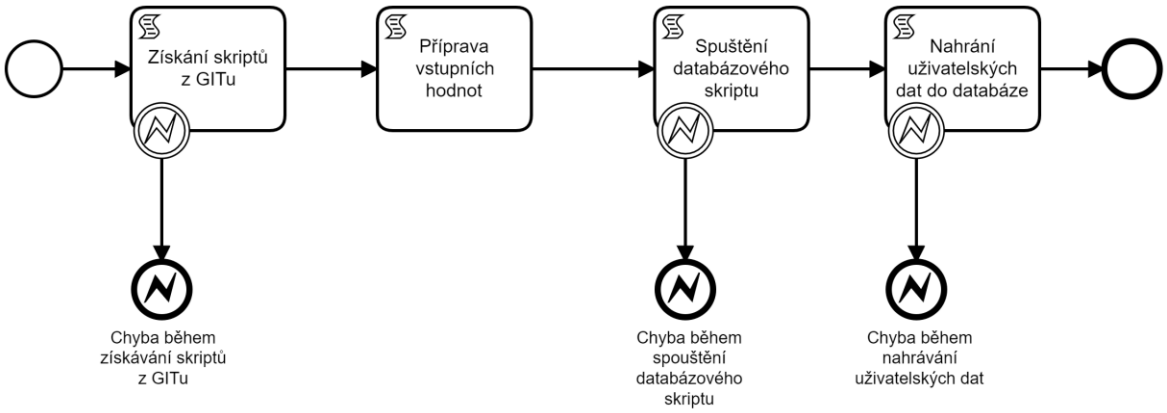
Pokud uživatelem vyplněný formulář je validní, následuje paralelní typ brány, která dočasně rozděluje dosavadní jeden tok na tři, jejichž pomocí se zároveň spustí tři úkoly, typu skript, jenž slouží k validaci zákazníkem dodaných souborů.

Pokud by některý z těchto tří skriptů odhalil jakoukoliv nepřesnost v dodaném souboru, došlo by k předčasnému ukončení procesu.

Tímto krokem se zabrání nahrání nevalidních dat do databáze. Jestliže všechny tři soubory budou správně vyplněné, sjednotí se jednotlivé toky zpět do jednoho.

### 3.13.2 Příprava databáze

Příprava databáze je název procesu, který se dále skládá z několika úkolů typu skript. Hlavním cílem tohoto procesu je kompletní nastavení databáze a následné nahrání zákazníkem dodaných dat.




Obrázek 19 Příprava databáze

Pomocí skriptu umístěného na počátku procesu se získá nejaktuálnější verze souborů uložených na GIT serveru. Tyto soubory obsahují skripty potřebné k nastavení databáze pro nového uživatele.

Následný krok slouží pouze k předpřipravení hodnot, které uživatel v prvotním kroku zadal do formuláře a které poslouží jako vstupní hodnoty v následujícím úkolu.

Jelikož se zde užívají pouze hodnoty, které již byly zkontrolovány v předchozích krocích, které jsou umístěny v hlavním procesu, neobsahuje tento krok průběžnou událost zachycující chybový výstup ze skriptu.

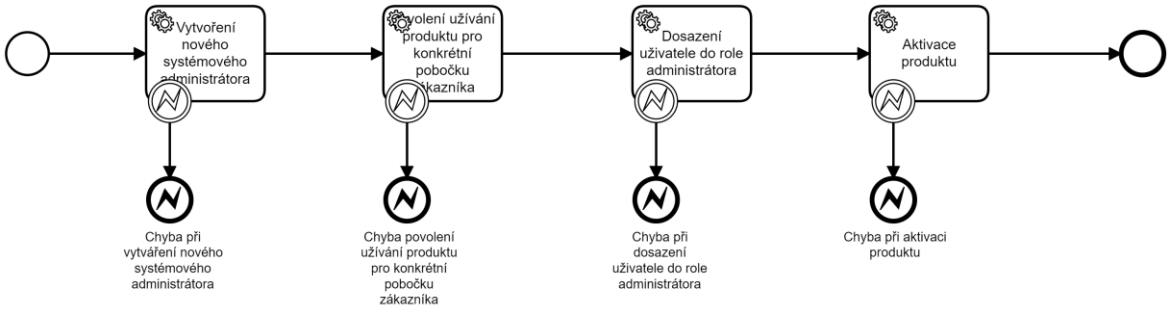
Jakmile jsou vstupní hodnoty připraveny k použití, spustí se databázový skript získaný v první aktivitě tohoto procesu.

Následuje poslední z úkolů v rámci procesu Přípravy databáze. V této fázi by již měla být veškerá nastavení databáze dokončená, a tudíž by měla být připravená na nahrání dat, které zákazník zaslal v jednotlivých souborech.

V případě, že by došlo během přípravy databáze k jakékoliv chybě, celý proces aktivace nového zákazníka by předčasně skončil v chybovém stavu.

### 3.13.3 Aktivace produktu Party

Jakmile jsou data o podniku nahraná v databázi, započne nastavování prvního z produktů „Party“, který je důležitý pro zpracování základních dat. Tato služba má na starosti nejen správu uživatelských profilů, ale například i jednotlivé týmy, oddělení, pobočky atd. Z toho důvodu musí být tato služba vždy aktivovaná.



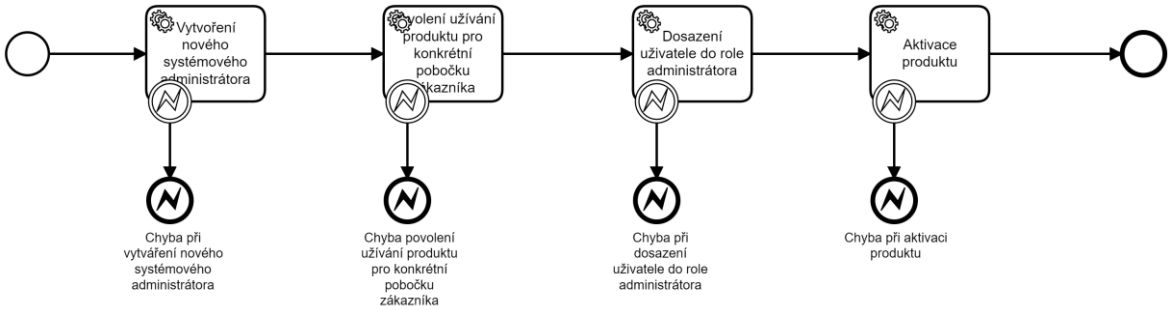

Obrázek 20 Aktivace produktu Party

Proces se skládá z několika kroků:

- Vytvoření nového systémového administrátora – Vytvoření systémového administrátora a přiřazení role a s ní spjaté oprávnění ke správě tohoto produktu.
- Povolení užívání produktu pro konkrétní pobočku zákazníka – Nastavení povolení používání produktu Party pro konkrétní společnost daného zákazníka.
- Dosazení uživatele do role administrátora – Spustí se logika, která slouží k dosazení konkrétního uživatele do role administrátora dané pobočky, pro kterou se tento produkt nastavuje.
- Aktivace produktu – V rámci tohoto kroku by mělo dojít k dokončení nastavení produktu pro konkrétní pobočku.

