Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Výběr a implementace API Gateway řešení pro cloud i on-premise použití

Plzeň, 2020 Tomáš Rozsypal

Prohlášení

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

V Plzni dne 21. července 2020

Tomáš Rozsypal

Poděkování

Tímto bych rád poděkoval svému vedoucímu bakalářské práce Ing. Stanislavu Horáčkovi za odborné vedení, za pomoc a cenné rady při zpracování této práce. Dále bych chtěl rád poděkovat svému konzultantovi bakalářské práce Ing. Romanu Moučkovi, Ph.D. za cenné rady, věcné připomínky a vstřícnost při konzultacích této práce.

Abstract

The aim of this work is to analyze and select a suitable API Gateway solution for the company AIMTEC a.s. This paper opens with an overview of the general issues of application programming interface gateways and describes its individual components. The next section of the paper compares existing API Gateway solutions. Among these solutions, a solution was selected that meets the specifications defined by AIMTEC a.s. The conducted research carried out selected and described a suitable solution, ie Gravitee.io. In the practical part, APIS were deigned on the Gravitee.io API Gateway platform. The API was created using their user interface. Another creation option was to use the implemented configuration tool, which imports the API represented by the JSON format. The conclusion describes testing of the configuration tool and API and suggestions for possible further extensions are proposed.

Abstrakt

Cílem této práce je analyzovat a vybrat vhodné API Gateway řešení pro společnost AIMTEC a.s. Práce nejdříve shrnuje obecnou problematiku bran aplikačních programovacích rozhraní a popisuje její jednotlivé komponenty. V další části se porovnali existující API Gateway řešení. Mezi tato řešení bylo vybráno řešení, které splňuje specifikace definované společností AIMTEC a.s. Provedeným výzkumem bylo vybráno a popsáno vhodné řešení, tedy Gravitee.io. V praktické části byly navrhnuty API na platformě Gravitee.io API Gateway. API byla vytvořena pomocí jejich uživatelského rozhraní. Jinou možností vytvoření bylo pomocí implementovaného konfiguračního nástroje, který importuje API reprezentováno JSON formátem. V závěru je popsáno testování konfiguračního nástroje a API a jsou zde navrhnuty náměty pro případná další rozšíření.

Obsah

1 Úvod

Aplikační rozhraní jsou v dnešní době důležitá pro vzájemnou spolupráci a komunikaci softwarových programů. Při vhodném spravování těchto aplikačních rozhraních lze docílit efektivní spolupráce mezi programy nebo aplikacemi. O správu API se stará API management, který je součástí API Gateway řešení. Cílem této práce je vybrat vhodné řešení pro společnost AIMTEC a.s. V této práci jsou popsány jednotlivé komponenty brány aplikačních programovacích rozhraní a je zde analyzován trh s API Gateway řešeními. Na základě tohoto průzkumu vybrat vhodné řešení tak, aby řešení splňovalo specifikace nadefinované firmou.

První část této práce je věnována teoretickému rozboru všech komponent. Jsou zde popsány aplikační rozhraní, brány, brány pro správu aplikačních programovacích rozhraní a jejich role v architektuře mikroslužeb. Dále se v této práci analyzují některé existující vybrané API Gateway řešení, které jsou zde porovnávány na základě získaných informací. Mezi tyto informace patří například základní popis, způsob využití, jejich funkce, k čemu byly tyto funkce navrženy, jejich možnost nasazení a licencování.

Firma si definovala požadavky, které musí řešení splňovat. Požadavky se zaměřují na funkce, vlastnosti, možnosti, pluginy a politiky daného řešení. Firmu také zajímá jak lze vytvářet, monitorovat a spravovat jednotlivá API. Řešení by dále mělo podporovat běh v kontejnerech, nasazení On-Premise a definování API pomocí YAML nebo JSON. Také se zde klade důraz na druh licencování. Na závěr této části je vybráno právě jedno řešení.

Další část této práce se zabývá vybranou a schválenou bránou pro správu API. V této části se popisuje platforma Gravitee.io, jeho vznik, architektura, koncept, komponenty, nástroje, možnosti a funkce. Dále popisuje jeho vytváří, monitorování a evidence aplikační programovacích rozhraní. V této praktické části je navrhnuta a popsána implementace konfiguračního nástroje, který umožňuje externí hromadné importování více API najednou. Na závěr se nachází testování, možná rozšíření práce a shrnutí výsledků.

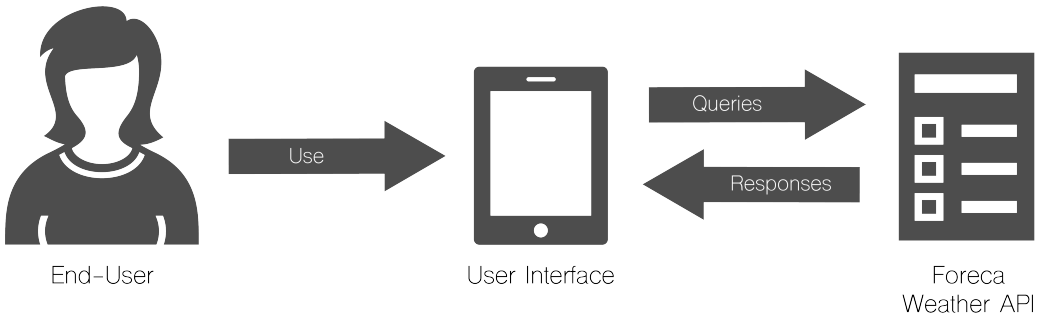
2 Analýza komponent v instanci API Gateway

V této kapitole budou obecně vysvětleny pojmy jako aplikační programovací rozhraní, brána, API Gateway atd. Dále se tato část zabývá jejich komunikací mezi sebou a propojení uživatelů s aplikacemi nebo programy. Jsou zde zodpovězeny otázky například jak tyto komponenty fungují, kde se používají a k čemu slouží. Také tu jsou popsány jejich procesy a jejich využití v praxi.

2.1 Aplikační programovací rozhraní

Aplikační programovací rozhraní nebo taky API (Application Programming Interface) je softwarový prostředník, který umožňuje dvěma aplikacím komunikovat mezi sebou. API se používá v softwarovém inženýrství. Jedná se o balík funkcí, knihoven, procedur, protokolů a tříd, které mohou být využívány programátory pro komunikaci se softwarem.

Komunikace mezi dvěma aplikacemi lze pochopit dvěma způsoby. Prvním způsobem je jednosměrná komunikace. To znamená, že první aplikace pošle data druhé aplikaci, ale ta nepošle nic zpět. Druhý způsob komunikace je komunikace obousměrná. Dochází zde tedy k výměně dat. Příkladem v praxi může být používání aplikací, jako je Facebook nebo Počasí. Posíláte zprávu někomu na Facebooku nebo kontrolujete počasí v telefonu, používáte API. Jak funguje komunikace cílového uživatele a API v aplikaci Počasí vidíte na obrázku 2.1 [1] [2].



Obrázek 2.1: Příklad použití API v aplikaci Počasí [34]

Uživatelé díky API mohou získat informace o svých produktech, mohou

se dostat k výpisům článků i k hromadnému rozesílání e-mailů nebo vystavování faktur.

Aplikační programovací rozhraní existuje na třech úrovních:

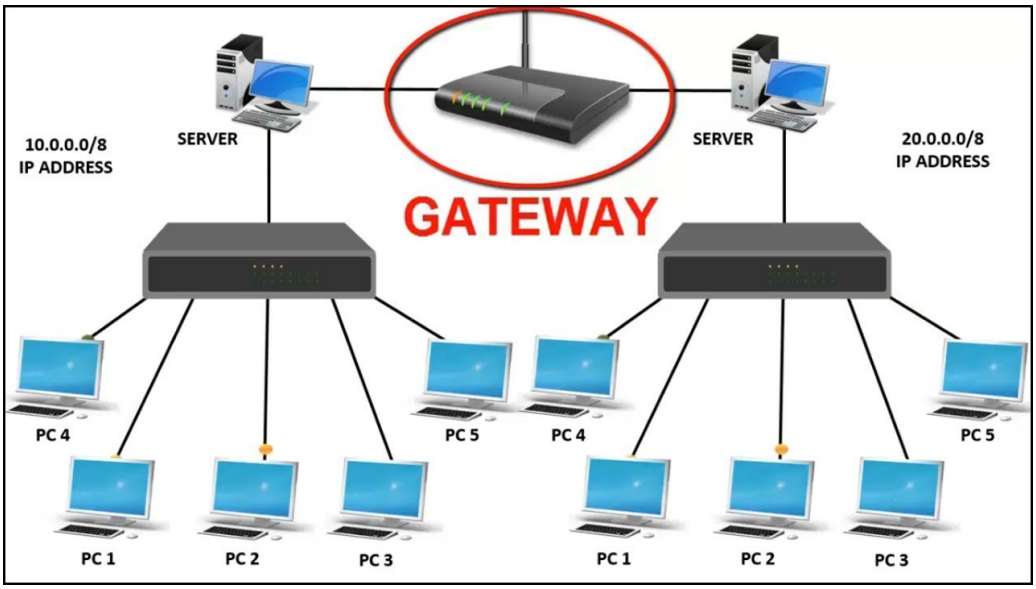
- na úrovni operačního systému
- na úrovni konkrétní aplikace nebo programu
- na úrovni webových služeb

V posledních letech API přijalo několik nových užitečných vlastností. Nyní už dodržují HTPP a REST standardy, které jsou snadno dostupné a srozumitelné pro programátory. Nové moderní API jsou specificky navrženy pro odběratele nebo zákazníky, tak aby splňovali jejich podmínky. Jsou také dokumentovány a verzovány tak, aby uživatelé věděli a měli přehled o jejich životním cyklu a jeho údržbě. Bezpečnost API se stále zlepšuje. Moderní API mají svůj vlastní životní cyklus (SDLC - Software Development Life Cycle), který lze monitorovat a spravovat. [1] [2].

2.2 Brána

Brána nebo také Gateway je uzel (router) v počítačové síti. Je klíčovým prvkem k zastavení příchozích nebo odchozích dat ze sítí. Díky branám můžeme komunikovat s okolním světem, odesílat nebo přijímat data. Bez nich by internet byl nepoužitelný. Na pracovištích je typicky bránou počítač, který směruje provoz do vnější sítě. V domácím prostředí je většinou bránou poskytovatel internetu.

Uzel je fyzické místo, kde dochází k zastavení dat kvůli transportu, čtení nebo použití. Na internetu je zastavovací uzel brána nebo hostitelský uzel. V bezdrátové síti je bránou modem, díky kterému se můžete připojit k síti vašeho poskytovatele internetu. Pokud spojení počítač-server funguje jako brána, tak také funguje jako firewall a proxy server. Na obrázku 2.2 je zobrazeno umístění brány v počítačové síti [3].



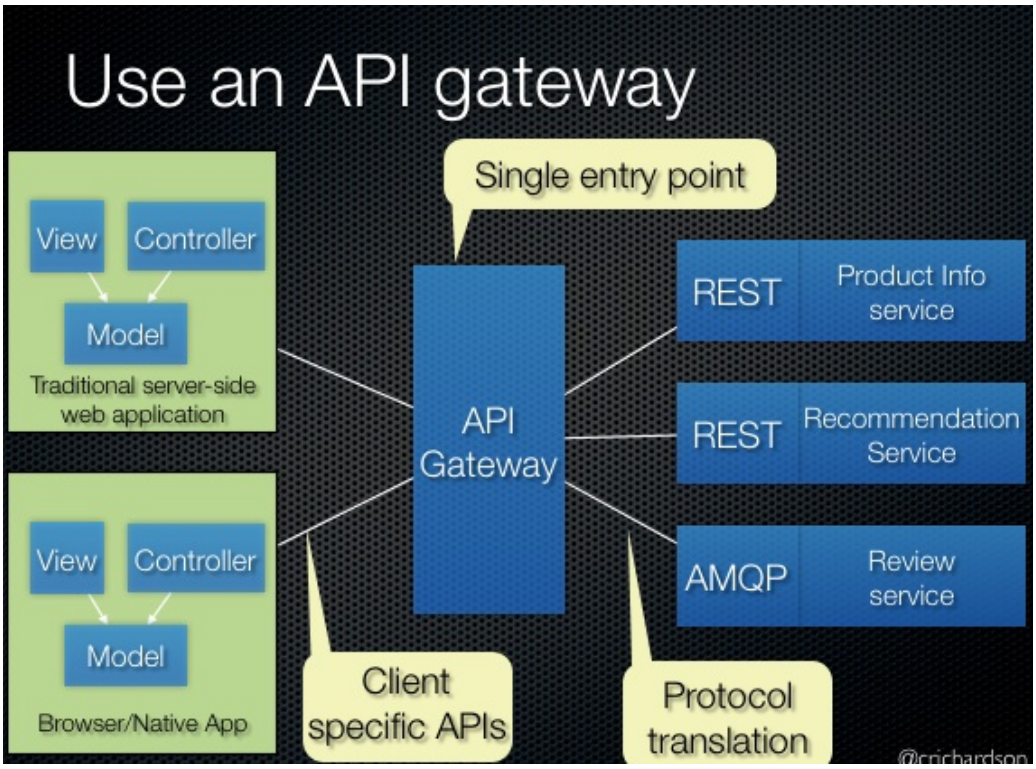
Obrázek 2.2: Brána v počítačové síti [35]

Výchozí brána sítě neboli default gateway je označení pro směrovač a je také nazývaná jako implicitní brána. Výchozí brána je důležitý síťový prvek v protokolu TCP/IP pro lepší práci s IP adresami a jejich propojení. Hlavní funkcí implicitní brány je zajištění datové komunikace paketů mezi podsítěmi. Díky ní lze připojit konkrétní uživatelský počítače k celé internetové síti a daný hostitel TCP/IP nemusí shromažďovat neustále nové měnící se data [4].