Pokud by nastal jakýkoliv problém, předčasně se celý proces aktivace nového zákazníka ukončí.

### 3.13.4 Aktivace produktu Feeds




Obrázek 21 Aktivace produktu Feeds

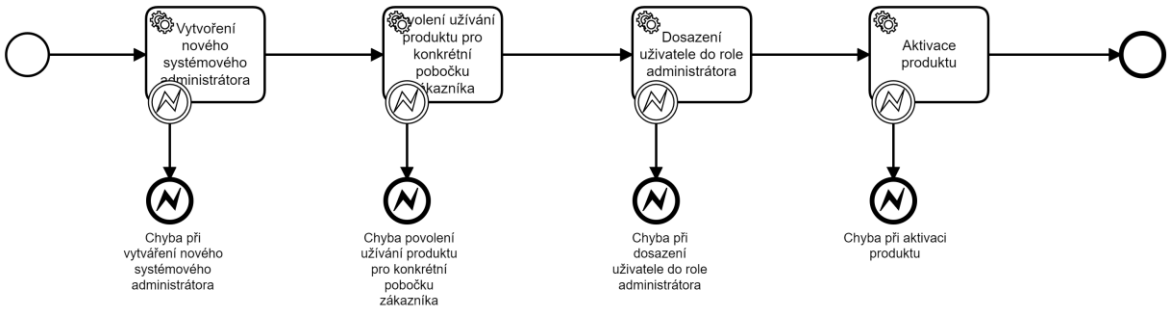
V případě, že uživatel vybral Feeds produkt, současně započne i jeho aktivace. Produkt Feeds musí být aktivovaný vzápětí po Party, jelikož následující mikroslužby od této přebírají určitá data, pokud jsou k dispozici.

Aktivace Feeds produktu bude probíhat téměř stejným způsobem, jako je tomu u aktivace Party produktu.

Nejdříve se přiřadí nová role Systémovému administrátorovi a následně se povolí užívání tohoto produktu pro konkrétní pobočku. Poté se k ní přiřadí určený administrátor a na závěr se celý produkt zaktivuje.

Hlavním rozdílem mezi Feeds produktu a Party je, že aktivační proces volá endpointy, které poskytuje jiná, tomu určena mikroslužba.

### 3.13.5 Aktivace produktu Search




Obrázek 22 Aktivace produktu Search

Jakmile je Feeds produkt připravený k užívání, nebo pokud případně nebyl vybrán, nastává čas na aktivaci dalšího produktu.

Jak již název napovídá, jedná se o produkt, který slouží k vyhledávání. Užívá se prakticky v celé aplikaci, kde je uživatelům nabídnuto libovolné a neomezené vyhledávání. Díky němu si tedy uživatel může dohledat konkrétní příspěvek, dle jeho popisku, profil jiného uživatele, jiný tým uživatelů ap.

Jako v případě Party produktu se jedná o službu, která se zákazníkovi aktivuje automaticky.

Stejně jako u předchozích dvou produktů, i zde probíhá postup aktivace totožně.

Nejdříve se přiřadí nová role Systémovému administrátorovi a následně se povolí užívání tohoto produktu pro konkrétní pobočku. Poté se k ní přiřadí určený administrátor a na závěr se celý produkt zaktivuje.

I v tomto případě je prakticky jediným rozdílem umístění REST API endpointů, které se vyskytují na Search mikroslužbě.

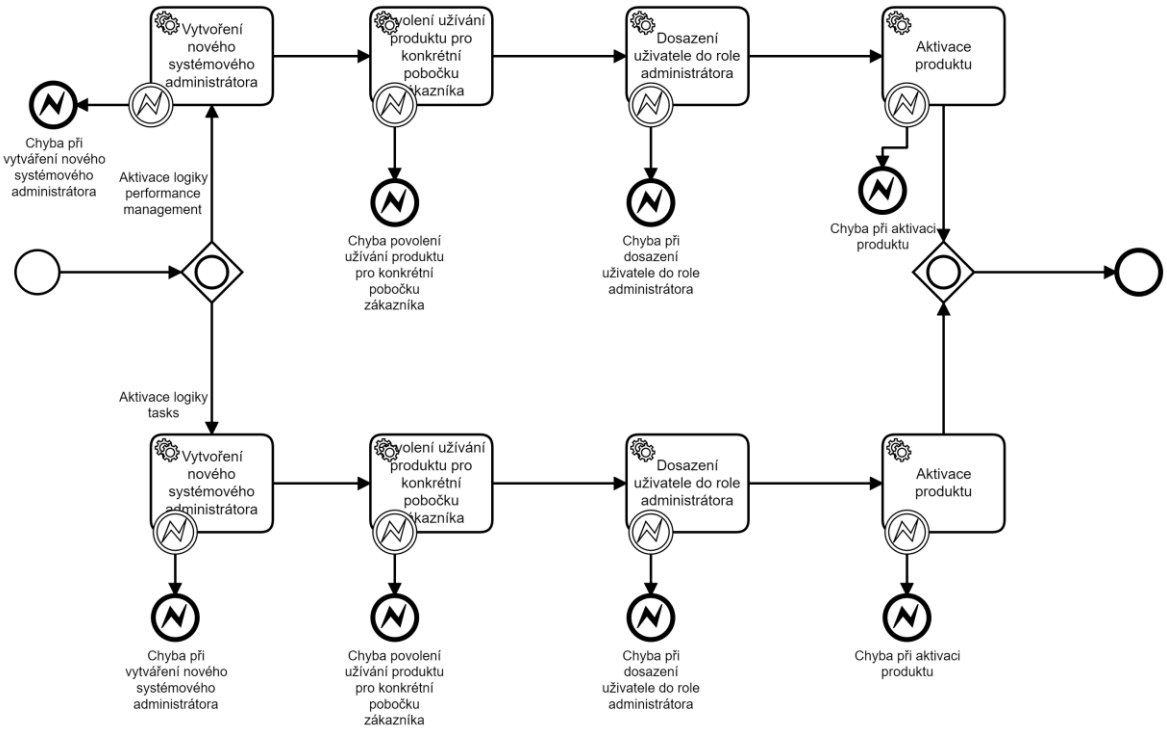
### 3.13.6 Aktivace ostatních produktů

Jakmile se dokončí proces Aktivace produktu Search, dospěje datový tok do bodu, kdy by mělo začít hromadné aktivování jednotlivých produktů. Toto aktivování je řešeno pomocí inkluzivní brány, ze které vede datový tok k jednotlivým podprocesům sloužícím k zaktivování jednotlivých produktů. Aby daný datový tok byl využit, musí být splněna podmínka, která ověřuje, zdali byl uživatelem právě tento produkt k aktivaci zvolen.

Kromě těchto toků obsahuje inkluzivní brána i jeden defaultní tok pro případ, že by uživatel již nechtěl žádný další produkt zaktivovat. Pomocí tohoto defaultního toku by se přeskočila fáze hromadného aktivování produktů a uživatel by se tak přímo dostal k procesu Ověření funkčnosti služeb.

#### 3.13.6.1 Aktivace produktu Performance and tasks

Pomocí inkluzivní brány se zavolá tento proces v případě, že se uživatel rozhodl aktivovat produkt Performance management and tasks.



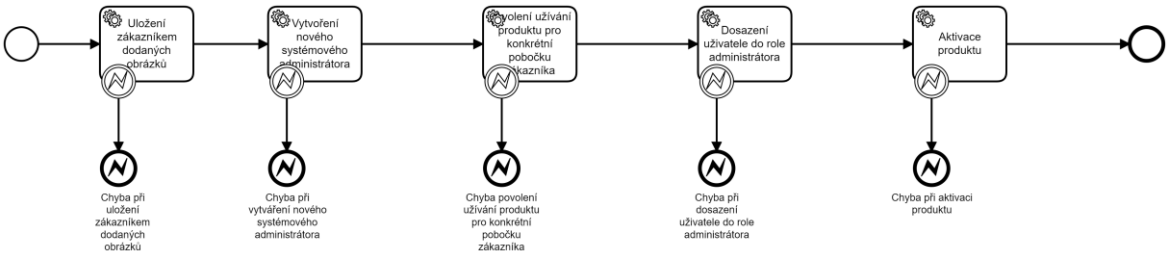

Obrázek 23 Aktivace produktu Performance management and tasks

I když to není příliš patrné, i v tomto případě je jedná prakticky o stejný postup aktivace produktu, který je již popsán v předešlých kapitolách.

Hlavním rozdílem je, umístění inkluzivní brány na začátek tohoto procesu. Pomocí této brány lze zaktivovat i pouze část z nabízené funkcionality tohoto produktu. Zákazník tak může po dokončení aktivace využívat pouze performance management či správu úkolů.

Vstup do tohoto procesu je podmíněn zvolením alespoň jedné z funkcionalit, které tento produkt nabízí. Vzhledem k tomu, že samotný vstup do tohoto procesu je podmíněn výběrem alespoň jedné z těchto dvou funkcionalit, není potřeba již umisťovat defaultní tok na inkluzivní bránu, která je na samotném začátku procesu.

### 3.13.6.2 Aktivace produktu Celebr8



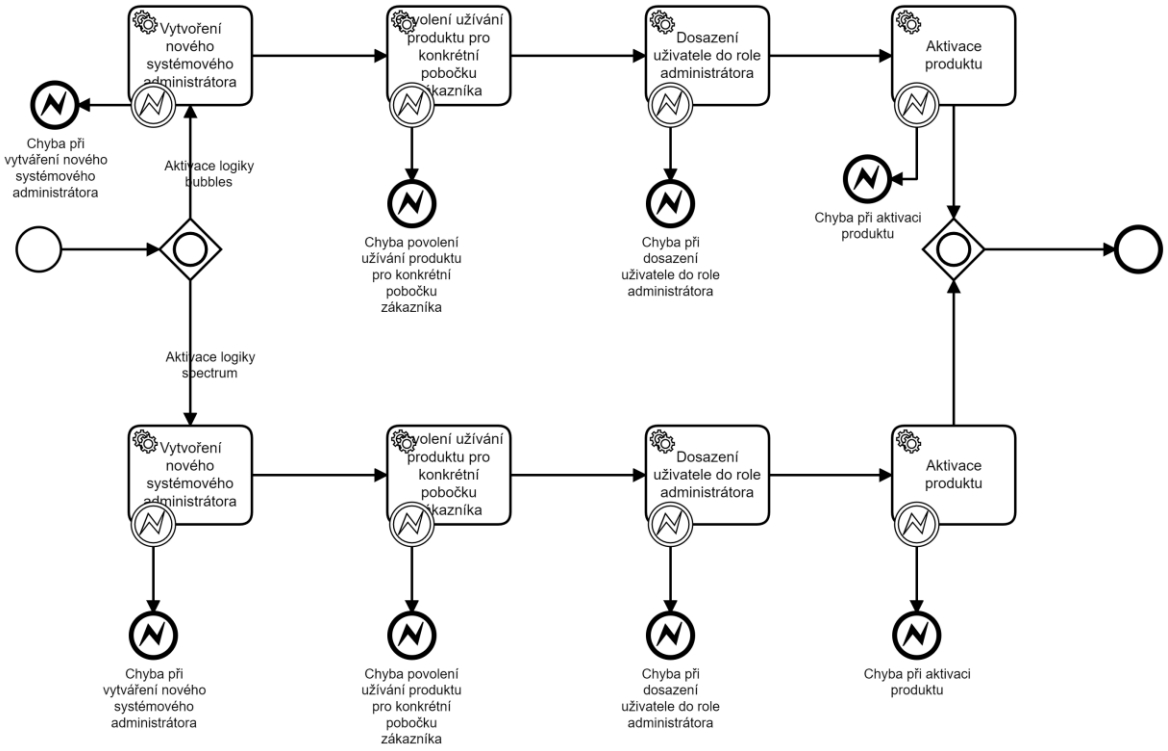

Obrázek 24 Aktivace produktu Celebr8

Prvním úkolem při aktivaci tohoto produktu je uložení zákazníkem dodaných obrázků. V rámci tohoto kroku se uloží na cloudové úložiště vybrané obrázky, které se využívají při vytváření certifikátů / ocenění.

Zbývající část procesu je naprosto totožná, jako u předešlých podprocesů. I zde vykonává roly Systémový administrátor, jenž bude tuto službu spravovat. Následně je přiděleno konkrétní pobočce oprávnění k užívání tohoto produktu a osobě, která bude vykonávat roli administrátora. Na závěr se celý produkt zaktivuje.

Pokud by nastal jakýkoliv problém, dojde k předčasnému ukončení celého procesu aktivace nového uživatele.