2.3 API Gateway

API Gateway neboli API brána přijímá všechny API volání (calls) od klientů a směřuje je do příslušných mikroslužeb (microservices) s vyžádáním směrování (routing), složení a překladu protokolu. Většinou zpracovává požadavek (request) voláním více mikroslužeb a agregováním výsledků, aby se určila nejlepší a nejrychlejší cesta požadavku [6].

Webové stránky e-shopů (elektronických obchodů) mohou používat API Gateway, aby poskytly mobilním klientům endpoint (koncový bod) pro načtení všech produktů a jejich podrobností jen v jednom požadavku. Tento požadavek může vyvolat několik různých služeb, jako třeba informace o produktech, jejich recenze, a vzápětí spojuje jejich odpověď (response). Jak funguje e-shop můžete vidět na obrázku 2.3 [5].



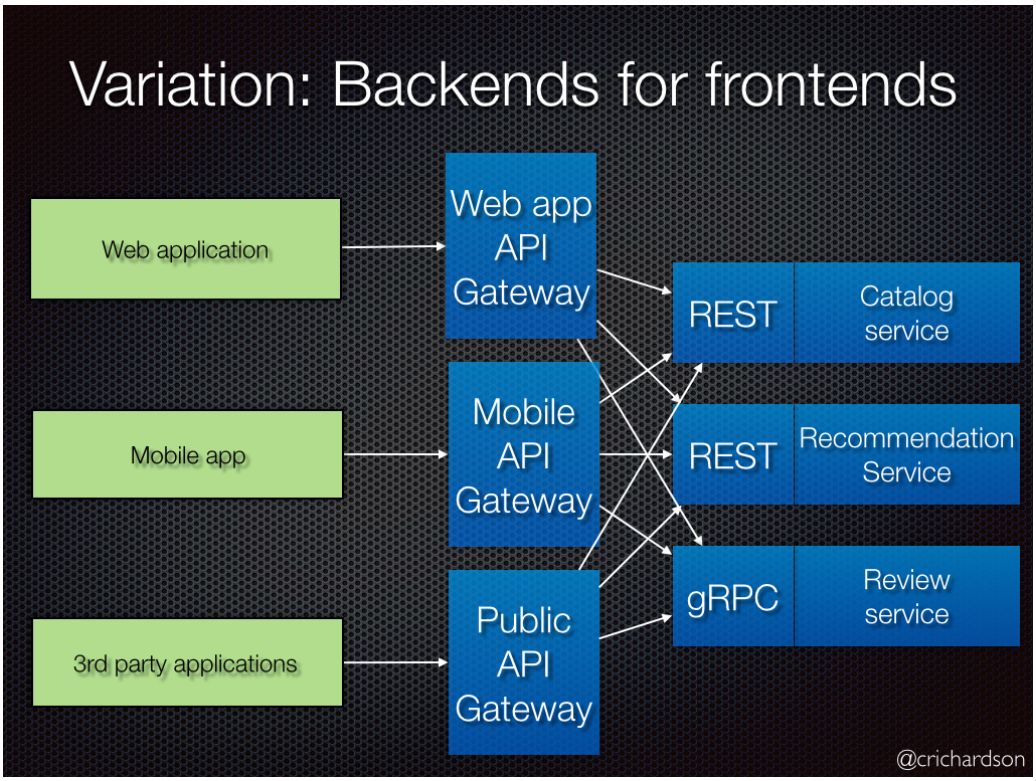
Obrázek 2.3: API brána e-shopu [5]

Jednou z hlavních výhod používání API bran je to, že vývojářům umožňují zapouzdřit vnitřní strukturu aplikace. Existuje několik způsobů jak lze tuto strukturu aplikace zapouzdřit. Záleží jen na případu použití [7].

2.3.1 Vytváření mikroslužeb pomocí API brány

API Gateway má smysl implementovat pro většinu aplikací založených na mikroslužbách. API Gateway se totiž nachází před aplikačním programovacím rozhraním (API) a funguje jako jediný vstupní bod do systému pro definovanou skupinu mikroslužeb. Tato brána je zodpovědná za směrování požadavků, složení a překlad protokolu a také ve většině případů může zrychlit a zlepšit efektivitu systému. Díky ní každá klientovaaplikace získá svoje vlastní API [5].

Některé požadavky zpracovává tak, že je jednoduše směruje na příslušnou backendovou službu (backend service). Ostatní požadavky zpracovává vyvoláním více backendových služeb a agregováním výsledků. Na obrázku 2.4 vidíte komunikaci mezi backendem a frontendem, kde se definuje API Gateway pro každý druh klienta. Pokud dojde v backend službách k selhání, tak je API Gateway může maskovat vrácením mezipaměti nebo výchozích dat [6] .

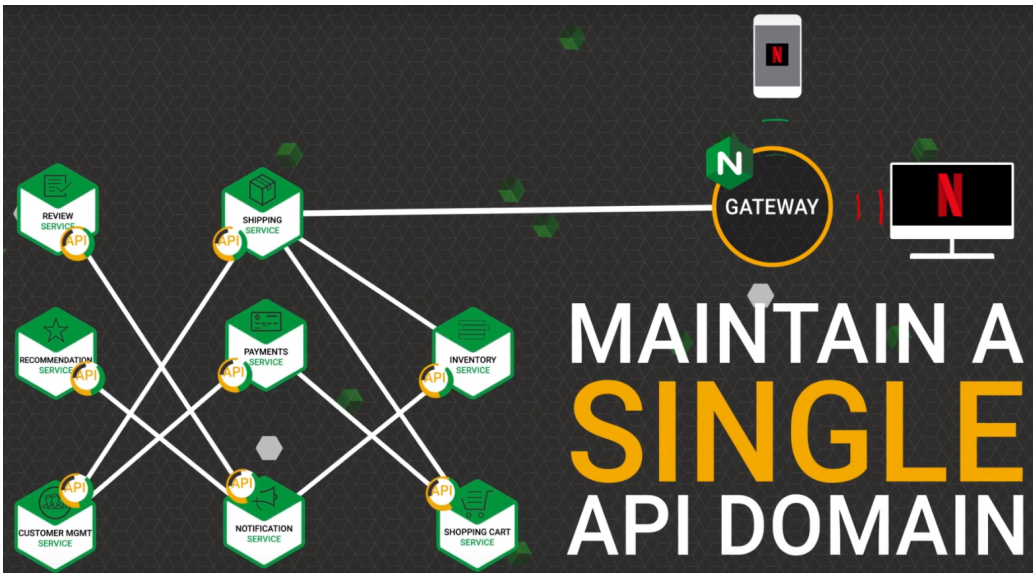


Obrázek 2.4: API Gateway backend a frontend [5]

V tomto příkladu existují tři druhy klientů: webová aplikace, mobilní aplikace a externí aplikace třetích stran. Existují tři různé API brány. Každá z nich poskytuje API pro svého klienta. API Gateway může také implementovat zabezpečení, např. ověření, zda je klient oprávněn k provedení požadavku [5].

2.3.2 Role API brány v architektuře mikroslužeb

API Gateway je dirigent, který organizuje požadavky zpracovávané architekturou mikroslužeb tak, aby uživateli poskytoval zjednodušené prostředí. Je to také překladatel, který bere více požadavků od klienta a mění je v jeden požadavek. Tím chce redukovat co největší počet volání a odpovědí mezi klientem a aplikací. API Gateway se vytvoří před mikroslužbami a stává se tak vstupním bodem pro každý nový požadavek z aplikace. Zjednodušuje to jak implementaci klienta tak i aplikaci mikroslužeb [6].



Obrázek 2.5: API Gateway v praxi [36]

Příklad funkce API brány v praxi vidíte na obrázku 2.5. Jsou tam vidět jednotlivá APIs, API Gateway a konečná aplikace, v tomto případě streamovací služba Netflix v telefonu a v televizi. Všem API požadavkům stačí jedna API Gateway.

2.3.3 Proč API brány?

Hnací silou mnoha velkých i malých aplikací jsou aplikační programovací rozhraní. Při vhodném výběru API managementu lze efektivně vylepšit spolupráci a komunikaci mezi programy nebo aplikacemi. Ať už se jedná o publikaci veřejné API nebo vytváření nového integrační trhu. API se stávají novým způsobem, jakým se dá podnikat. Stejně jako webová éra měla HTTP servery, které obsluhovaly weby, tak API mají API brány, aby obsluhovaly APIs. API brány můžou pomáhat v dodávání aplikačních programovacích rozhraní s vysokou dostupností pro zákazníky a partnery. Jedná se o typ proxy serveru, který vykonává funkce, jako je třeba autentizace, omezení rychlosti, směrování veřejných koncových bodů do vhodných mikroslužeb, vyvažování zátěže napříč několika interními službami [8].

Z historického hlediska potřeba API bran vzrostla z integračních výzev. Před REST API a GraphQL API společnosti vytvořily SOAP a XML založené na APIs skládajících se ze strukturovaných nebo nestrukturovaných dat. API brány mohou poskytovat sjednocená rozhraní a propojovat více starších aplikací dohromady. V takových případech použití mohou API brány vzít legacy (starší) službu SOAP a použít transformaci dat do API a to trans-

formací ze SOAP na REST a z JSON na XML. Tyto transformace obvykle nejsou automatické. Například rozhraní RESTful API má velmi odlišné základní principy než SOAP, a proto jejich transformace není tak jednoduchá jako transformace XML na JSON [8].

SOAP *(Simple Object Access Protocol) je protokol sloužící k výměně zpráv založených na XML přes síť pomocí HTTP.* ***XML*** *(eXtensible Markup Language) je rozšiřitelný značkovací jazyk, který je navrhnutý na uchovávání strukturovaná data.* ***REST*** *(Representational State Transfer) je architektura rozhraní, navržená pro distribuované prostředí.* ***JSON*** *(JavaScript Object Notation) je formát určený pro přenos nebo výměnu dat [12][13].*

Architektura mikroslužeb je strategie pro budování a nasazení nezávislých služeb, které umožní vytvořit jednu větší aplikaci. Z jiné perspektivy architekturu mikroslužeb můžeme brát jako způsob vytváření aplikačních programovacích rozhraní. Dalším plusem je možnost pracovat na jedné velké aplikaci v nezávislých týmech, aniž by si navzájem lezli do zelí nebo se rušili.

Kromě mikroslužeb existují ještě menší výpočetní jednotky, jako třeba nanoservices (nanoslužby). Vzhledem ke složitosti správy stovek nebo tisíce služeb a požadavků na poskytování rozhraní nebo uzavření smlouvy s vašimi klienty se API brány stávají běžným místem v architektuře, ve kterém se používají mikroslužby [8].

2.3.4 Výhody a nevýhody API bran

Použití API bran má následující výhody:

- izoluje klienty od toho, jak je aplikace rozdělena do mikroslužeb
- izoluje klienty od problému s určováním umístění instancí služby
- poskytuje optimální API pro každého klienta
- snižuje počet požadavků a jejich cest -> méně požadavků také znamená méně režijních nákladů a zvýšení uživatelského komfortu
- zjednodušuje klienta přesunutím logiky pro volání více služeb z klienta na API bránu
- převádí se ze „standardního“ veřejného webového protokolu API na jakékoli protokoly, které se používají interně

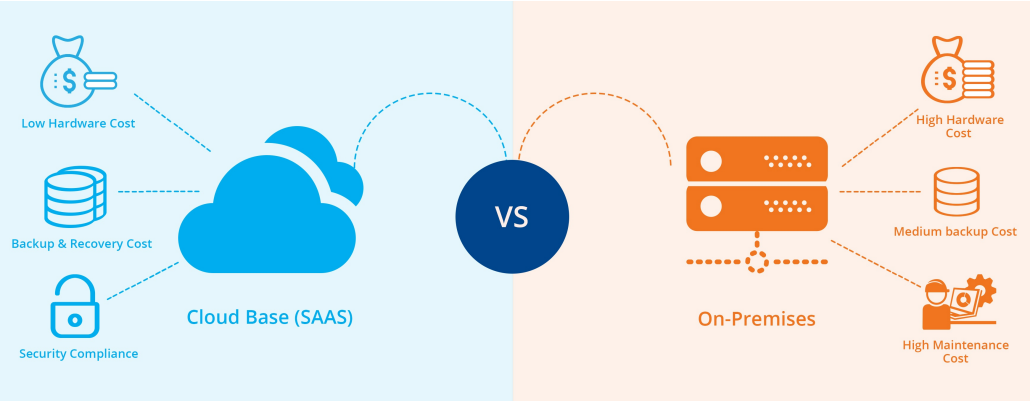
API Gateway model má i několik nevýhod:

- zvyšuje složitost - API Gateway je další částí, která musí být vyvinuta, nasazena a spravována

- kvůli přidanému síťovému bodu (tedy API Gateway) se prodloužila doba na odpověď - u většiny aplikací jsou však náklady na další cestu tam a zpět nevýznamné [5].

2.4 On-Premise vs Cloud řešení

On-Premise software a Cloud software jsou dva odlišné modely licencování a dodání. V této sekci se nachází porovnání těchto dvou modelů.



Obrázek 2.6: Cloud vs On-Premise [37]

2.4.1 On-Premise

On-Premise software je nainstalován lokálně na vašich serverech a počítačích. Většinou se platí jednorázový licenční poplatek. Musíte se však jako správce softwaru také starat o jeho údržbu a správu.

Cena a bezpečnost

Většinou jsou zapotřebí velké počáteční investice, takže skončíte s vysokými kapitálovými výdaji. Na druhou stranu za softwarovou licence platíte pouze jednou, takže TCO řešení může být v některých případech nižší než u Cloud aplikací. Kvalita bezpečnosti u toho řešení závisí pouze na vás a vašem výběru bezpečnostních opatření [11].

2.4.2 Cloud

Software as a Service neboli SaaS je software, který je hostován centrálně, většinou s licencí v podobě předplatného (subscription). Přístup do Cloudu je většinou zařízen prostřednictvím klienta. Tento model využívá většina IT společností.