3.13.6.3 Aktivace produktu My spectrum and bubbles



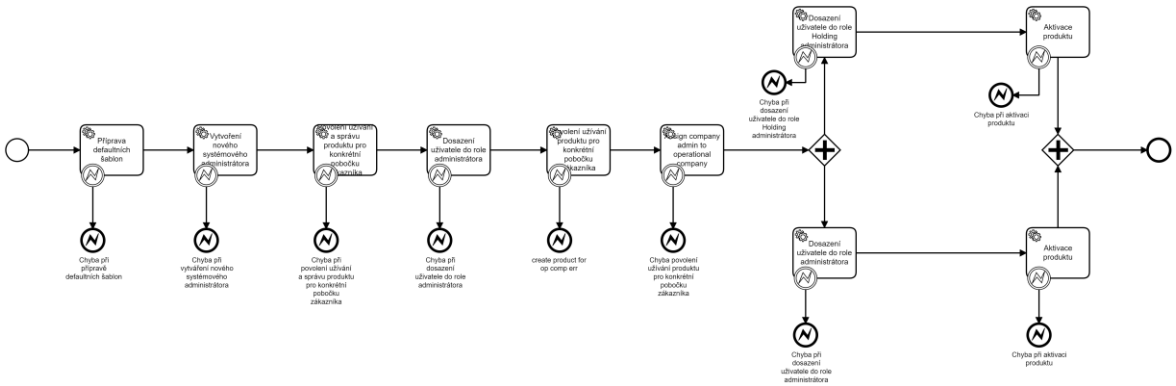

Obrázek 25 Aktivace produktu My spectrum and bubbles

Proces aktivace produktu My spectrum and bubbles je o něco složitější než u většiny procesů.

Na počátku tohoto procesu je umístěna inkluzivní brána, pomocí které lze rozdělit datový tok až na dva, v závislosti na vybraných funkcionalitách, které tento produkt nabízí k aktivaci. Jelikož ke vstupu do tohoto procesu musí uživatel zvolit k aktivaci alespoň jednu funkcionalitu, nenabízí tato brána žádný defaultní datový tok. Jinými slovy, předpokládá se, že datový tok bude pokračovat alespoň jednou z nabízených cest.

Prvky tohoto procesu jsou totožné jako v ostatních procesech. I zde se přiřazují Systémovému administrátorovi role, které ho opravňují ke správě produktu a obdobné kompetence i administrátorovi pro konkrétní pobočku. Následně se přiřadí tento administrátor k dané pobočce a na závěr se daný produkt zaktivuje.

3.13.6.4 Aktivace produktu Friday 6




Obrázek 26 Aktivace produktu Friday 6

Jak je již na první pohled zřejmé, postup aktivace Friday 6 produktu je značně rozdílný oproti předešlým procesům.

Příprava defaultních šablon je prvním z mnoha úkolů, který se nachází v tomto procesu. Jedná se o úkon, kdy se na základě nastavení jazyka vygeneruje směs předpřipravených defaultních otázek.

Následně se Systémovému administrátorovi přiřadí nová role, pomocí které bude moci tento produkt spravovat.

V rámci následujícího úkolu se nejenže povolí užívání tohoto produktu dané pobočce firmy, ale zároveň tato pobočka bude mít i právo tento produkt spravovat. Poté se přiřadí konkrétní zaměstnanec na pozici Holding administrátora, který bude mít za úkol dané dotazníky spravovat.

V dalším kroku se nastaví konkrétní pobočce daného zákazníka právo k užívání tohoto produktu.

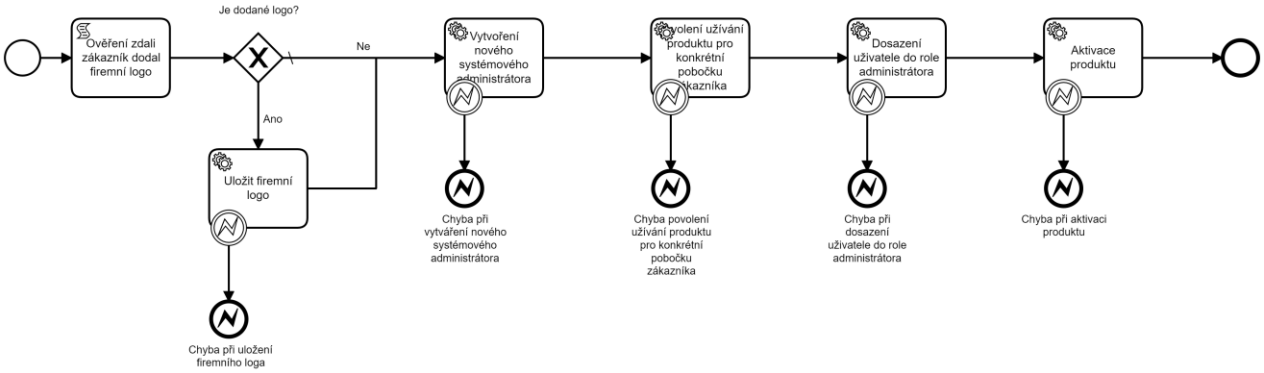
Poté se spustí logika, která slouží k dosazení konkrétního uživatele do role administrátora dané pobočky, pro kterou se tento produkt nastavuje.

Vzhledem k tomu, že následující kroky mohou běžet souběžně, je zde umístěna paralelní brána, pomocí které se datový tok štěpí na dva.

V prvním z datových toků se přiřadí nově vytvořený Holding administrátor k hlavní pobočce firmy a následně dojde k dokončení nastavení produktu pro tuto konkrétní pobočku.

Ve druhém datovém toku probíhá takřka stejný proces, i zde se přiřadí administrátor ke konkrétní pobočce a na závěr je dokončena aktivace produktu.

### 3.13.6.5 Aktivace produktu Feedback




Obrázek 27 Aktivace produktu Feedback

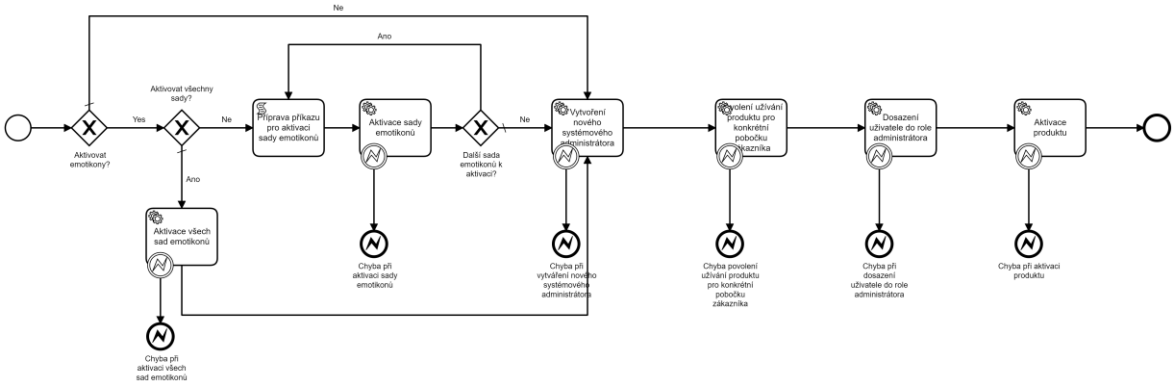
Postup aktivace tohoto produktu je velmi podobný těm, které jsou uvedeny v předchozích kapitolách. Hlavním rozdílem je samotný začátek procesu.

V rámci prvního úkolu se za pomoci skriptu ověřuje, zdali konkrétní k tomuto účelu existující složka obsahuje firemní logo zákazníka. Pokud by skript složku nedohledal, nebo pokud by byla prázdná, pokračuje datový tok defaultní cestou pojmenovanou Ne.

Pokud by skript firemní logo ve složce nalezl, pokračuje datový tok do úkolu Uložit firemní logo, který slouží k nahrání firemního loga do systému k pozdějšímu užití. V případě, že zákazník své logo nedodá, užívá se defaultní obrázek.

Zbývající část tohoto procesu je již stejná jako v předchozích mikroslužbách. Systémovému administrátorovi se přiřadí nová role, následně je produkt přiřazen konkrétní pobočce, poté se k ní přiřadí administrátor a na závěr se celý produkt zaktivuje.

### 3.13.6.6 Aktivace chatu




Obrázek 28 Aktivace možnosti chatu

Platforma LutherOne dává zákazníkům možnost v rámci chatu zakázat užívání jakýchkoliv emotikonů, povolení jen vybraných sad emotikonů či povolení všech.

Z tohoto důvodu je na začátku tohoto procesu umístěna exkluzivní brána, sloužící k rozhodnutí, zdali chce uživatel aktivovat alespoň jednu sadu emotikonů či nikoliv.

Pokud firma uživatelům ponechá možnost užívání emotikonů, přechází se k další exkluzivní bráně, kde se rozhoduje, zdali se zaktivují všechny dostupné sady či jen některé. Pokud se zákazník rozhodne zaktivovat všechny sady, datový tok povede k úkolu Aktivace všech sad emotikonů, který je typem, služba.

V případě, že si zákazník vybere k zaktivování pouze některé sady z dostupných emotikonů, které jsou uloženy v poli textových řetězců, pokračuje datový tok k úkolu Příprava příkazu pro aktivaci sady emotikonů, typu skript. V rámci tohoto úkolu se předpřipraví požadavek na zaktivování první ze sad emotikonů. V následujícím kroku se poté vyšle požadavek na zaktivování pomocí RESTové služby.

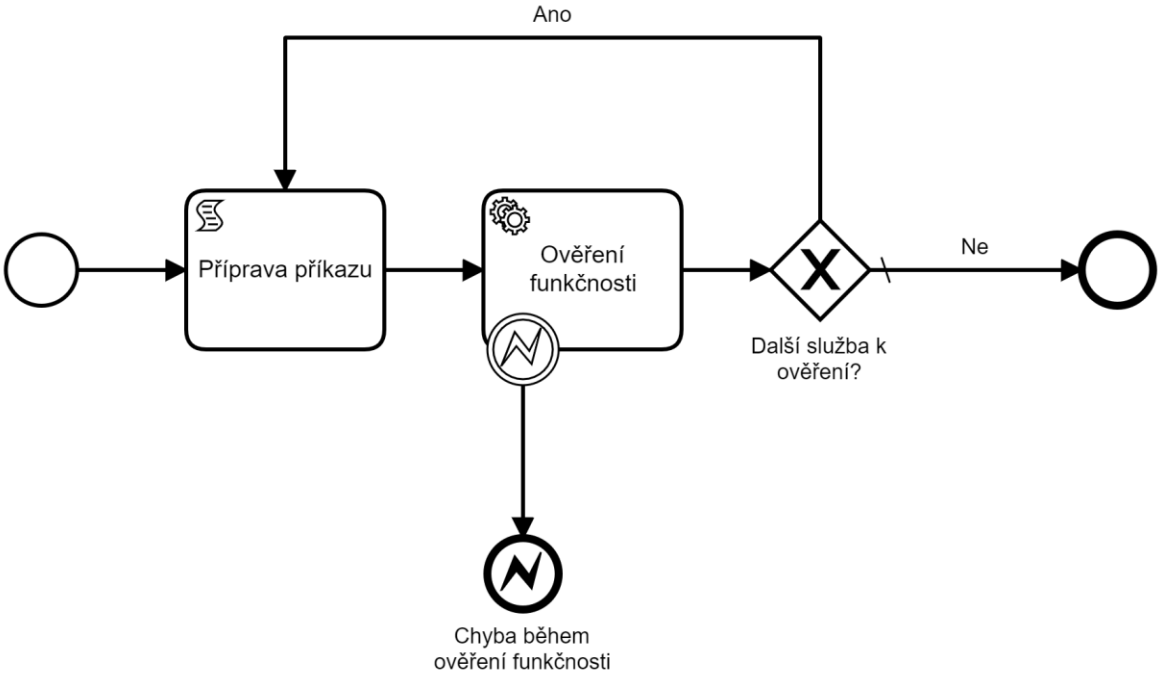
Po dokončení aktivace první z vybraných sad přechází datový tok k další exkluzivní bráně, kde se na základě podmínky ověří, zdali již byly všechny vybrané sady emotikonů zaktivovány.

Pokud se tak ještě nestalo, vrací se datový tok zpět do úkolu Příprava příkazu pro aktivaci sady emotikonů, kde se předpřipravuje další požadavek na aktivaci. Tímto způsobem se opakovaně datový tok cyklí, až jsou všechny zákazníkem vybrané sady emotikonů zaktivovány.

Následně se Systémovému administrátorovi přiřadí nové role, pomocí kterých bude moci tento produkt spravovat.

Dále se tento produkt a později i příslušný administrátor přiřadí konkrétní pobočce, a na závěr se celý produkt zaktivuje.

### 3.13.7 Ověření funkčnosti služeb




Obrázek 29 Ověření funkčnosti služeb

Poté, co jsou všechny vybrané produkty zaktivovány, se přechází k procesu, který slouží pouze ke kontrole, že všechny mikroslužby fungují správně.