Cena a bezpečnost

Některým společnostem vadí zabezpečení Cloudu a nechtějí přesouvat svá citlivá data někam "veřejně"na internet. Většina organizací chce zaručit zákazníkům vysokou bezpečnost, která je v dnešní době velice žádaná. Avšak opak je pravdou. Poskytovatelé Cloudu většinou používají taková bezpečností opatření, jaká si většina organizací nemůže dovolit [11].

3 Analýza existujících API Gateway řešení

V této kapitole budou analyzovány jednotlivá API Gateway řešení. Tyto API brány budou hodnoceny a mezi sebou porovnávány. Vybraná API Gateway musí co nejlépe splňovat podmínky, které si firma AIMTEC a.s. definovala. Tyto podmínky vyplývají z jejich nároků na potřebné funkce nebo vlastnosti, které budou potřebovat v budoucím vývoji jejich produktu. Mezi tyto funkce patří například zobrazování statistik do přehledných grafů, komunikace s webovým serverem (registrace, přihlašování a synchronizace dat), možnost implementování různých dalších pluginů nebo politik, jejich možnost nasazení, využití v praxi, cena licencování atd. Nejprve jsou stručně popsány vybrané potenciální řešení a na konci se nachází shrnutí a následné vybraní vhodného řešení.

3.1 Specifikace podmínek

Vlastnosti, funkcionality nebo nástroje, které API brány musí obsahovat. Jsou seřazeny podle váhy a důležitosti pro firmu.

- peněžní náklady (licencování)
- běh v kontejneru (Docker, Kubernetes)
- definice API (YAML nebo JSON)
- evidence a správa APIs
- On-Premise
- kompozice požadavků (Java, GraphQL)
- Cloud
- mockování API

Především jde o přehled jednotlivých aplikačních programovacích rozhraní a jaké mají dané API Gateway řešení funkce. Díky tomu lze snadno monitorovat a popřípadě omezovat jednotlivá API. AIMTEC také chce, aby šlo

definovat API pomocí uživatelského rozhraní, ale také aby šly API importovat externě a hromadně. Dost zákazníků chce, aby vybrané řešení běželo u nich na jejich lokálních serverech, tedy on premise. Dále se firma zaměřuje na kompozici požadavků. Klient pošle jeden požadavek, ale Gateway jich pošle několik (klidně i na více služeb najednou). Gateway poté poskládá více odpovědí, které dostala od služeb, do jedné odpovědi, kterou zašle klientovi. AIMTEC také klade důraz na jednoduché nasazování, takže hledá Gateway, která podporuje Docker/Kubernetes. Také jim záleží na druhu licencování, kde chce AIMTEC samozřejmě co nejlevější řešení.

Ideálně by také firma chtěla, aby vybrané API Gateway řešení mělo možnost si vytvořit nebo implementovat svoje vlastní pravidla, politiky nebo pluginy. Tato možnost by mohla vyřešit třeba chybějící funkci nebo vlastnost.

Na základě průzkumu trhu jsem narazil na tyto řešení 3scale, Apigee, Axway AMPLIFY, Kong, TIBCO Mashery, Akana, CA Technologies, Cloud Elements, IBM API Connect, Dell Boomi, Oracle API Manager, MuleSoft, SAP, Software AG, Azure API Gateway, Express API Gateway, App42, DreamFactory, Informatica US, Tyk, WSO2, Nevatech Sentinet, KrakenD, Ceptor, APIman, Wicked.haufe.io, Fusio, Gravitee.io, Repose, API Umbrella a Apigility.

Díky analýzám, hodnocením, zpětným vazbám, průzkumům a licencováním jsem bych schopen seznam API Gateway řešení zúžit. V následující části jsou stručně popsány tyto řešení: Apigee, Kong, Tyk, Amazon AWS API Gateway, Wicked.haufe.io, APIman, Fusio, Repose a Gravitee.io.

3.2 Existující řešení

3.2.1 Apigee

Apigee je jedna z nejstarších API Gatewayí. Byla navržena v roce 2004 a v 2016 ji získala společnost Google. Není open source a je navržena v enterprise Javě. Zpočátku byla Apigee spuštěna jako XML / SOA aplikace, ale jejich cílem byl od začátku API management. Apigee byla navržena tak, aby přeměňovala starší monolity na aplikační rozhraní, které pak budou moci používat firmy třetí strany. Méně se zaměřují na mikroslužby a interní API.

Nasazení

Protože Apigee má složitou architekturu a její nasazení je mnohem složitější vzhledem ke konkurenci open source API Gatewayí. Apigee Edge on premise vyžaduje běh minimálně 9 uzlů a zahrnuje také běh klastrů Cassandra,

Zookeeper a Postgres. Důsledkem toho je nasazení složité a časově náročné.

Většina zákazníků Apigee používá on premise verzi, ale po připojení do Googlu se zpřístupní Cloud, tedy Googlem hostované řešení. Je to však blíže IaaS (Infrastructure as a service) než SaaS (Software as a service). Navíc to musí být nasazeno do konkrétního Google Cloud datového centra. U Cloudové verze hostované někým jiným se zvětší latence a vyžaduje zabezpečení vlastních služeb. Další latenci může způsobit používaní hostované API Gatewaye v jiném datovém centru než jsou jiné předcházející služby.

Funkce

Na rozdíl od jiných podporuje Apigee integrovanou fakturaci (integrated billing) pro zpeněžení aplikačních rozhraní. Apigee management portál využívá Drupal. Drupal je softwarový systém pro správu obsahu využíván především v žurnalistice. V závislosti na perspektivě se Apigee může zdát buď nedostačujícím nebo naopak dostačujícím řešením, které splňuje veškeré požadavky. Zároveň je proprietární a nemá moc velkou komunitu vývojářů, která by vytvářela nové pluginy nebo rozšíření [8].

3.2.2 Kong

Kong je open source API Gateway. Základem Kongu je NGINX, což je velmi populární open source HTTP proxy server. Přestože je Kong open source má v nabídce licenci KongHQ, která poskytuje údržbu a podporu většinou velkým podniků. V open source verzi jsou k dispozici základní funkce Api Gatewaye chybí tam některé funkce jako například Admin UI, zabezpečení a developer portal. Tyto funkce jsou přístupné jen s enterprise licencí.

Nasazení

Jednou z největších výhod Kongu je jeho široká škála možností instalace. S předem připravenými kontejnery v Dockeru a Vagrantu urychlí nasazení této API Gatewaye. NGINX je nejoblíbenější HTTP serverem teda až po Apache a IIS. Je také velmi výkonný při velkém množství požadavků. K úspěšnému nasazení Kongu je potřeba mít rozběhlé klastry Cassandra nebo Postgres. Některé pluginy jako je například plugin omezující rychlost vyžaduje optimálně další úložiště dat jako třeba Redis. Nasazení do produkce je jednodušší než třeba u Apigee.

Funkce

Kong poskytuje mnoho očekávaných základních funkcí k API managementu, jako je třeba vytváření API klíčů nebo směrování do více mikroslužeb. Kong nemá moc transformačních vrstev jako třeba SOAP nebo XML, má jen transformace založené na HTTP. Pokud však nemáte moc starších aplikací, pravděpodobně ještě potřebujete složitější transformaci dat. Administrační a managementové úkoly lze provádět pomocí příkazů CLI nebo curl do REST API, které usnadňuje management.

Kong má koncepty služeb, tras (cest) a zákazníků, které poskytují velkou flexibilitu při práci se stovkami mikroslužeb, které tvoří API. To umožňuje připojit pluginy a transformace ke konkrétní trase nebo dokonce ke konkrétnímu spotřebiteli. Kong má velkou komunitu, která vyvíjí různé pluginy. V roce 2018 spustili Kong Hub, který má již desítky aktivních pluginů.

Kong je jednou z vysoce doporučovaných API Gatewayí. Je moderní, určený pro správu moderních mikroslužeb. Stále se vyvíjí a zlepšuje. Má i rychle rostoucí množství pluginů díky API analytikům nebo ověřování pomoc JWT (JSON Web Tokens) [8].

3.2.3 Tyk

Stejně jako Kong je i Tyk open source, ale je pod licencí MPL, což je méně přípustná licence než Apache 2.0 u Kongu. Avšak enterprise uživatel Tyku používá stejnou gateway jako community uživatel. U Tyku není potřeba přidávat další pluginy nebo Lua skripta. Používá funkci jako třeba autentizace, u které podporuje OIDC, OAuth2, Bearer Token, Basic Auth, Mutual TLS, HMAC. Tyk také podporuje XML -> JSON, JSON -> XML a JSON Schema Validation.

Tyk je postaven na GoLangu, což je systémový jazyk určen pro vysokou propustnost a paralelismus. Společnost, která spravuje Tyk.io, poskytuje Cloud verzi a licence odborné podpory. Na rozdíl od Konga (NGINX) může být pro některé GoLang modernější nebo jednodušší pro programování. Díky podpoře Javascriptu a Lua má Tyk možnost spouštění pluginů v jiných jazycích.

Nasazení

Tyk nabízí řešení SaaS hostované v cloudu nebo on-premise. Instance můžete nasadit na Heroku nebo AWS. Open source verze je relativně jednoduchá na deploy (nasazení) a vyžaduje pouze Redis (databáze), zatímco Kong vyžaduje spuštění klastrů Cassandra nebo Postgres.

Funkce

Tyk má funkce jako Správa klíčů, Kvóty, limitování rychlosti, verze API, řízení přístupu, ale žádné integrované fakturační (billing) funkce. Tyk má jak REST API, tak webový dashboard pro provádění administrativních úkolů. Tyk nemá tak velkou komunitu nebo plugin hub jako má Kong. Avšak jejich API Gateway se snaží udržet dobře navrženou a tenkou [8].

3.2.4 Amazon AWS API Gateway

Amazon AWS, jako největší dodavatel cloudů, má také AWS API Gateway. Podporuje pouze cloud řešení. Pokud již využíváte AWS Lambdu nebo EC2, tak můžete použít AWS API Gateway, která bude ve stejném datovém centru jako jiné vaše upstream služby. To by mělo zamezit vysoké latenci.

Nasazení

Nasazení AWS API Gateway je snadné. Lze ji nasadit na portálu AWS několika kliknutími. Při použití ve spojení s AWS Lambda poskytuje AWS API Gateway příjemné řešení pro Serverless API (API bez serverů). Serverless je taková posílená mikroslužba, která vyžaduje nepřekonatelnou správu API endpointů směrováním příchozích API volání na příslušnou serverless funkci.

AWS API Gateway nabízí kromě AWS Lambdy nejlepší on-click řešení pro směrování příchozích API volání do jiných služeb AWS, jako jsou třeba služby Amazon Kinesis a Amazon DynamoDB. Kromě toho můžete použít svou stávající infrastrukturu IAM (Identity and Access Management) k ověřování do aplikačních rozhraních bez větší problémů.

Funkce

Amazon API Gateway je plně spravovaná služba, která cílovým vývojářům usnadňuje publikování, údržbu, monitorování, zabezpečení a provozování API v libovolném měřítku. Jedná se o službu pay-as-you-go (průběžné financování). Tato služba se stará o všechny API, aby bezpečně a spolehlivě běželi. Podporuje mimo jiné WebSocket a RESTful aplikační rozhraní a JWT autorizaci.

Tato API Gateway je to srovnatelná například s Kongem. AWS API Gateway však nemá tak rozsáhlou komunitu vývojářů, která by navrhovala a vytvářela nová rozšíření nebo pluginy. Jedním z největších problémů používání AWS API Gatewaye je Amazon lock in [8].

3.2.5 Wicked.haufe.io

Wicked poskytuje API Portal a API Management, jehož základ je postavený na API Gateway Kong od společnosti Mashape. Kong je podle nich jen komponenta, která konfiguruje pouze REST API. Wicked API portál usnadňuje používání Kongu a navíc nabízí více funkcí než API Gateway Kong.

Funkce

Jejich API Gateway využívá tedy Mashape Kong a poskytuje jim výkonnou API Gateway, díky který si můžete zabezpečit API. Pomocí Wicked portálu mohou vývojáři přistupovat ke svým API, ke kterým se dostanou přes API Gateway pomocí API klíčů nebo OAuth Credentials. V rámci API portálu lze dokumentovat API a to pomocí OpenAPI Specification neboli Swagger.

Nasazení

Wicked.haufe.io je Open Source API Gateway řešení. Pod licencí Apache 2.0 mají zpřístupněný zdrojový kód. Nejpřesvědčivější funkcí není to, co software dokáže, ale spíše to, jak jej lze nasadit. U většiny jiných API management řešení se snaží zakomponovat mnoho funkcí do svých API Gateway řešení. Wicked se zaměřuje především na kompatibilitu s jinými nástroji zejména na DevOps Support a na Docker Support. Díky DevOps podpoře lze implementovat do API managementu jakoukoli jinou aplikaci pomocí CI / CD (Continuous Integration/Continuous Deployment) Pipelane například Jenkins, Travis, GoCD. Podporuje také Phoenix Deployments nebo Blue / Green Deployments. Lze také uložit celé konfigurační řešení do API Managementu. Zpravidla chtějí, aby šlo všechno spouštět v kontejnerech, tj. pomocí Dockeru [22].

3.2.6 APIman.io

APIman.io je JBoss open source nástroj od společnosti Red Hat pro API management. Je k dispozici již několik let. V současné době se aktivně vyvíjí a je zpřístupněn na GitHubu. Software je postaven na dvou hlavních komponentách API Manager a API Gateway.

Funkce

API manager je komponenta, ve které probíhá většina API administrace a vytváření aplikačních rozhraní. Uživatelé si mohou definovat API a implementovat policies (politiky), které jsou základními stavebními bloky. Uži-

vatelé dále mohou procházet již existující API a porozumět jejich policies, která implementovali.