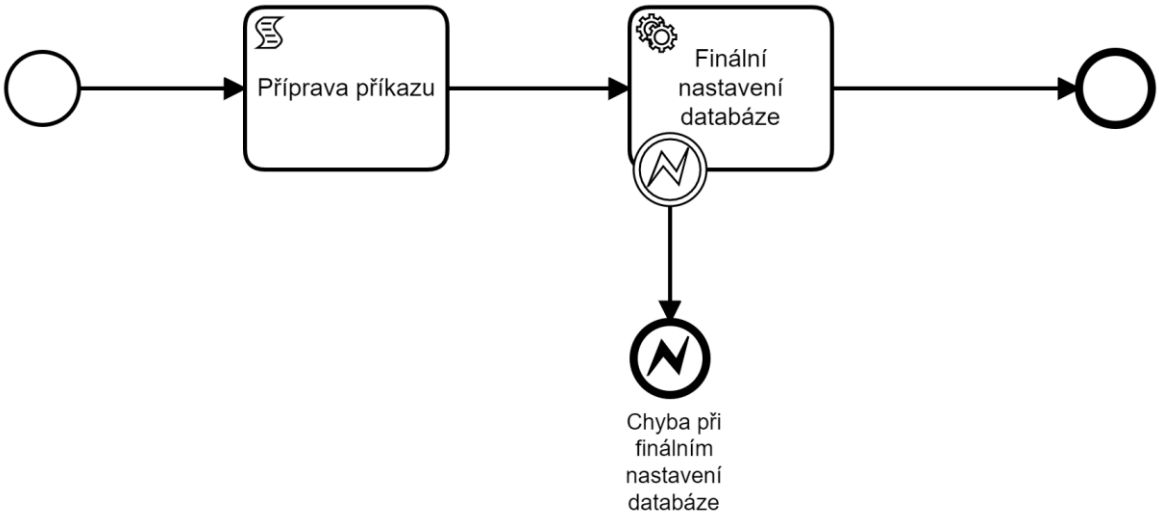
V rámci tohoto procesu se postupně provolá tzv. health check endpoint každé z mikroslužeb, jejímž úkolem je vrátit validní kód, pomocí kterého se přesvědčíme, že mikroslužba fungovala tak, jak měla.

Na počátku tohoto procesu je umístěn úkol Příprava příkazu typu skript, v rámci kterého se předpřipraví požadavek na ověření funkčnosti dané mikroslužby. V případě, že by byl službou vrácen nevalidní kód, je jisté, že došlo k nějaké chybě a celý aktivační proces bude ukončen.

Pokud ale služba vrátí validní kód, pokračuje datový tok do exkluzivní brány, kde se ověří, zdali je zapotřebí zkontrolovat další mikroslužbu. Pokud existuje zaktivovaná služba, která ještě nebyla ověřená, vrací se datový tok zpět do úkolu Příprava příkazu, kde se předpřipravují požadavky na ověření funkčnosti další z mikroslužeb.

Jakmile jsou všechny zaktivované mikroslužby prověřeny, pokračuje datový tok do koncového stavu procesu.

### 3.13.8 Finální nastavení databáze




Obrázek 30 Finální nastavení databáze

Pokud je již ověřeno, že všechny zaktivované mikroslužby fungují správně, přechází datový tok do procesu Finální nastavení databáze.

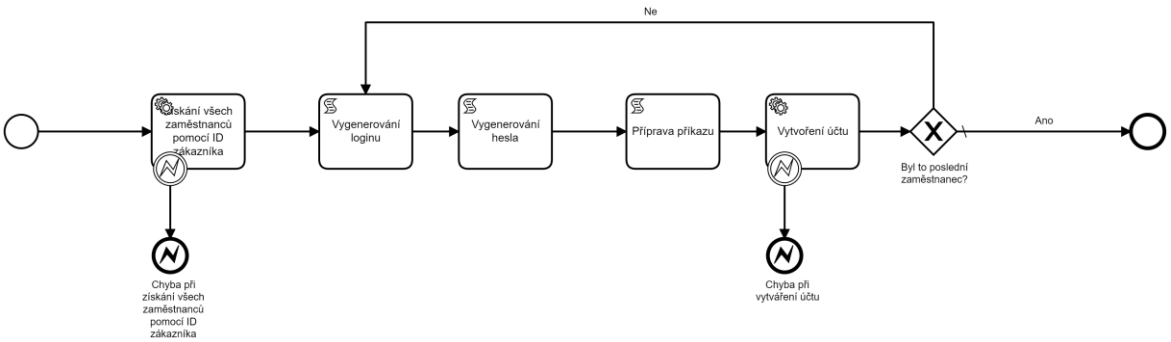
Jedná se o velmi jednoduchý úkon, kde se pouze předpřipraví příkaz, který zašle na jednu z mikroslužeb, pomocí které se dokončí finální nastavení databáze, aby byla kompletně připravená k okamžitému užívání.

### 3.13.9 Získání ID zákazníka

Po finálním nastavení databáze se pomocí dotazu získá unikátní identifikátor, sloužící k detekci zákazníka na straně databáze.

Tento identifikátor se posléze použije v následujících krocích.

### 3.13.10 Vytvoření uživatelských účtů




Obrázek 31 Vytvoření uživatelských účtů

V prvním kroku tohoto procesu se pomocí identifikátoru zákazníka získá z databáze seznam všech zaměstnanců, pro které se bude vytvářet nový uživatelský účet.

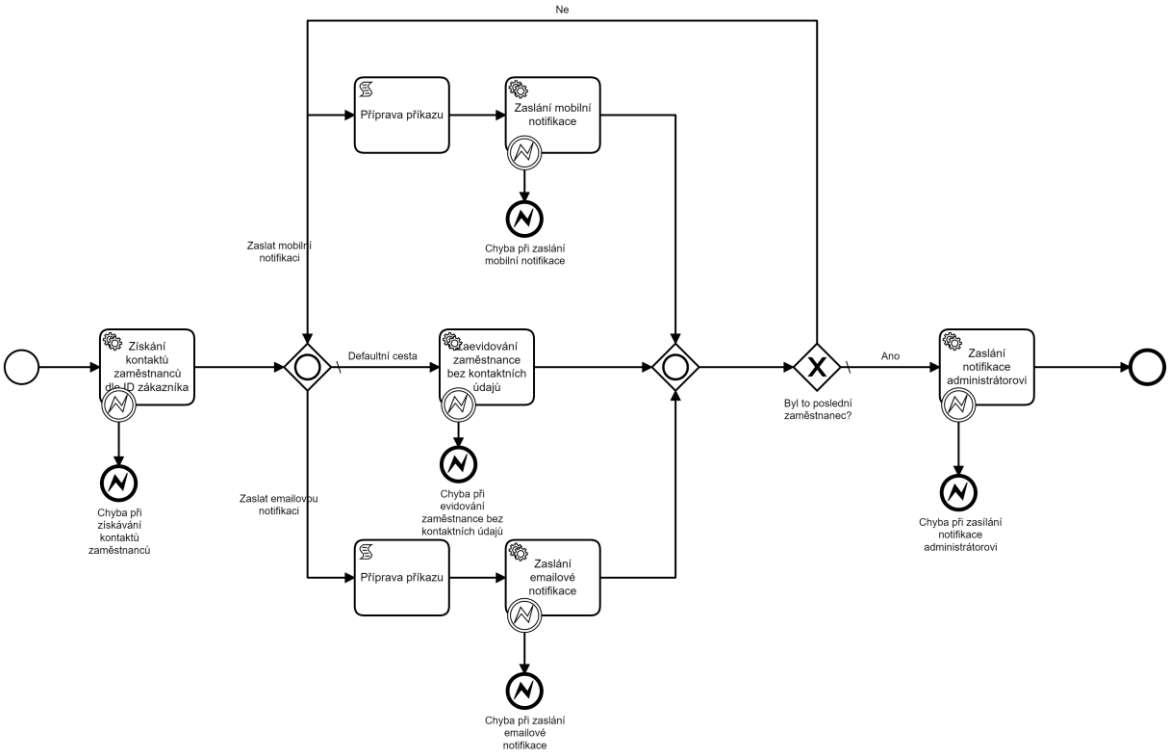
Následující kroky se cyklicky opakují pro každého ze zaměstnanců. Na základě jména zaměstnance se vygeneruje jeho přihlašovací jméno.

Následuje skript pro vygenerování defaultního hesla, které uživatel bude muset použít pro první přihlášení.

Jakmile jsou přihlašovací údaje připravené, zašle se do systému příkaz na vytvoření nového uživatelského účtu.

Po zaslání tohoto příkazu následuje exkluzivní brána ověřující zdali již byly vytvořeny všechny účty. V případě, že se tak prozatím nestalo, vrací se datový tok zpět do úkolu Vygenerování loginu, kde se začne připravovat vytvoření dalšího z účtů.

### 3.13.11 Zaslání oznámení




Obrázek 32 Zaslání notifikací

Po vytvoření uživatelských účtů se přechází k poslednímu kroku, kterým je zaslání notifikací.

V prvním kroku tohoto procesu se získají kontaktní údaje všech zaměstnanců určitého zákazníka za využití již dříve získaného unikátního identifikátoru. Tyto údaje mohou být dvojího typu, email či telefonní číslo.

Následuje fáze procesu, která se cyklicky opakuje pro zaslání notifikace každého ze zaměstnanců. Tato cyklická část začíná inkluzivní bránou rozhodující, zdali se zaměstnanci zašle oznámení formou emailu, SMS zprávy či oběma způsoby. Pokud zákazník nedodal

kontakt na daného zaměstnance, pokračuje datový tok defaultní větví, kde dojde k zaslání informace do systému o uživateli, který prozatím nedostal oznámení o vytvoření účtu. Bez takto získané notifikace se uživatel do systému nemůže přihlásit.

Pokud však má zaměstnance přidělený alespoň jeden kontaktní údaj, zašle se mu notifikace obsahující jeho přihlašovací údaje.

Po zaslání notifikací pokračuje datový tok do exkluzivní brány, kde se ověřuje, zdali již byla zaslána oznámení všem uživatelům. Pokud existuje alespoň jeden uživatel, kterému se tato notifikace stále neposlala, vrací se datový signál zpět do inkluzivní brány ověřující, jakou formou se zašlou uživateli přihlašovací údaje.

Po zaslání notifikací všem uživatelům se přejde k finálnímu úkolu, kde se zašle notifikace i odbornému pracovníkovi spravujícímu celý proces, aktivace nového zákazníka, který ho informuje o jeho úspěšném dokončení.

# 4 Výsledky a diskuse

Tématem této diplomové práce bylo využití architektury mikroslužeb pro automatizaci firemních procesů.

Základem bylo důkladné prostudování problematiky za využití odborné literatury v tištěné i elektronické podobě, které autor využil především při vypracování praktické části práce.

Jejím výsledkem je komplexní návrh využití architektury mikroslužeb pro automatizaci firemních procesů. Za pomoci tohoto návrhu by mělo být možno vytvořit funkční mikroslužbu zajišťující automatizaci všech procesů v konkrétním podniku.

Úspěšné dokončení návrhu řešení záviselo především na vhodném výběru jednotlivých technologií na základě požadavků zadavatele. Po jejich důkladné analýze se zhotovitel rozhodl vytvořit návrh mikroslužby pomocí programovacího Jazyku Java. Návrh automatizace jednotlivých procesů poté byl vyvíjen za využití softwaru Camunda.

Další podstatnou podmínkou úspěšného dokončení byl i následný návrh integrace nově vzniklé mikroslužby do již existující infrastruktury společnosti. Zhotovitel se rozhodl vytvořit samostatnou mikroslužbu, která bude komunikovat s ostatními mikroslužbami za využití RESTového rozhraní.

Po tomto návrhu zhotovitel vytvořil i možné řešení automatizace již konkrétního procesu v podniku. Jednalo se o proces sloužícího k aktivaci platformy pro nově příchozího zákazníka.

Funkčnost tohoto návrhu byla posléze prověřena během implementace a následném otestování v provozu. Dle odezvy zadavatele lze návrh považovat za úspěšný.

# 5 Závěr

Hlavním cílem této diplomové práce bylo vytvoření a ověření využití návrhového vzoru mikroslužeb při automatizaci firemních procesů.

V rámci této práce autor navrhl řešení automatizace jednotlivých procesů ve vybraném podniku. Aby dosáhl stanoveného cíle, musel si nejdříve důkladně prostudovat danou problematiku za využití odborné literatury v tištěné i elektronické podobě. Jednotlivé získané poznatky byly posléze zaznamenány v teoretické části diplomové práce.

Během ověřování využití zvoleného návrhového vzoru autor zanalyzoval požadavky od zadavatele, kterým byla firma LutherOne. Na jejich základě vybral vhodné technologie, které byly užity buď během návrhu automatizace vybraného firemního procesu či během implementace navrženého systému jako celku.

Autor navrhnuté řešení posléze otestoval v průběhu automatizace vybraného firemního procesu, který slouží k aktivaci produktu pro nově příchozího zákazníka.

Na základě odezvy ze strany zadavatele je patrné, že automatizace tohoto procesu je pro firmu přínosem a je předpoklad, že firma bude v budoucnu automatizovat i další ze svých procesů.