API Gateway je runtime (běhové prostředí), ve kterém jsou návrhy aplikačních rozhraních implementovány a vystaveny zákazníkům. Apiman podporuje mimo jiné použití více API Gatewayí, což umožňuje oddělení API na základě jejich charakteristik nebo potřeb. Poskytovatel API tedy může nasadit všechna svá veřejná API na jednu gateway a všechna svá interní soukromá API na jinou gateway například z důvodu jiné správy provozu nebo zabezpečení. O všechny brány včetně nově přidaných do konfigurace se postará API Manager.

Nasazení

Apiman poskytuje řadu způsobů, jak spustit instanci, včetně možnosti vložit vlastní modul politiky do aplikace nebo spustit Java application runtime. Apiman také podporuje Docker Deployments a díky tomu jsou kontejnery v Dockeru nejjednodušší metodou spuštění. Při spuštěném Docker kontejneru se lze přihlásit do API Manageru a začít vytvářet konfiguraci, která bude posléze nasazena do API Gatewaye. To, co ve své konfiguraci skutečně vytvoříte, je do značné míry určeno datovým modelem Apiman.

Stejně jako u většiny API management řešení implementuje i Apiman vlastní interní model, který poskytovatelům API pomáhá popsat jejich API, aby je poté mohli řídit a spravovat. Apiman model umožňuje prvkům opakované použití a také organizační kontroly. Organizations (organizace) popisují samotného poskytovatele API a jsou kontejnerem pro Plans (plány) a pro aplikační rozhraní, a také poskytují management kontext pro tyto entity. Plány jsou entitou správy, které jsou vytvořeny pro lepší organizaci aplikačních rozhraních a používají se k popisu chování, které lze aplikovat na API. Plány jsou tedy účinné opakovatelně použitelné objekty, které lze použít kdekoli v celé organizaci. Pokud se například poskytovatel API rozhodl implementovat společný plán pro všechna svá aplikační rozhraní, může tak učinit pomocí tohoto objektu. Plány proto zahrnují schopnost definovat politiky, jako jsou kvóty, autorizace, caching, whitelist atd.

Role API v Apiman modelu je zřejmá, ale stojí za zmínku, že poskytují spojení mezi organizací a klientskou aplikací. Existuje několik různých aspektů, díky kterým lze API zapouzdřit. API Implementation (API implementace) umožňuje definici aplikačního rozhraní se připojit k backendu založenému na REST nebo SOAP. API Definition (API definice) volitelně umožňuje ukládat OpenAPI Specification dokument do aplikačního rozhraní. Jak už je popsáno výše, API mohou odkazovat na jeden nebo více

API Plans definovanými pomocí API Organizací. Tím jsou více prosazovány politiky obsažené v plánech. Přímo v API můžete také implementovat politiky. Politiky jsou potřebné ve specificky se chovajících aplikačních rozhraních [23].

3.2.7 Fusio

Fusio je další open source systém pro management aplikačních rozhraní, který pomáhá budovat nejmodernější REST API. Umožňuje konstrukci API z různých typů dat. Fusio pomáhá vyvíjet skutečné API endpointy. Umí tedy transformovat vyžádaná data z databáze. Nemá žádná omezení na proxy požadavky k jiným API. Fusio používá backend služby k získání API response nebo ke zpracování request dat. Backendovou službou může být databáze, fronta zpráv nebo jiné aplikační rozhraní. Je také velmi snadné implementovat vlastní připojení podpory jakéhokoli druhu služby.

Aplikační rozhraní mohou používat aplikace, které jste vyvinuli vy sami nebo vývojáři třetích stran. Tyto aplikace mohou být například Javascriptové aplikace jako AngularJS nebo React, mobilní aplikace nebo desktop aplikace, které musí komunikovat s API. Fusio backend aplikace používá také ke konfiguraci systému interní API. Z tohoto důvodu je také velmi snadné integrovat Fusio do existujícího systému, protože ve skutečnosti můžete také nakonfigurovat celý systém pomocí API [16].

Funkce

Poskytuje funkce jako je autorizace OAuth2, omezení rychlosti, podpora OpenAPI, tvorba webhooků, generování sady SDK a monitorování životního cyklu. Příchozí požadavek se díky Fusio přiřadí ke vhodné trase (route). Trasa obsahuje všechny informace o příchozím požadavku a odchozích odpovědích. Tyto informace se používají také v dokumentaci, která je automaticky dostupná. Pokud bylo poskytnuto request schéma, příchozí request body bude po tomto schéma ověřeno. V případě, že je vše v pořádku, provede se action (akce) přiřazená k dané trase [15].

Akce představuje kód, který zpracovává příchozí požadavek a vytváří odpověď. Každá akce může použít toto připojení ke splnění svého úkolu. Připojení využívá knihovny, které usnadňují nebo pomáhají pracovat s remote (vzdálenou) službou. Například SQL připojení používá knihovnu Doctrine DBAL pro práci s databází. Připojení vždy vrací plně nakonfigurovaný objekt. Kromě toho již existuje mnoho různých akcí, které můžete použít. Třeba vytvořit aplikační rozhraní pomocí databázové tabulky.

Fusio se snaží mít co nejméně vrstev, aby zákazníci mohli ve svých akcích pracovat přímo s konkrétní knihovnou. Z tohoto důvodu nemá Fusio žádný svůj datový model nebo entitní systém. Místo toho třeba doporučují psát obyčejný SQL pro práci s relační databází. Domnívají se, že vytváření API koncových bodů na základě modelů nebo entit omezuje způsob, jakým lze navrhnout odpověď. Lze popsat požadavek a odpověď v JSON formátu. Ve Fusio můžete vytvářet kompletně přizpůsobené odpovědi na základě SQL dotazů, vkládáním message queue nebo pomocí HTTP calls [16].

Nasazení

Fusio poskytuje oficiální a aktuální verzi na GitHubu, kterou je zde volně ke stažení. Podporuje také Composer, Electron a Docker, ve kterém si můžete stáhnout docker kontejner nebo jednotlivá images. Fusio je pod licencí AFPLv3 (GNU Affero General Public License). Každý může kopírovat a distribuovat kopie, ale jeho změna není povolena [17] [18].

3.2.8 Repose

Repose je open source RESTful middleware platforma, která transparentně integruje s vaší stávající infrastrukturou. Repose poskytuje vysoce škálovatelná a rozšiřitelná řešení pro API management. Repose navíc umožňuje službám používat Enterprise integration Patterns. Repose přijímá příchozí požadavky od klientů, které posléze upravuje tak, aby byli připraveny ke spotřebě. Upravuje je prostřednictvím několika rozšiřitelných filtrů. Tyto filtry poskytují několik různých funkcích. Repose může běžet jako samostatný proxy server mezi klientem a origin službou. Repose instance může být na stejném hostiteli jako služba původu, ale nemusí to být.

Nasazení

Mají několik způsobů nasazení. Například Valve, ve kterém klient komunikuje s Repose a API Gateway komunikuje s origin službou. Dále lze Repose spustit v servlet kontejneru jako je GlassFish nebo Tomcat u WAR deploymentu. Repose umí provádět změny v konfiguraci za běhu, což usnadňuje samotnou konfiguraci a testování. Repose lze nakonfigurovat tak, aby používal službu distribuovaného datového úložiště, ve kterém jsou cached informace vyměňovány mezi více uzly. Díky tomu je Repose tolerantní k poruchám výkonu.

Funkce

Repose poskytuje sérii přizpůsobitelných filtrů, které můžete nakonfigurovat pro provádění velkého počtu API úkolů. Mezi tyto funkce patří například překlad požadavků a odpovědí tak, aby je služby dostávaly ve formátu, který očekávají. Možnost omezit rychlost požadavků a odpovědí, díky tomu lze kontrolovat a řídit provoz v síti. Pro zvýšení bezpečnosti Repose umožňuje ověřování požadavků. Ověření požadavku tak, aby vyhovoval XSD nebo WADL specifikacím. Logging pro uchovávání soupisů událostí, blacklisting zabraňuje podvodnému ověřování. Role-based access control (RBAC) může omezit přístupy uživatelů. Repose umožňuje mapování stavu vašeho aplikačního rozhraní. Repose je licencován pod Apache License Version 2.0 s volitelnou Saxon EE licencí [19] [20].

3.2.9 Gravitee.io

Řešení poskytnuté portálem Gravitee.io bylo vybráno jako vhodné řešení. Popis Gravitee.io API Gateway naleznete v sekci 4.

3.3 Výběr řešení

Na základě prezentace poznatků a výsledků společnosti AIMTEC a. s., se vedoucí bakalářské práce a jeho tým rozhodl pro Gravitee.io API Gateway řešení. Vyhovovalo nám nejvíce. Je to open source řešení s přehledným grafickým uživatelským rozhraním. Má možnost importovat aplikační rozhraní pomocí JSON a Swagger. Můžeme si sami definovat politiky nebo vytvářet nové funkce. Toto API Gateway řešení bude v další kapitole detailně popsáno.

4 Gravitee.io

Gravitee.io je výkonná a nákladově efektivní Open Source API platforma s jednoduchým použití v praxi, která pomáhá organizacím zabezpečit, publikovat a analyzovat jejich API.

V současné době API Management pomáhá organizacím publikovat jejich API externím vývojářům, partnerům a interním vývojářům s cílem uvolnit potenciál jejich dat a služeb. Dále pomáhá firmám přejít na OpenAPI a OpenData dynamiku. Společnosti tak mohou rozšířit svou činnost jako digitální platforma vytvořením nových komunikačních kanálů a získáním nových zákazníků.Poskytováním takových zařízení společnosti vidí rostoucí počet spotřebitelů a partnerů a musí čelit novým výzvám:

- Jak zkrátit čas na registraci nových zákazníků/partnerů
- Jak identifikovat partnery a spravovat jejich API spotřebu
- Jak měřit spotřebu z pohledu spotřebitele nebo výrobce
- Jak sdílet existující API
- Jak spravovat životní cyklus API, verzování, dokumentaci

Pak je otázkou jak tyto problémy vyřešit najednou. Proto potřebujete jeden centrální nástroj, který vyřeší tyto problémy pro všechna vaše API. Tím centrálním nástrojem je Gravitee.io API Management.

4.1 Gravitee.io API Management

Gravitee.io API Management je flexibilní, lehké a rychlé open source řešení, které pomáhá organizacím přesně kontrolovat, kdo, kdy a jak přistupuje k aplikačním rozhraním. Gravitee.io API Management je jednoduchý produkt, který funguje jako globální řešení pro API Management [26].

4.2 Proč vzniklo Gravitee.io?

Autoři Gravitee.io na začátku hledali open-source řešení pro správu jejich API. Žádné z již existujících řešením pro správu aplikačních rozhraní jim nevyhovoval nebo nesplňoval jejich potřeby. To je hlavní důvod, proč vzniklo Gravitee.io. Jejich cílem je poskytnout uživatelům vysoce integrovatelné

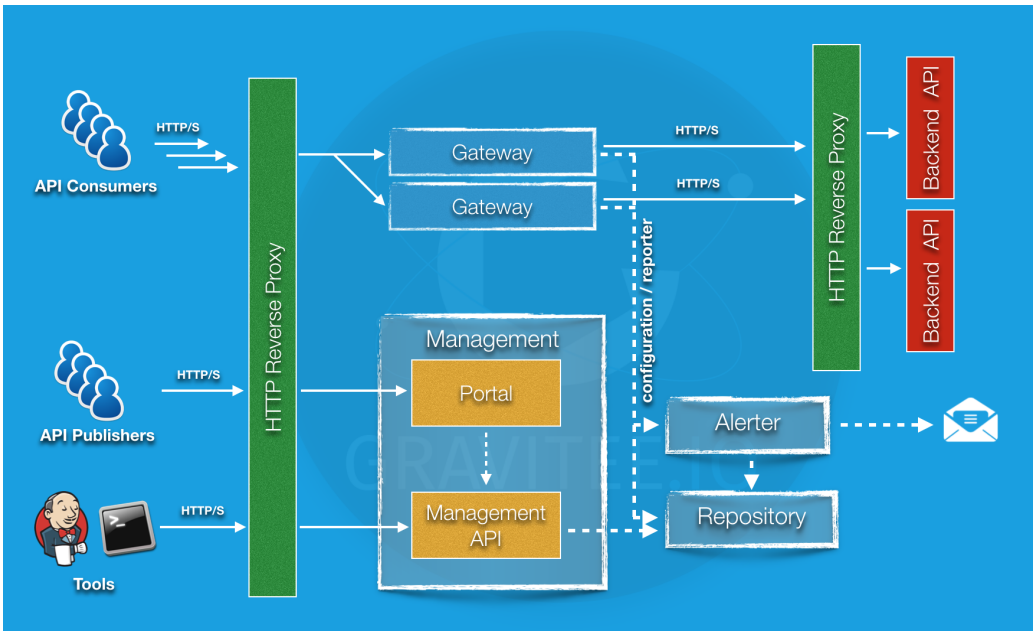
a škálovatelné řešení, které je schopno dokonale vyhovovat jejich obchodním požadavkům a infrastruktuře.

Gravitee.io bylo vyvinuto a navrženo tak, aby byl plně rozšiřitelný pomocí interního systému pluginů. Můžete si tedy implementovat jakýkoli plugin, který bude Gravitee.io podporovat. V podstatě si můžete dělat co chcete: definovat svou vlastní politiku, rozvíjet svůj vlastní systém hlášení a spoustu dalších věcí.

Všechny komponenty poskytované Gravitee.io (včetně brány a api managementu) opravdu jednoduchá na používaní. Gravitee.io má agresivní přístup k řízení CPU a paměti. Díky tomu má vysokou dostupnost [27].

4.3 Architektura

Na obrázku 4.1 vidíte architekturu Gravitee.io. Jsou na něm také vidět pojmy, kterým je třeba porozumět.



Obrázek 4.1: Architektura Gravitee.io [38]

Gravitee.io se skládá ze tří komponent, které má rozdělené i v dockeru do tří images. Jedná se o image Gravitee.io Gateway, Gravitee.io API Management a Gravitee.io UI. Pokud images budete startovat ručně, tak záleží na pořadí startování. Začněte s Gateway, pak API Management a poslední User Interface [38].

4.3.1 Pojmy

API

API je root pojem definovaný a používaný Gravitee.io. V podstatě je to výchozí bod pro vystavení služeb prostřednictvím brány.

Publisher

Publisher nebo taky vydavatel (API publisher) je jednou ze dvou základních rolí definovaných na platformě. Tato role se používá k zastupování někoho, kdo je schopen deklarovat a spravovat API.

Consumer

Consumer nebo taky spotřebitel (API consumer) je role definovaná pro spotřebu API. Spotřebu API lze provést až po přihlášení k tomuto API.

Aplikace

Aplikace je přechodná část mezi spotřebitelem a API. Spotřebitel ji používá k přihlášení k odběru před tím, než ji může spotřebovat [29].

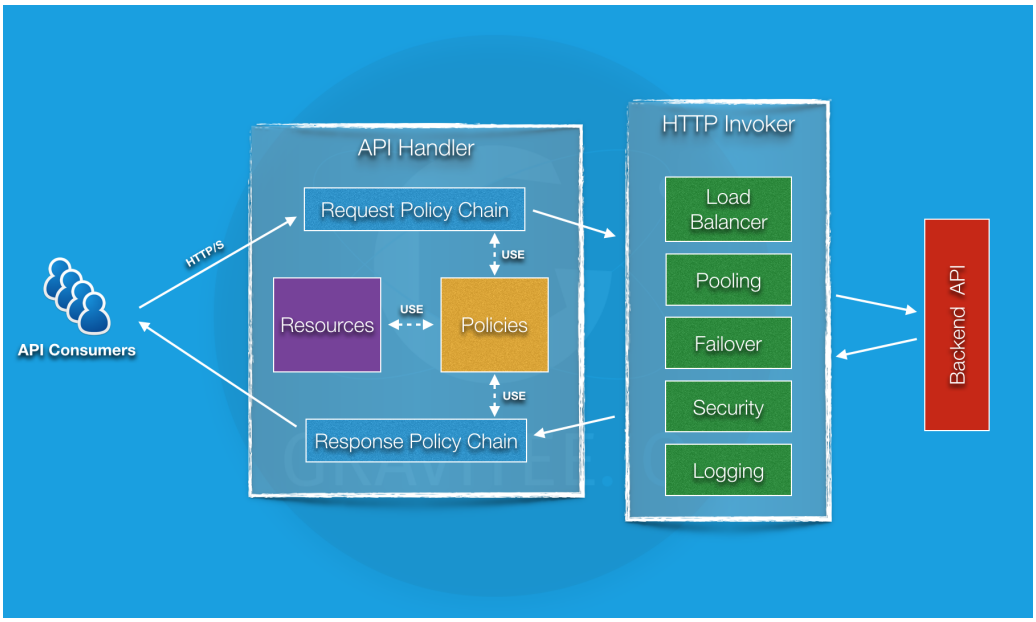
4.4 Komponenty

Další sekce popisuje komponenty, které jsou součástí Gravitee.io.

4.4.1 API Gateway

API Gateway je základní součástí platformy Gravitee.io. Můžete jej porovnat se smart proxy server, abyste pochopili její cíl.

Na rozdíl od tradičního HTTP proxy je gateway schopna aplikovat politiky (tj. pravidla) na HTTP požadavky a odpovědi podle potřeb. To znamená, že můžete vylepšit zpracování požadavků a odpovědí přidáním jiné transformace nebo zabezpečení a mnoha dalších funkcí.



Obrázek 4.2: Gravitee.io API Gateway [30]

4.4.2 Management API

Restful API poskytuje spoustu služeb pro správu a konfiguraci globální platformy. Všechny služby jsou omezeny autentizací a autorizací.

4.4.3 Management UI

Tato komponenta slouží jako webové uživatelské rozhraní pro Gravitee.io Management API. Toto uživatelské rozhraní funguje jako nástroj pro správu API Publishers (vydavatelů).

4.4.4 Portál

Webové uživatelské rozhraní pro API Cnsumers (konzumenty), ve kterém mohou vyhledávat, hledat, zkoušet nebo se přihlásit k odběru API [30].

4.5 Pluginy

Pluginy jsou další komponenty, které lze připojit do API brány nebo Management API. Pluginy lze upravit podle vašich specializacích, tak aby vyhovovali přesně vašim potřebám nebo technickým požadavkům. Sledují danou konvenci adresářové struktury.

Při používání Gravitee.io se setkáte s několika typy pluginů, tyto typy vidíte v tabulce 4.1

4.5.1 Politiky

Politika se chová jako požadavek nebo odpověď zpracovaná v bráně. Může být zřetězena řetězcem politiky požadavků nebo řetězcem politiky odpovědi pomocí logického pořadí. Politiky mohou být považovány za proxy controller a tím zaručí, že se dané obchodní pravidlo během zpracování požadavku nebo odpovědi splní.

Dobrým příkladem politiky jsou:

- Autorizace pomocí klíče API (api-key policy)
- Použití transformaci parametrů hlavičky (header) nebo dotazu (query)
- Použití omezení rychlosti nebo kvóty, aby se zabránilo zaplavení API.

4.5.2 Reportéři

Reportér je používán instancí brány k hlášení mnoha typů událostí:

- Metriky požadavků/odpovědí: doba odpovědi, délka obsahu, api klíč
- Monitorování metriky: CPU, využití haldy
- Metriky kontroly stavu: stav, kód odpovědi

Reportéři „out of the box“ jsou:

- Elasticsearch reportér
- Souborový reportér

Vlastní reportéry si můžete vytvořit, používat a nasazovat stejně jako u pluginu.

4.5.3 Repozitáře

Komponenta zásuvného úložiště pro konfigurace API, konfigurace politik, analytik atd.

Tabulka 4.1: Plugins [31]

Poskytovatelé identity	
Komponenta:	Management API
Příklad:	LDAP, Oauth2, InMemory

Sběratelé	
Komponenta:	Management API
Příklad:	HTTP, GIT

Politiky	
Komponenta:	Management API / Gateway
Příklad:	API Key, Rate-limiting, Cache

Reportéři	
Komponenta:	Management API
Příklad:	Elasticsearch, Accesslog

Repozitáře	
Komponenta:	Management API / Gateway
Příklad:	MongoDB, Redis, Elasticsearch

Zdroje	
Komponenta:	Management API / Gateway
Příklad:	Oauth2, Cache, LDAP

Služby	
Komponenta:	Management API / Gateway
Příklad:	Sync, local-registry, health-check, monitor

Oznamovatelé	
Komponenta:	Alert Engine
Příklad:	Email

Upozornění	
Komponenta:	Management API / Gateway
Příklad:	Gateway
Vertx	

4.5.4 Notifiers (Oznamovatelé)

K zasílání oznámení se používá oznamovatel. Prozatím jediným dostupným oznamovatelem je e-mail, ale brzy se plánují další.

4.5.5 Upozornění

Upozornění umožňuje odeslat spoušť nebo událost do Alert Engine, který bude zpracován pro odeslání oznámení s konfigurovaným oznamovatelem pluginu. Za konfiguraci oznamovatele odpovídá spoušť [31].

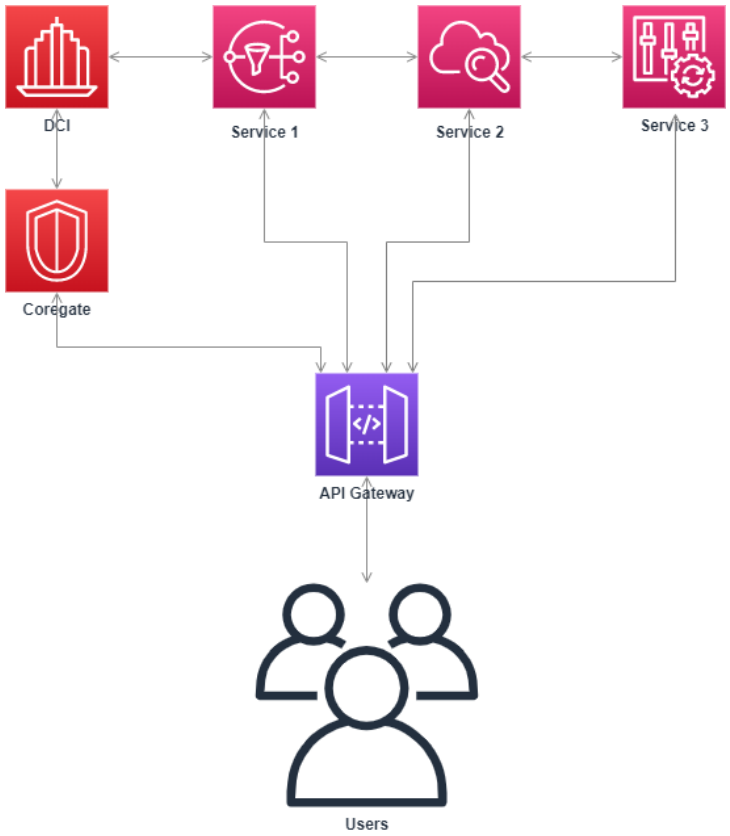
4.6 Cloud a On-Premise řešení

Gravitee umožňuje jak on premise tedy lokální řešení tak i cloud tedy hostované řešení. Dále také podporuje hybridní řešení, což je mix mezi on premise řešením a cloudem. Jestliže váš systém běží na hybridní architektuře, existují určitá omezení, která se mohou týkat technického omezení nebo omezení nákladů. Ty mohou bránit v rozmístění všech komponent, která Gravitee API Management vyžaduje v každém datovém centru. Gravitee.io nabízí hybridní komponenty (ve formě pluginů), které vám pomohou s definicí vaší architektury a vaší deployment vizi.

4.7 Gravitee a Aimtec

4.7.1 K čemu je firmě Aimtec API Gateway?

API Gateway je užitečná především pro Aimtec, který díky ní může zákazníkovi (v určitých případech) nabídnout rychlé úpravy bez nutnosti něco programovat v Javě. Stačí vytvořit definic API pro Gravitee. Díky Gravitee.io API Gateway vidí zákazník na Gravitee User Interface a díky tomu může přistupovat ke svým aplikačním rozhraním. Na obrázku 4.3 vidíte příklad jak může fungovat API Gateway. Šipky znázorňují možnou komunikaci. DCI zde představuje existující skladovací systém, Service 1 - 3 jsou nějaké služby. V tomto případě se může jedna o API Gateway Gravitee.io.



Obrázek 4.3: API Gateway

Což znamená, že má přístup ke svým datům a může si libovolně upravovat svá data například změnit cenu položky, přidat nové položky do systému atd. Dále si může vytvářet nadstavby pro své vlastní potřeby. Rest API se stará o komunikaci se systémem DCI firmy AIMTEC a.s. Pomocí aplikačního rozhraní Data Integrátor lze dostat data do DCI systému nebo z něj data vybrat. Jednotlivá aplikační rozhraní jsou zdokumentovaná pomocí Swaggeru. Pro dobré zabezpečení Aimtec využívá dvě autorizační API.

4.7.2 Cloud a On-Premise

Gravitee.io on-premise funguje dobře. Primárně se počítá, že většina zákazníků bude chtít on-premise řešení. Aimtec ale ani s Cloudem nezaostává, a vytvořili si svůj vlastní AimtecCloud, který je implementovaný, tak aby s Gravitee spolupracoval a komunikoval správně.

Zákazník si tedy může vybrat, jestli chce Cloud nebo On Premise řešení. Přičemž při výběru jakéhokoli řešení se požadují minimálně čtyři nebo pět serverů. Záleží samozřejmě na zákazníkovi a na jeho potřebách. Optimálně

by na jednom serveru běžela databáze a na třech serverech by běželo Kubernetes.

Rozdíl

Při instalaci on premise se nejprve na servery zákazníka nainstaluje kubernetes infrastruktura a až pak do ní DCI.

Při instalaci do Cloudu se použije Kubernetes infrastruktura příslušného Cloudu (je vyzkoušen Amazon EKS) a instalace DCI je už pak téměř totožná.

4.8 Jiné reprezentace API

Aplikační rozhraní lze vytvořit pomocí externího importu. API může být reprezentováno formátem pro přenos dat tedy například json objektem.

4.8.1 JSON

JSON neboli JavaScript Object Notation je formát sloužící hlavně pro výměnu dat. Je čitelný, lehce zapisovatelný a relativně snadno analyzovatelný. Lze ho i generovat automaticky. Vznikl z programovacího jazyka JavaScript a ECMA-262. JSON je jazykově nezávislý textový formát. Je kompatibilní s programovacími jazyky z rodiny C (C, C++, C#, Java, JavaScript, Perl, Python a dalších). Proto se tento formát hodí pro výměnu dat.

JSON je založen na dvou strukturách:

- Kolekce párů název/hodnota
- Seřazený seznam hodnot

Kolekce párů název/hodnota

Tato kolekce bývá většinou realizována jako objekt, struktura, slovník, hash tabulka, klíčový seznam nebo asociativní pole.

Seřazený seznam

Tento seznam je většinou realizován jako pole, seznam, vektor nebo posloupnost.

JSON je univerzální datová struktura, kterou ve své podstatě podporují skoro všechny moderní programovací jazyky. Z toho vyplývá, že by měl existovat nějaký nezávislý výměnný formát, který je na této datové struktury založen.

V JSON jsou struktury realizovány těmito konstrukcemi:

- Objekt (neuspořádaná množina párů název/hodnota)

```
{
    "name": "name",
    "object": {}
}
```

- Pole (seřazená kolekce hodnot)

```
{
    "array": []
}
```

- Klíč/Hodnota (řetězce ve dvojitých uvozovkách)

```
{
    "key": "value"
}
```

Hodnotou se rozumí číslo, string, true, false, null, objekt nebo pole. Tyto struktury mohou být vnořovány.

Řetězec je také uzavřených do dvojitých uvozovek. Může být nula nebo více znaků kódování Unicode. Lze u něj využít únikovou sekvenci (escape sequence) s použitím zpětného lomítka. Dále se také mohou používat bíle znaky (whitespace).

Obrázek 4.4: API jako JSON Tree

Na obrázku 4.4 vidíte json objekt v tree podobě. Nejdůležitějšími informacemi nebo parametry, tedy potřebnými (required) ke správné funkčnosti aplikačního rozhraní, jsou *name, version, paths, plans, path mappings a proxy*. Objekt *proxy* obsahuje pole *virtual hosts*, ve kterém se nachází objekt obsahující parameter *path*. Tento parametr musí být vždycky originální, protože je to endpoint, který se volá.

Každé aktivní aplikační rozhraní musí mít nastavený aktivní plán. Tento plán, jak vidíte na obrázku 4.5, má nastavený status na *PUBLISHED* (tzn. je publikovaný).

Obrázek 4.5: Aktivní veřejný plán

4.9 Jolt transformace

Jolt neboli JsOn Language for Transform je transformační knihovna napsaná v Javě, která umožňuje vývojářům převést jednu JSON strukturu na jinou. Jolt sám o sobě poskytuje sadu typů transformací, z nichž každá má své vlastní DSL (specifikace), které definují novou strukturu pro odchozí JSON data.

JSON je jazykově nezávislý datový formát, který se běžně používá pro komunikaci mezi klientem a serverem, zejména pokud jde o webové služby, konfigurační soubory, jiné deskriptory dat. Příchozí JSON data musí být pro použití jednoduše formátována nebo re-labeled (znovu označena), tak aby mohla být použita v jiném systému nebo úložišti, jako je Hive, HBase, MongoDB nebo Elasticsearch. JSON musí získat správný Output než projde Joltem. Což znamená, že JSON Input musí projít různou přeměnou.

JSON Input -> XML -> XSLT / STX -> XML -> JSON Output

Potom už stačí jen formátovat JSON.

JSON Input -> spec.json -> JSON Output

4.9.1 Výhody Joltu

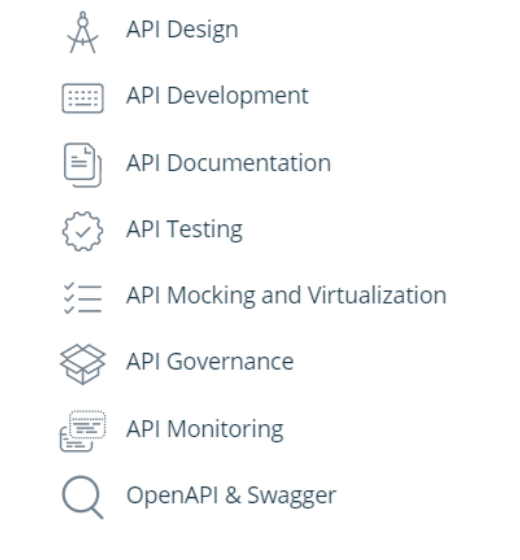
Minimalizuje změny Java kódu, protože většina práce se provádí pomocí transformací. Jakákoli komplexní transformační logika, kterou nelze vyjádřit standardně, lze připojit pomocí rozšíření v Javě. Jolt se vyhýbá se přeměnění jednoho objektového modelu do jiného. Díky Joltu odpadnou dvě transformace (XML -> XSLT / STX -> XML).

Je to v podstatě kompozice aplikačního rozhraní, ve které klient zavolá jedno API a Gravitee udělá několik transformací odpovědi. Vzápětí vybere tu správnou, a to je ta, kterou chce zákazník.

4.10 SWAGGER

Pole *pages* z předchozího obrázku u JSON 4.4 obsahuje swagger objekt, který je napsaný ručně pomocí swagger kukátka a obsahuje informace o API.

Swagger zjednodušuje vývoj aplikačního rozhraní pro uživatele, týmy a podniky. Je to sada Open Source nástrojů, díky kterým lze lehce navrhnout a zdokumentovat API v jakémkoli měřítku.



Obrázek 4.6: Nástroje Swaggeru

Pomocí všech dostupných nástrojů lze lehce vytvořit aplikační rozhraní dle představ společností. Jak vidíte na obrázku 4.6 API nebude chybět jediná komponenta.

4.11 Nastavení

Dále jsem dostal za úkol navrhnout a implementovat nastavení API brány, které by bylo importovatelné pomocí datové struktury json. Jedná se o nastavení, které se dá provést i manuálně, tedy klikáním. Myšlenka však směřovala k automatizaci.

Proběhla tedy analýza nastavení API brány. Po úspěšné analýze jsem navrhl strukturu jsonu a nastavil na optimální (výchozí) nastavení.

Obrázek 4.7: Struktura json objektu pro nastavení

Dle obrázku 4.7 se nastavení skládá ze 7 objektů. Objekt *Company* obsahuje jméno firmy, *Management* se skládá z názvu API Gateway a bezpečnostních plánů. Bezpečnostní plány určují zabezpečovací plány brány, jako třeba jwt, oauth2 a api-key plán. Dále *Portal* určuje název portálu a nastavuje třeba aktivitu google analytik nebo aktivuje podporu nebo hodnocení. *Theme* nastaví logo a popřípadě gif pro načítání, *Schedulers* určuje notifikace, *Documentation* dokumentaci a *Api quality* se stará o zpětnou kontrolu API.

Díky této analýze nastavení může AIMTEC vytvářet nová upravená nastavení jednotlivých aplikačních rozhraní v JSON formátu, které lze lehce importovat do platformy Gravitee.io API Gateway. Toto nastavení bylo testováno na Gravitee.io API Gateway.

5 Konfigurační nástroj

Jednou z podmínek, kterou vybraná API Gateway měla mít, je definice API pomocí YAML nebo JSON. Tato podmínka totiž usnadní práci s API definicí pomocí externího importování API. Každý jednotlivý zákazník bude mít více než jedno aplikační rozhraní, a proto vznikla myšlenka mít toto importování API automatické přes nástroj implementovaný v programovacím jazyce Java.

5.1 Návrh

Cílem je vytvořit nástroj, který při první instalaci nakonfiguruje Gravitee.io a naimportuje všechny json soubory, ve kterých je API definice, ze zadaného adresáře.

Zákazník většinou dostane dvě základní API. Prvním a důležitým aplikačním rozhraním je *data integrátor API*, který zajišťuje získání dat ze systému a odesílání dat do systému. Druhé aplikační rozhraní je *autorizační API*, které zajistí přístup do gravitee.

K dispozici jsem měl napsaný SWAGGER, který se pak vložil do jsonu. Gravitee.io umožňuje vytváření API buď klikáním, tedy online na webovém prohlížeči a nebo pomocí importu swaggeru nebo jsonu.

Implementace konfiguračního nástroje, který na základě vstupního parametru, tedy adresáře, ve kterém se nachází JSON API definice, vytvoří nebo aktualizuje již existující API. K přístupu do Gravitee.io je potřeba unikátní přihlašovací jméno, heslo a endpoint pro Gravitee API Management. Tyto tři parametry musí být nastaveny. Lze je nastavit v operačních systémech v proměnném prostředí nebo přiložený souboru *.env* to nastaví za vás bez námahy.

Nástroj je konfigurovatelný:

- vstupní parametry (adresář s json soubory)
- environmentální proměnné pro Gravitee.io
  - Management API endpoint
  - Username
  - Password

Upload API pomocí externích json souborů funguje na soubory v UTF-8 i UTF-8 s BOM.

5.2 Implementace

Konfigurační nástroj je rozdělen do tří tříd. První třída *FileManager* zajistí načtení souborů, druhá třída *Configurator* spouští nástroj a třetí třída *GraviteeManagement* se stará o manipulaci s API.

Spuštěná aplikace informuje uživatele o průběhu nástroje pomocí loggeru, který postupně podává informace o úspěšném nebo neúspěšném průběhu nástroje.

Nástroj dále zjistí, jestli již dané API existuje. Zavolá metodu *getApis()*, která uloží všechny API a jejich API ID do HashMapy. Pokud API již existuje se zjistí pomocí API parametru *path*, které je vždycky originální. Takže se porovnají API *path* parametry a pokud se schodují, tak jsou API stejná. V tom případě si uloží jeho API ID. Nejdříve vybrané API zastaví, pak ho reimportuje pomocí nového json souboru, poté ho nasadí a na závěr to API nastartuje.

5.2.1 FileManager

Tato třída slouží k načtení všech souborů .json v adresáři, který je zadaný jako parametr. Obsahuje metodu *findJsonFilesInDir()*, která projde celý adresář a uloží si ke všem json souborů jejich cestu. Metoda *readFile()* načte jednotlivé soubory. Tato metoda *removeUTF8BOM()* odstraní BOM (Byte Order Mark) ze začátku json souboru, který pak zabraňuje správnému deserializování jsonu.

5.2.2 GraviteeManagement

GraviteeManagement zajišťuje správné manipulovaní s příslušným aplikačním rozhraním. Nejprve pomocí metody *getApis()* získá všechny již existující aplikační programovací rozhraní. Metoda *executePost()* odešle post request na Gravitee.io a vrátí jeho odpověď (response). K úspěšném odeslání požadavku je potřeba autorizace. Údaje k přihlášení do Gravitee.io jsou nastaveny v proměnných prostředích (environment variables) v počítači a metoda *graviteeAuth()* nastaví požadavku hlavičku s autorizací do Gravitee.io.

Příklad endpointů pro práci s API:

http://localhost/management/apis/57329a16-147d-4f56-b29a-16147d5f56e8

Import API
/import
Start API
?action=START
Deploy API
/deploy
Stop API
?action=STOP
Reimport API Pro úspěšné reimportování aplikačního rozhraní se musí nejdříve dané API zastavit (Stop API), pak importovat (volá import nad svým vlastním API ID), poté se API nasadí (Deploy API) a nastartuje se (Start API).

Po importování nového API se musí dané API nastartovat (Start API), aby běželo.

kde *57329a16-147d-4f56-b29a-16147d5f56e8* je ID daného API

5.2.3 Configurator

Configurator je hlavní třídou konfiguračního nástroje. Obsahuje tedy hlavní metodu *Main()* a také metody pro ověření vstupních parametrů - adresáře a environment variables. Dále také ověřuje odpověď (response) na POST nebo GET požadavek (request) Gravitee.io. Metoda *readAndSendJsonConfigs()* se stará o získávání jednotlivých souborů, jejich následnou úpravu a odeslání požadavku na Gravitee.io. Také porovnává existující aplikační rozhraní s nově importujícími a díky tomu určí jestli se jedná o import nebo reimport.

5.3 Dokumentace, verzování a komentáře

K projektu je vytvořený README.md a je v rootu repozitáře. Verzování proběhlo v soukromém gitu Aimtecu. Postup práce a řešení se zaznamenávalo do interní Jiry. Komentáře byly napsány primárně v anglickém jazyce, avšak vznikla i česká verze.

6 Testování

6.1 API Gateway

Většinu vybraných API Gateway řešení jsem si lokálně stáhl a nainstaloval. Testoval jsem nasazení v software Docker. Dále jsem analyzoval evidenci a správu aplikačních programovacích rozhraní. Také jsem zjišťoval jejich funkce, nástroje, možnosti vytváření jednotlivých API, pluginy, politiky atd. Brána byla testována navržením a vytvořením několika API, které běžely současně, ale za různých podmínek.

Dále bych navrhoval, aby bylo Gravitee testováno pomocí E2E testování. End-to-end testování testuje kus softwaru od začátku do konce.

6.2 Konfigurační nástroj

Testování nástroje proběhlo na operačním systému Windows s VM Virtuálním Boxem, kde byl nainstalován systém Kubernetes a na operačním systému Linux s Kubernetes. Bylo navrhnuto několik aplikačních rozhraních reprezentováno v datové struktuře json. Tyto soubory byly vytvářeny různými nástroji a s různým kódováním, aby odhalily co nejvíce chyb.

Konfigurační nástroj byl dále testován pomocí Unit testů. Unit testy testují malé izolované části kódu. Proběhly zde i integrační testy, které představují způsob testování, ve kterém se kombinují jednotlivé části. Ty se pak testují společně jako jeden celek.

Díky testování se odhalily chyby a výjimky. Testy například odhalily nedostatky v ošetření Null Pointer výjimek, v závislosti souborů json na znakovém kódování nebo třeba v deserializaci json. Chybu v json deserializaci, díky které nešla importovat aplikační programovací rozhraní, způsoboval BOM. BOM je unicode znak (hexadecimálně FEFF), který určuje pořadí ukládání bajtů. Také existuje v kódovém značení UTF-8, UTF-16 a UTF-32 [33].

6.3 Možná rozšíření práce

Konfigurační nástroj lze vylepšit pomoc ConfigMap. Konfigurační mapa by vylepšila automatizaci API definice. ConfigMap je jeden koncept konfigurace v kubernetes. Kubernetes je Open Source systém pro automatizaci nasazení, škálování a správu kontejnerových aplikací. Seskupuje kontejnery,

které tvoří aplikaci, do logických jednotek pro snadnou správu a vyhledávání. ConfigMap umožňuje oddělit konfiguraci specifickou pro prostředí od container images, takže aplikace jsou snadno přenosné. Díky tomu by nástroj neprohledával celý adresář ale jen kubernetes, kde by našel určitý label. API definice by přicházeli rovnou se službami.

Také by šlo přidat ošetření private API. Pokud API již existuje a je private, tak po reimportování z něj může vzniknout public API. Při získávání existujících aplikačních rozhraní si uložit informaci o přístupnosti API. Díky analýze API Gateway nastavení lze vytvořit soubor, který reprezentuje nastavení v JSON formátu. Tento by po vložení do API brány nastavil nebo přenastavil dané API.

7 Závěr

V této práci byly popsány různé existující API Gateway řešení. Na základě specifikací bylo vybráno právě jedno řešení, které splňovalo všechny podmínky. Jedná se o API Gateway Gravitee.io. Výběru vhodného řešení předcházel teoretický rozbor, který se zabýval jak analýzou trhu existujících API Gateway řešení, tak i analýzou komponent. Na závěr této sekce byla popsána architektura a kompozice vybraného řešení.

Praktická část se věnovala návrhu aplikačních programovacích rozhraní a konfiguračního nástroje, který byl následně implementován. Výsledné API vylepší komunikaci mezi existujícím systémem firmy AIMTEC, a. s. a zákazníkem. Zákazníkům se zlepšil přístup ke svým datům. Také si mohou navrhnout nadstavbu nebo rozšíření pro své vlastní potřeby a zvolit typ nasazení řešení. Díky lepšímu přístupu k datům můžou data upravovat, například měnit ceny, přidávat nebo upravovat položky v systému. Pomocí implementovaného API Data Integrátor lze dostat data do DCI systému nebo z něj data vybrat.

Konfigurační nástroj automaticky importuje nebo aktualizuje všechny API nacházející se v dané API bráně. Tento nástroj společnosti a jejím zákazníkům ušetří čas s ručním vytvářením nebo importováním API. Na závěr je implementované API otestováno a jsou zde navrhnuty náměty pro případná další rozšíření nebo vylepšení.

V příloze se nachází popis a návod na vytvoření aplikačního programovacího rozhraní na platformě Gravitee.io pomocí jejich uživatelského rozhraní.

Seznam zkratek

- **API** Application Programming Interface - rozhraní pro programování aplikací
- **YAML** Ain't Markup Language - formát pro serializaci strukturovaných dat
- **JSON** JavaScript Object Notation - formát určený pro přenos nebo výměnu dat
- **HTTP** Hypertext Transfer Protocol - internetový protokol určený pro komunikaci s WWW servery.
- **REST** Representational State Transfer - architektura rozhraní navržená pro distribuované prostředí
- **SDLC** Systems Development Life Cycle - životní cyklus informačního systému
- **XML** eXtensible Markup Language - rozšiřitelný značkovací jazyk, který je navrhnutý na uchovávání strukturovaná data
- **SOAP** Simple Object Access Protocol - protokol sloužící k výměně zpráv založených na XML přes síť pomocí HTTP
- **TCP** Transmission Control Protocol - nejpoužívanějším protokolem transportní vrstvy v sadě protokolů TCP/IP
- **IP** Internet Protocol - základní protokol pracující na síťové vrstvě
- **TCO** Total Cost of Ownership - zkratka pro celkové náklady spojené s vlastnictvím
- **SOA** Service Oriented Architecture - sada principů a metodologií
- **UI** User interface - uživatelské rozhraní
- **CLI** Command Line Interface - uživatelské rozhraní, ve kterém uživatel komunikuje s programy nebo operačním systémem
- **JWT** JSON Web Token - standard, který definuje, jak předávat data mezi dvěma systémy

- **OIDC** OpenID Connect - protokol v autentizační vrstvě
- **TLS** Transport Layer Security - kryptografický protokol poskytující možnost zabezpečené komunikace na Internetu
- **HMAC** Hash-based Message Authentication Code - typ autentizačního kódu zprávy počítané s použitím kryptografické hašovací funkce v kombinaci s tajným šifrovacím klíčem
- **AWS** Amazon Web Services
- **EC2** Amazon Elastic Compute Cloud - webová služba, která poskytuje zabezpečení v Amazonu
- **IAM** Identity and Access Management - funkce Amazonu, zabezpečení uživatelů a rolí
- **SQL** Structured Query Language - strukturovaný dotazovací jazyk používaný pro práci s daty v relačních databázích
- **GNU** GNU's Not Unix - počítačový operační systém
- **XSD** Schema - rozšiřitelný značkovací jazyk, který popisuje přípustný obsah dokumentu
- **WADL** Web Application Description Language - XML popis webových služeb založených na
- **RBAC** Role-Based Access Control - přiřazuje práva a pomáhá je spravovat
- **CPU** Central Processing Unit - centrální procesorová jednotka (procesor)
- **LDAP** Lightweight Directory Access Protocol - protokol pro ukládání a přístup k datům na adresářovém serveru
- **URL** Uniform Resource Locator - řetězec znaků s definovanou strukturou, který slouží k přesné specifikaci umístění zdrojů informací na Internetu
- **DCI** - Skladovací systém, produkt společnost AIMTEC a.s.
- **EKS** Managed Kubernetes Service - funkce Amazonu, která pomáhá se správou Kubernetes

- **SWAGGER** - framework pro návrh, tvorbu, dokumentaci a konzumaci RESTful web API
- **UTF** Unicode Transformation Format - způsob kódování znaků
- **BOM** Byte order mark - znak hexadecimálně zapsaný jako FEFF
- **VM** Virtual Machine - vitruální stroj (software), který vytváří virtualizované prostředí mezi platformou počítače a operačním systémem
- **ECMA-262** - skriptovací jazyk
- **XSLT** Extensible Stylesheet Language Transformations - slouží k převodům zdrojových dat ve formátu XML do libovolného jiného požadovaného formátu
- **E2E** End-to-End - testování celého systému a pracovní postup

Literatura

[1] What is an API? (APPLICATION PROGRAMMING INTERFACE) [online]. MuleSoft [cit. 2019/11/19] Dostupné z: https://www.mulesoft.com/resources/api/what-is-an-api

[2] CO JE TO API (APPLICATION PROGRAMMING INTERFACE) [online]. TOPRANKER.CZ [cit. 2019/11/19] Dostupné z: https://topranker.cz/slovnik/co-je-to-api-application-programming-interface/

[3] What is a Gateway and What Does it Do? [online]. WhatIsMyIPAddress.com, [cit. 2019/11/20]. Dostupné z: https://whatismyipaddress.com/gateway

[4] Co je výchozí brána [online]. Správa Sítě, [cit. 2019/11/20]. Dostupné z: https://www.sprava-site.eu/vychozi-brana/

[5] Chris Richardson Pattern: API Gateway / Backends for Frontends [online]. Microservice Architecture, [cit. 2019/11/21]. Dostupné z: https://microservices.io/patterns/apigateway.html

[6] API Gateway [online]. NGINX, [cit. 2019/11/21]. Dostupné z: https://www.nginx.com/learn/api-gateway/

[7] Margaret Rouse API gateway [online]. TECHtarget, [cit. 2020/07/15]. Dostupné z: https://whatis.techtarget.com/definition/API-gateway-application-programming-interface-gateway

[8] Xing Wang How to choose the right API Gateway for your platform: Comparison of Kong, Tyk, Apigee, and alternatives [online]. moesiF, 06. března 2019 [cit. 2020/01/20]. Dostupné z: https://www.moesif.com/blog/technical/api-gateways/How-to-Choose-The-Right-API-Gateway-For-Your-Platform-Comparison-Of-Kong-Tyk-Apigee-And-Alternatives/#

[9] Kubernetes [online]. Kubernetes [cit. 2020/06/20]. Dostupné z: https://kubernetes.io/

[10] ConfigMaps [online]. kubernetes [cit. 2020/06/20]. Dostupné z: https://kubernetes.io/docs/concepts/configuration/configmap/

[11] CLOUD VS ON-PREMISE: STRUČNÁ PŘÍRUČKA [online]. globema, 13. prosinec 2017 [cit. 2019/11/24]. Dostupné z: https://www.globema.cz/cloud-vs-premise-strucna-prirucka/

[12] Marek Běhálek Kapitola 8. XML, Stručný úvod do XML [online]. Katedra informatiky VŠB-TU Ostrava, 2017 [cit. 2020/06/15]. Dostupné z: http://www.cs.vsb.cz/behalek/vyuka/pcsharp/text/ch08s01.html

[13] Úvod do JSON [online]. json.org [cit. 2020/06/15]. Dostupné z: https://www.json.org/json-cz.html

[14] O Drupalu [online]. Drupal.cz [cit. 2020/06/24]. Dostupné z: https://www.drupal.cz/o-drupalu

[15] Christoph Kappestein Laravel API Management system [online]. Medium, 9. únor 2020 [cit. 2020/07/10]. Dostupné z: https://medium.com/@christoph.kappestein/laravel-api-management-system-947921c7e11f

[16] Fusio - About [online]. Fusio [cit. 2020/07/10]. Dostupné z: https://www.fusio-project.org/about

[17] Fusio - Download [online]. Fusio [cit. 2020/07/10]. Dostupné z: https://www.fusio-project.org/download

[18] GNU GENERAL PUBLIC LICENSE [online]. GNU [cit. 2020/07/10]. Dostupné z: https://www.gnu.org/licenses/gpl-3.0.html

[19] REPOSE [online]. REPOSE, 2019 [cit. 2020/07/10]. Dostupné z: http://www.openrepose.org/

[20] REPOSE ATLASSIAN [online]. REPOSE [cit. 2020/07/10]. Dostupné z: https://repose.atlassian.net/wiki/spaces/REPOSE/overview?homepageId=525994

[21] Amazon API Gateway Features [online]. aws [cit. 2020/07/11]. Dostupné z: https://aws.amazon.com/api-gateway/features/

[22] Introducing wicked.haufe.io [online]. HAUFE.Group [cit. 2020/07/11]. Dostupné z: http://work.haufegroup.io/introducing-wicked-haufe-io/

[23] Chris Wood Review of Red Hat's Apiman Open Source API Management [online]. NORDIC APIS, 13. května 2016 [cit. 2020/07/11]. Dostupné z: https://nordicapis.com/apiman-red-hat-review/

[24] Gravitee.io API Platform [online]. Gravitee.io API Platform [cit. 2020/06/24]. Dostupné z: https://docs.gravitee.io/apim/1.x/apim_installguide_hybrid_deployment.html#introduction

[25] Publish your first API [online]. Gravitee.io API Platform [cit. 2020/06/24]. Dostupné z:https://docs.gravitee.io/apim/1.x/apim_quickstart_publish.html#start_your_api

[26] Introduction [online]. Gravitee.io API Platform [cit. 2020/06/24]. Dostupné z: https://docs.gravitee.io/apim/1.x/apim_overview_introduction.html

[27] Why Gravitee.io ? [online]. Gravitee.io API Platform [cit. 2020/06/24]. Dostupné z: https://docs.gravitee.io/apim/1.x/apim_overview_why.html

[28] Architecture [online]. Gravitee.io API Platform [cit. 2020/06/24]. Dostupné z: https://docs.gravitee.io/apim/1.x/apim_overview_architecture.html

[29] Concepts [online]. Gravitee.io API Platform [cit. 2020/06/24]. Dostupné z: https://docs.gravitee.io/apim/1.x/apim_overview_concepts.html

[30] Components [online]. Gravitee.io API Platform [cit. 2020/06/24]. Dostupné z: https://docs.gravitee.io/apim/1.x/apim_overview_components.html

[31] Plugins [online]. Gravitee.io API Platform [cit. 2020/06/24]. Dostupné z: https://docs.gravitee.io/apim/1.x/apim_overview_plugins.html

[32] Saurav Daruka Jolt Transformation Overview [online]. Medium, 16. února 2019 [cit. 2020/06/24]. Dostupné z: https://medium.com/@sauravdaruka/https-medium-com-sauravdaruka-jolt-transformation-overview-ed769d28a3

[33] Značka bajtového pořadí - Byte order mark [online]. QW, [cit. 2020/07/14]. Dostupné z: https://cs.qwe.wiki/wiki/Byte_order_mark

[34] Weather API [online]. FORECA [cit. 2019/11/24]. Dostupné z: https://corporate.foreca.com/en/weather-data/weather-api

[35] Abhishek Gupta WHAT IS GATEWAY IN NETWORKING | FUNCTION OF GATEWAY [online]. LEARNABHI.COM, 13. října 2017 [cit. 2019/11/24]. Dostupné z: https://www.learnabhi.com/gateway-computer-network/

[36] What is an API Gateway? [online]. NGINX, 24. října 2019 [cit. 2019/12/10]. Dostupné z: https://youtu.be/hYgP0cBORVg?t=105

[37] Rabeea Tahir Video Hosting Costs in Cloud vs. On-Premises Infrastructure [online]. [cit. 2019/12/10]. Dostupné z: https://blog.vidizmo.com/cloud-vs.-on-premises-video-hosting-costs

[38] Architecture [online]. Gravitee.io API Platform [cit. 2020/06/24]. Dostupné z: https://docs.gravitee.io/apim/3.x/apim_overview_architecture.html

Přílohy

A Instalační příručka

Tato příloha obsahuje návod k instalaci vytvořené mobilní aplikace a serveru.

A.1 Instalace Gravitee.io

Pro správný a funkční běh API Gatewaye Gravitee.io potřebujete naistalovat Docker. Docker pro Windows 10 HOME version 2004+, Windows 10 Pro/Enterprise, Linux Engine a Mac lze stáhnout na

https://www.docker.com/get-started. Musím mít 64 bit verzi a v BIOS musíte mít povolenou virtualizaci.

Pro Windows 10 Home s menší verzí než 2004, jako mám třeba já, můžete stáhnout Docker Toolbox

https://github.com/docker/toolbox/releases, který si automaticky stáhne a nainstaluje Kinematic (Alpha), Docker Quickstart Terminal a pokud nemáte tak i Virtual Machine Virtual Box. Dále potřebujete nástavbu na docker docker-compose, který umožní běh ve více kontejnerech. Verze pro Windows (Docker Desktop a Docker Toolbox) již docker-compose obsahují, není tedy potřeba to odděleně instalovat. V ostatních případech si stačí stáhnout docker-compose pomocí pip

```
pip install docker-compose
```

Poté co si Docker úspěšně stáhnete a nainstalujete, si stáhnete soubory .env, docker-compose.yaml a Dockerfile přiložené k bakalářské práci. Nyní můžete spustit Docker. Pokud používáte operační systém Windows, tak nejprve musíte zadat příkaz

```
docker build .
```

ve složce `image`. Tento příkaz vytvoří image, který je potřebný pro běh Gravitee ve Windows. Dále zadáte příkazu

```
docker-compose up -d
```

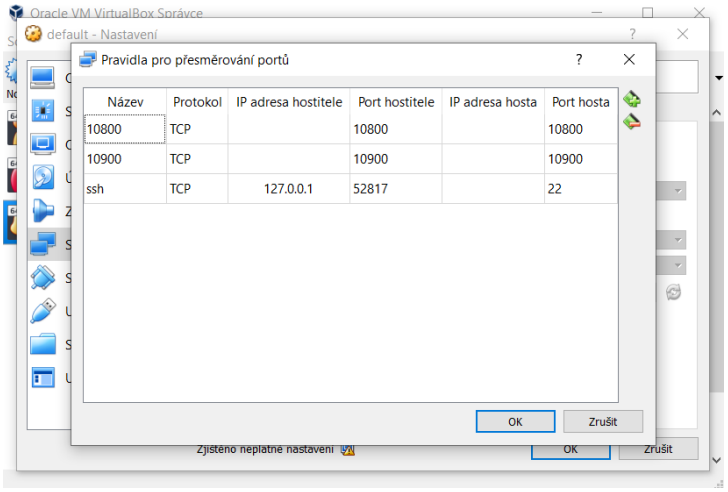
nad složkou `gravitee`, kde máte umístěné .env a docker-compose.yaml soubory. Díky yaml souboru si stáhne docker potřebné images. Potřebujete tři databáze:

- konfigurační (management) databázi
- databázi na ukládání ratelimit dat
- elasticsearch na analytická data

Ke splnění prvních dvou podmínek stačí mongodb, ale můžou se použít i redis nebo relační databáze jdbc.

Pokud používáte operační systém Linux, který podporuje Docker, tak můžete přeskočit až nakonec této kapitoly. Pro ty, co používají operační systém Windows musí ještě pokračovat.

Zkontrolujte jestli máte pro VirtualBox nastavený dostatečně velké uložiště. Dále je potřeba přemapovat VirtualBoxu porty. Vyberte váš virtuální disk a jděte do *Nastavení*. Vyberte možnost *Síť*, rozklikněte *Pokročilé*. Po kliknutí na *Předávání portů* vám vyskočí tabulka, ve které lze přemapovat porty. Pokud máte VM Virtual Box v anglickém jazyce, tak cesta bude následující: *network / adapter / advanced / port frowarding*.



Obrázek A.1: Porty VirtualBoxu

Z obrázku A.1 lze vyčíst port hostitele (port vašeho počítače) a port hosta (port Virtual Machine). Port hosta se definuje v souboru *docker-compose.yml* pro každý image v sekci port. Port si můžete přemapovat na kterýkoli port.

Spuštění API Managementu

Nyní už stačí jen zkontrolovat, jestli všechny images běží správně. Stačí zadat tento příkaz, který vypíše všechny aktivní images. Měli byste tam najít mongodb, elasticsearch, haproxy, management api, gateway a management ui.

```
docker ps
```

Když nyní zadáte správnou URL, tak se dostanete na uživatelské rozhraní Gravitee.io API Management. Defaultní port je 10900.

http://localhost:<port>/management/

Shození docker compose

Tento příkaz zastaví všechny images v dockeru, které jsme potřebovali na běh Gravitee.io.

```
docker-compose down
```

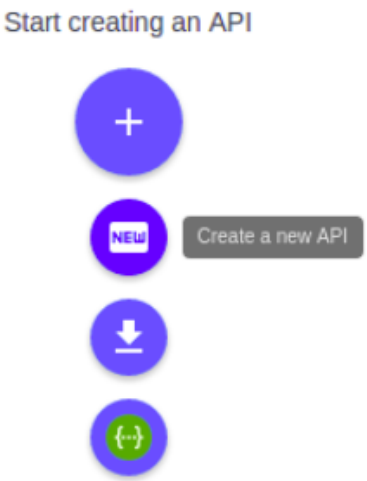
Návody na jednotlivé instalace naleznete na stránkách

https://docs.docker.com/desktop/

B Uživatelská příručka

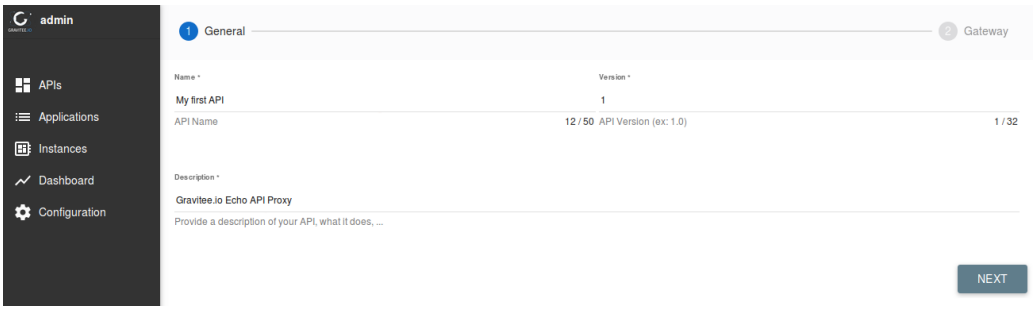
B.1 Vytváření API v Gravitee.io

Přihlašte se do Gravitee.io. Defaultní účet administrátora je admid/admin. Poté klikněte na plus a *Create a new API*.



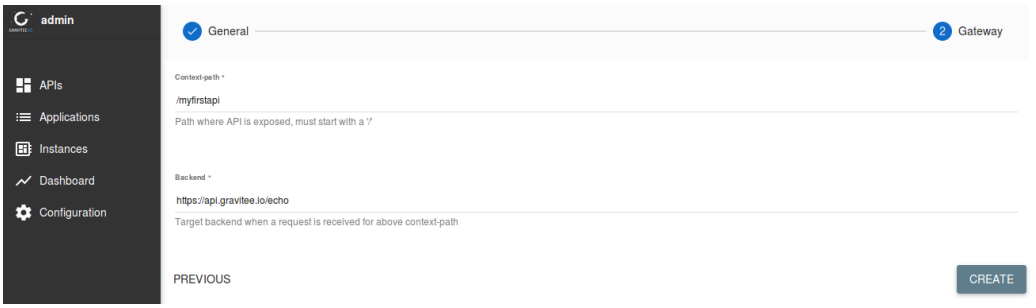
Obrázek B.2: Tlačítko pro vytvoření API

Dejte svému prvnímu aplikačnímu rozhraní jméno, verzi a stručný popis.



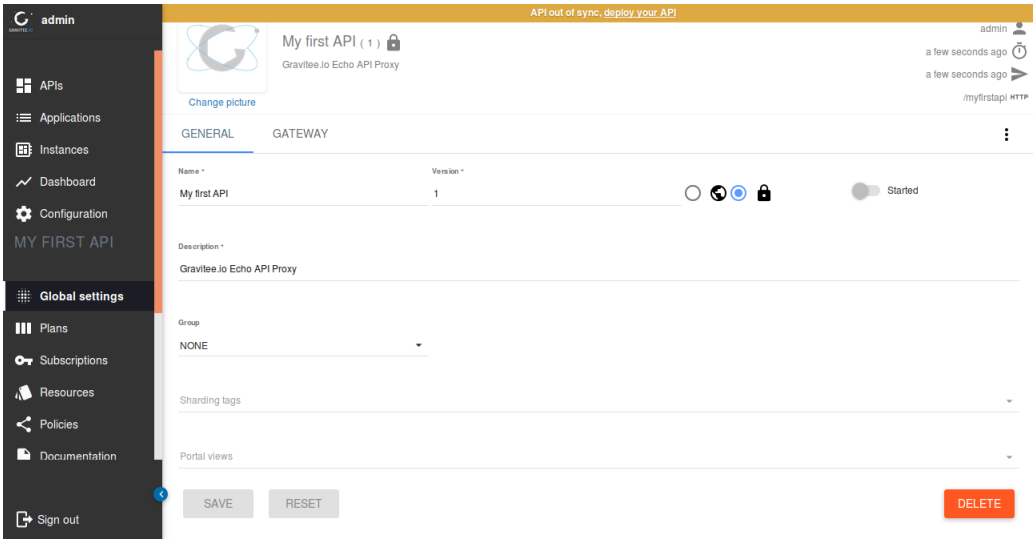
Obrázek B.3: Vytváření aplikačního rozhraní

Dále vyplňte context path (kontextovou cestu) a kolonku backend. Path je část URL za hostname a portem. V podstatě je to endpoint. Do Backend zadejte url. Například *https://api.gravitee.io/echo*



Obrázek B.4: Vytváření API – context path

Potom už stačí jen kliknou na tlačítko *Create* vpravo dole. Na obrázku B.5 můžete vidět úspěšně vytvořené aplikační rozhraní.

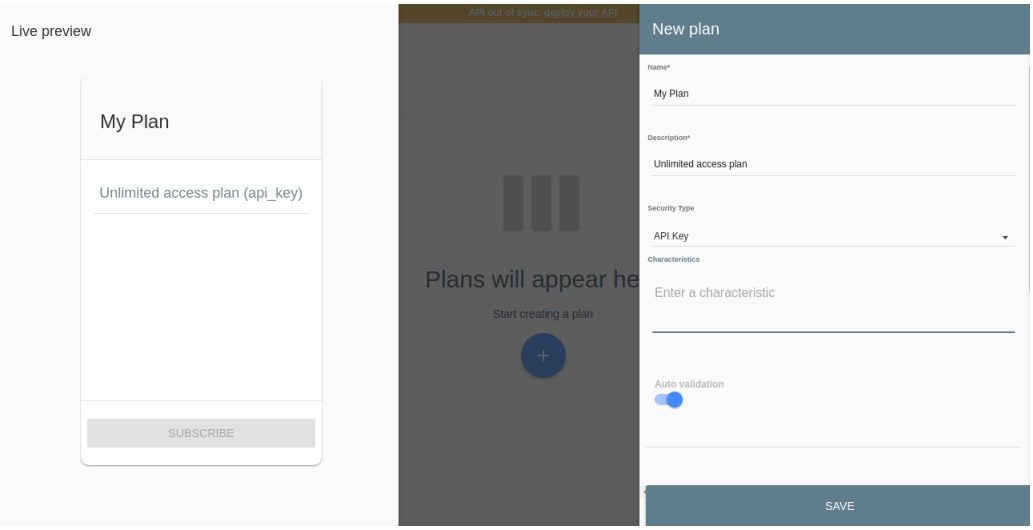


Obrázek B.5: Vytvořené API

B.2 Vytváření API Plánu

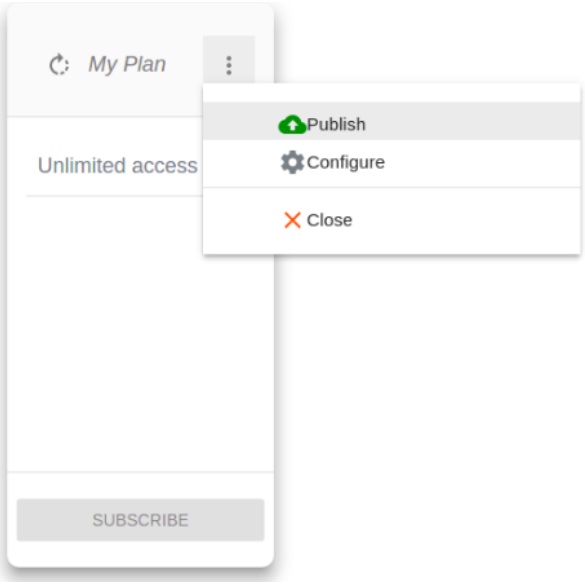
K úspěšnému publikování vašeho aplikačního rozhraní potřebujete aktivní API plán, který vám umožní získat přístup k API operacím. K vytvoření vašeho prvního API plánu klikněte na *Plans* (položku v menu) a postupujte podle návodu.

Nastavte vašemu plánu jméno, stručný popis, typ ochrany (security type) a potvrďte *Auto validation*.



Obrázek B.6: Vytvářené API plánu

Nyní tento plan uložte pomocí tlačítka *Save*. Váš plán je vytvořený. Teď ho musíte publikovat, aby byl viditelný pro vaše aplikační rozhraní. Klikněte na tři tečky vedle jména vašeho API a zvolte *Publish*. Tím se váš API plán zveřejní.



Obrázek B.7: Publikování API plánu

B.3 Nasazení API

Pro nasazení vašeho API do instance Gravitee.io Gateway stačí kliknutí na *deploy your API* uprostřed nahoře na žlutooranžovém proužku.

Obrázek B.8: Nasazení aplikačního rozhraní do API Gateway

B.4 Nastartování API

Nyní už stačí jen dané aplikační rozhraní nastartovat. Vaše API pustíte tak, že zmáčknete tlačítko *Start*.

Úspěšně jste vytvořili a publikovali aplikační rozhraní a plán. Nyní ho může aktivně spotřebovávat a upravovat.

B.5 Spuštění konfiguračního nástroje

Konfigurační nástroj lze spustit pomocí příkazu

```
java -jar config_tool.jar C:\\user\\zdrojove_kody\\api
```

C Obsah přiloženého CD

Popis obsahu přiloženého CD:

- `readme.txt` - textový soubor popisující obsah cd a návodu na spuštění konfiguračního nástroje
- `text` - složka obsahující text bakalářské práce a zdrojové kódy textu v LaTeXu
- `zdrojove_kody` - Složka obsahující dva adresáře. První s názvem `config_tool`, který obsahuje zdrojové kódy konfiguračního nástroje a spustitelný soubor `config_tool.jar`. Ve druhém adresáři `api` je zdrojový soubor `api.json` pro importování API přes konfigurační nástroj.
- `gravitee` - Složka obsahující dva soubory a jeden adresář. První soubor `docker-compose.yml` definuje služby potřebné pro běh Gravitee.io v Docker. Druhý soubor `.env` definuje potřebné environmentální proměnné a adresář `image`, který obsahuje soubor `Dockerfile`, který vytvoří novou image v Docker.