# 6 Seznam použitých zdrojů

## 6.1 Knižní zdroje

[1] ŘEPA, Václav. Podnikové procesy: procesní řízení a modelování. 2., aktualiz. a rozš. vyd. Praha: Grada, 2007. Management v informační společnosti. ISBN 978-80-247-2252-8.

[2] ŘEPA, Václav. Procesně řízená organizace. Praha: Grada, 2012. Management v informační společnosti. ISBN 978-80-247-4128-4.

[3] SVOZILOVÁ, Alena. Zlepšování podnikových procesů. Praha: Grada, 2011. Expert (Grada). ISBN 978-80-247-3938-0.

[4] ŠMÍDA, Filip. Zavádění a rozvoj procesního řízení ve firmě. Praha: Grada, 2007. ISBN 8024716798.

[7] HEROUT, Pavel. Učebnice jazyka Java. 5., rozš. vyd. České Budějovice: Kopp, 2010. ISBN 978-80-7232-398-2.

[9] KOEGH, James. Java bez předchozích znalostí: průvodce pro samouky. Brno: Computer Press, 2005. ISBN 978-80-251-0839-0.

[12] Indrasiri, K., & Siriwardena, P. (2018). Microservices for the enterprise: Designing, developing, and deploying (1st ed.). APRESS.

[13] NEWMAN, Sam. Building microservices. Sebastopol, CA: O´Reilly, [2015]. ISBN 978-1491950357.

## 6.2 Internetové zdroje

[5] BPMN Specification – Business Process Model and Notation. BPMN Specification – Business Process Model and Notation [online]. Dostupné z: http://www.bpmn.org/

[6] Workflow and Decision Automation Platform | Camunda. Workflow and Decision Automation Platform | Camunda [online]. Copyright © 2020 [cit. 31.10.2020]. Dostupné z: https://camunda.com/

[10] KING, Gavin, BAUER, Christian, ANDERSEN, Max Rydahl, BERNARD, Emmanuel, EBERSOLE, Steve a FERENTSCHIK, Hardy. Hibernate Reference Documentation [online]. [cit. 2019-03-05] Dostupný z WWW: http://docs.jboss.org/hibernate/core/3.6/reference/en-US/pdf/hibernate_reference.pdf

[11] JOHNSON, Rod, JUERGEN, Hoeller, donald, Keith, SAMPALEANU, Colin, HARROP, Rob, ARENDSEN, Alef, RISBERG, Thomas, DAVISON, Darren, KOPYLENKO, Dmitriy, POLLACK, Mark, TEMPLIER, Thierry, VERVAET, Erwin, TUNG, Portia, HALE, Ben, COLYER, Adrian, LEWIS, John, LEAU, Costin, FISHER, Mark, BRANNEN, Sam, LADDAD, Ramnivas, POUTSMA, Arjen, BEAMS, Chris,

ABEDRABBO, Tareq, CLEMENT, Andy, SYER, Dave a GIERKE, Oliver. Spring Framework: Reference Documentation [online]. [cit.2019-03-05]. Dostupný z WWW: https://docs.spring.io/spring/docs/3.0.x/spring-framework-reference/pdf/springframework-reference.pdf

[14] VETTOR, Rob and Nick SCHONNING, 2020. Database-per-microservice. .NET application architecture guide [online]. Available at: https://docs.microsoft.com/enus/dotnet/architecture/cloud-native/database-per-microservice

[15] LutherOne [online]. Copyright © [cit. 19.01.2021]. Dostupné z: https://www.lutherone.com/video/performance/okr.png

[16] LutherOne [online]. Copyright © [cit. 19.01.2021]. Dostupné z: https://www.lutherone.com/video/performance/team.jpg

[17] LutherOne [online]. Copyright © [cit. 19.01.2021]. Dostupné z: https://www.lutherone.com/_nuxt/img/Tasks1.9cddd94.svg

[18] LutherOne a.s., Praha IČO 07335377 - Obchodní rejstřík firem | Kurzy.cz. Obchodní rejstřík firem - vazby a vztahy z justice.cz | Kurzy.cz [online]. Copyright © 2000 [cit. 23.01.2021]. Dostupné z: https://rejstrik-firem.kurzy.cz/07335377/lutherone-as/

[19] LutherOne [online]. Copyright © [cit. 19.01.2021]. Dostupné z: https://www.lutherone.com/_nuxt/img/certificate.1903f6d.svg

[20] LutherOne [online]. Copyright © [cit. 19.01.2021]. Dostupné z: https://www.lutherone.com/_nuxt/img/awards.430c7cb.svg

[21] LutherOne [online]. Copyright © [cit. 19.01.2021]. Dostupné z: https://www.lutherone.com/_nuxt/img/badges.8b60890.svg

[22] LutherOne for Android - APK Download. Download APK free online downloader | APKPure.com [online]. Copyright © 2014 [cit. 23.01.2021]. Dostupné z: https://apkpure.com/lutherone/com.lutherx.lutherone

[23] LutherOne [online]. Copyright ©Y [cit. 24.01.2021]. Dostupné z: https://www.lutherone.com/_nuxt/img/friday6-card-opened.251a18a.png

[24] Feedback. LutherOne [online]. Copyright ©2021 LutherOne [cit. 24.01.2021]. Dostupné z: https://www.lutherone.com/platform/products/feedback

[25] MILLER, Ron. How AWS came to be. In: TechCrunch [online]. New York City, New York, USA: Oath, 2018, 2016-07-02 [cit. 2018-01-08]. Dostupné z: https://techcrunch.com/2016/07/02/andy-jassys-brief-historyof-the-genesis-of-aws/

[26] SAJEE, Mathew. Overview of Amazon Web Services. In: Amazon Web Services: AWS Whitepapers [online]. Seattle, Washington, USA: Amazon Web Services, 2015 [cit. 2017-03-28]. Dostupné z: https://d0.awsstatic.com/whitepapers/aws-overview.pdf

[27] AWS logo
Dostupné z: https://d3c9ouasuy8pg6.cloudfront.net/dist/images/aws-logo-light_2a8d69e93c95850234f1c278e70f7ddb.png

[28] LutherOne [online]. Dostupné z: https://www.lutherone.com/_nuxt/img/Feed1.d58b50b.png

[29] LutherOne [online]. Dostupné z: https://www.lutherone.com/cs/platform/products/performance-management#task-management

[30] JSON. JSON [online]. Dostupné z: https://www.json.org/json-en.html

[31] Roy Thomas Fielding, Architectural Styles andthe Design of Network-based Software Architectures, CHAPTER 5 Representational State Transfer (REST) [online]. Dostupné z: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm