FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání diplomové práce

Název:	Správa identity a přístupů v mikroservisní architektuře
Student:	Bc. Dominik Dosoudil
Vedoucí:	Ing. Martin Komárek
Studijní program:	Informatika
Obor / specializace:	Webové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2023/2024

Pokyny pro vypracování

Hlavním cílem práce je nalézt a prakticky ověřit vhodnost open-source nástroje pro správu identity a přístupů v systémech s mikroservisní a cloud-native architekturou. Požadavky na reálný provoz pravidlelně konzultujte s odborníky minimálně ze dvou nezávislých firem (např.: INSOFT s.r.o., Stratox Cloud Native s.r.o.) .
Dílčí úkoly:
Proveďte rešerši open-source nástrojů typu "Identity provider a OAuth provider". Při rešerši nástrojů zohledněte především následující vlastnosti a možnosti:
- Podpora mikroservní architektury.
- Cloud-native kompatibilita.
- Bezvýpadkové škálování při nasazení v Kubernetes.
- Spustitelnost na OS Ubuntu jako nativní aplikace a nebo v python runtime.
- Bezestavová autentizace a autorizace (access + refresh token).
- Revokace refresh tokenu.
- Podpora RBAC a ACL.
- Konfigurovatelnost typu přihlášení.
- Podpora SPNEGO Kerberos.
- Snadnost/možnost customizace UI login page.

Vyberte dva nejvhodnější nástroje a na ukázkovém projektu s mikroservisní architekturou implementujte zabezpečení jednotlivých služeb a ověřte podporu v rešerši uvedených vlastností. Výsledky diskutujte.

Elektronicky schválil/a Ing. Jaroslav Kuchař, Ph.D. dne 17. února 2023 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Správa identity a přístupů v mikroservisní architektuře

Bc. Dominik Dosoudil

Katedra softwarového inženýrství
Vedoucí práce: Ing. Martin Komárek

4. května 2023

Poděkování

Chtěl bych vyjádřit poděkování všem, kteří přispěli k dokončení této práce. V první řadě děkuji vedoucímu práce Ing. Martinu Komárkovi za konzultace postupu, přístup a ochotu. Dále děkuji Janu Klepalovi z firmy INSOFT s.r.o. za poskytnuté konzultace. Zároveň bych rád poděkoval i všem ostatním zaměstnancům firmy INSOFT s.r.o. za podporu a shovívavost v mé nepřítomnosti v práci. Nakonec bych rád poděkoval svým spolužákům protože nic mě nepřimělo jít dál víc, než vědomí, že nejsem sám. Zejména pak děkuji své přítelkyni Iloně nejen za rady, ale hlavně, že vždy stála při mně i v těch nejtěžších chvílích.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principu při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisu. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisu, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programu, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli zpusobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelum). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným zpusobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 4. května 2023 . . . . . .

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Dosoudil, Dominik. *Správa identity a přístupů v mikroservisní architektuře.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2023.

Abstrakt

Firmám INSOFT s.r.o. a Stratox Cloud Native s.r.o. nevyhovuje jejich dosavadní způsob správy uživatelů, a tak potřebují najít náhradu. Cílem je najít pro obě firmy vhodný nástroj na základě jejich požadavků a kompatibility s jejich produkty. Práce obsahuje rešerši vlastností existujících aplikací určených pro správu uživatelů a přístupů. Nalezené vlastnosti jsou kvantifikovány ve veřejných interaktivních tabulkách pro další využití. Na základě hodnocení jsou vybrány 2 nástroje. Tyto nástroje jsou následně integrovány do ukázkového systému využívajícího mikroservisní architekturu. Po ověření funkčnosti jednotlivých vlastností služeb je firmám doporučen nástroj ZITADEL. Pro firmu Stratox Cloud Native s.r.o. nástroj splňuje všechny zadané požadavky a firma ho může nabízet v rámci svého produktu jako alternativu ke stávajícímu systému pro autentizaci a autorizaci uživatelů. Firma INSOFT s.r.o. má komplexní systém autorizací a náhrada nebude bez nutnosti úprav, ale na základě této práce má firma podklady k tomu, jaké úpravy je potřeba provést pro možnost nahrazení správy uživatelů.

Klíčová slova autorizace, autentizace, správa identit a přístupů, přihlašování uživatelů, mikroservisní architektura, cloud prostředí, horizontální škálování, ZITADEL, Logto

Abstract

Companies INSOFT s.r.o. and Stratox Cloud Native s.r.o. aren't satisfied with their current state of user management. Therefore they need to find a replacement. The aim of this thesis is finding a suitable tool that satisfies needs of each company and is compatible with their product. The thesis contains research of features of various user management and authorization applications. Found features and attributes are quantified in public interactive spreadsheets. Those spreadsheets can be used by anyone. Based on score computed in spreadsheets two tools are selected and integretad into an example application that uses microservice architecture. Experimental integration allowed to verify previously found features of analyzed tools. After the verification and summarization, ZITADEL was recommended as an identity and access management tool to both companies. ZITADEL satisfies requirements stated by Stratox Cloud Native s.r.o. and can be offered within their software product as an alternative to their current user authentication and authorization service. Although requirements of company INSOFT s.r.o. weren't fully fulfilled because of their complex authorization system, ZITADEL still matches their demands better than Logto.

Keywords authentication, authorization, identity provider, user login, microservice architecture, cloud environment, horizontal scaling, ZITADEL, Logto

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Kvalita poskytovaných on-line služeb neustále roste, aby se zvyšovala přívětivost pro jejich uživatele. Tím rostou i velikosti systémů, které zajišťují tuto kvalitu, která uživatele přiměje používat právě jejich službu.

Čím větší systém je, tím hůře se udržuje, opravuje a modernizuje. Bývá komplikované provést změnu, která se týká celého kódu. Například někdy je problematické už jen zvýšit verzi programovacího jazyka nemluvě o kompletní změně jazyka.

Pro umožnění změn na menších jednotkách lze použít tzv. mikroservisní architekturu, ve které jsou jednotlivé služby v systému samostatné aplikace.

Tyto samostatně běžící služby však potřebují ověřit identitu a oprávnění uživatele. Nastává otázka, jak takové zabezpečení zajistit. Nabízí se, aby s tímto problémem pomohla další samostatná služba.

Ovšem architektura, která rozděluje systém do několika částí vyžaduje, aby tomu byla přizpůsobená i daná autentizační a autorizační služba. Pokud je navíc potřeba takovou službu spustit v cloud prostředí a škálovat ji, je potřeba, aby na to tato služba byla také připravena.

Kapitola 1

Cíl práce

Cílem této diplomové práce je vybrat vhodný nástroj pro správu identit a přístupů pro firmy INSOFT s.r.o. a Stratox Cloud Native s.r.o. Obě firmy mají konkrétní požadavky na vlastnosti a schopnosti těchto služeb odvíjející se od prostředí, které firmy používají a systémů, které vyvíjí.

Cílem teoretické části této práce je provedení rešerše různých nástrojů typu „Identity provider“ a zjištění, zda podporují požadované vlastnosti. Výsledkem analýzy bude konfigurovatelné srovnání nástrojů v interaktivní podobě, což umožní upravit důležitost nebo hodnocení vlastností podle svých potřeb. Tyto tabulky budou následně nastaveny podle potřeb obou firem, načež se zobrazí přizpůsobené hodnocení všech nástrojů. Z nejlépe hodnocených nástrojů budou vybráni 2 kandidáti tak, aby alespoň jeden z nich byl vhodný pro alespoň jednu firmu.

Rešerše si klade za cíl zohlednit především tyto parametry:

1. podpora mikroservisní architektury,
2. cloud-native kompatibilita,
3. bezvýpadkové škálování při nasazení v Kubernetes,
4. spustitelnost na OS Ubuntu nativně, nebo v Python runtime,
5. bezestavová autentizace a autorizace (access + refresh token),
6. revokace refresh tokenu,
7. podpora RBAC a ACL,
8. podpora SSO a federace identit (např. pomocí Kerberos),
9. snadnost/možnost přizpůsobení UI login page.

Cílem praktické části je vytvořit ukázkový systém používající mikroservisní architekturu, integrovat do něj obě vybrané služby a ověřit funkčnost vlastností. Dále vyhodnotit výsledky ověření vlastností a vybrat doporučený nástroj pro obě firmy.

Kapitola 2

Problematika správy identit a přístupů

Tato kapitola nejprve popisuje rozdíl mezi ověřením přístupu do aplikace v monolitní a mikroservisní architektuře. Následně vysvětluje technologie, protokoly a jiné pojmy, které tato práce diskutuje.

2.1 Monolit

Pokud je aplikace monolitní a není rozdělena na jednotlivé služby, je možné provádět autentizaci a autorizaci v rámci této aplikace. Existuje několik způsobů, jak toho dosáhnout. Jedním z možných způsobů je po ověření uživatelských přihlašovacích údajů vytvořit unikátní náhodné ID, kterým se uživatel prokáže při dalších dotazech. Tím odpadá nutnost vyplňování přihlašovacích údajů při každé interakci se systémem a zároveň nejsou uživatelovy přihlašovací údaje nikde uloženy.

Ovšem aby systém dokázal s id spárovat správného uživatele, musí si tuto informaci někam uložit. Při následné interakci se systémem nejprve dotaz zpracuje autentizační vrstva a k dotazu přiloží odpovídajícího uživatele. [11] Dotaz pak předá ke zpracování dalším vrstvám, které již mají dostupného uživatele a informace o něm. Na základě těchto informací se mohou rozhodnout, jestli má tento uživatel k akci oprávnění.

Tento mechanismus má následující klíčové prvky:

1. Provedení uživatelem požadované akce vždy předchází zpracování autentizační vrstvou.

2. Aplikace si drží stav potřebný k párování id přihlášení a uživatele.

2.2 Mikroservisní architektura

Metoda použitá v monolitním systému může být v mikroservisní architektuře nepraktická, jelikož by jednotlivé služby musely mezi sebou sdílet stav, aby mohly ověřit validitu přihlášení. Sdílení stavu by pak šlo proti myšlence oddělování jednotlivých částí systémů od sebe.

Jedním z řešení je zavedení tzv. bezestavové autentizace, která spočívá v držení informace o přihlášeném uživateli v klientské části aplikace a posílání této informace spolu s každým dotazem. Tato informace je zabalená do tokenu, což je obvykle JWT (2.2.1). Token je navíc podepsaný, aby ho nebylo možné podvrhnout. [12]
Výhody:

- žádné dotazy navíc

Nevýhody:

- potřeba ověření tokenu v každé službě

Další možností je použít reverzní proxy. Pak dotaz vždy prochází skrz proxy, ta ověří identitu uživatele a zdali je uživatel autorizován k danému dotazu. Pokud ano, odešle dotaz odpovědné službě, která dotaz zpracuje. [11]
Výhody:

- celý systém implicitně zabezpečený
- vnitřní služby systému kompletně oddělené od autentizace a autorizace

Nevýhody:

- při každém dotazu je nutné, aby dotaz prošel autentizačním mechanismem této proxy

Obě metody mají výhody i nevýhody a nelze obecně tvrdit, zdali je jedna lepší než druhá. Podle mého názoru záleží i na tom, v jakém stavu je systém, do kterého je služba integrována. Například pokud již proxy používá k jiným účelům, může být efektivnější použít proxy řešení.

2.2.1 JWT

„JSON Web Token (JWT) představuje způsob pro bezpečnou výměnu informací mezi dvěma stranami. JWT je JSON objekt, který se skládá z hlavičky (header), dat (payload) a podpisu (signature). Podle specifikace RFC 7519. Cílem JWT je možnost ověření autenticity dat – skutečnosti, že data nebyla cestou změněna. Nikoli však skrýt obsah dat. Účelem zakódování dat je převod struktury dat do lépe přenositelné podoby.“ [13]

2.3 On-premise nasazení

„On-premise řešení představuje software či hardware, který je uložen v interní infrastruktuře a prostoru společnosti. U on-premise řešení veškerá údržba, aktualizace i zabezpečení spadají do správy společnosti samotné, či případně společnost najímá externí pracovníky, kteří zajišťují serverovou podporu.“ [14]

2.4 Cloud computing

V dnešní době se čím dál častěji využívá pro nasazení aplikací tzv. cloud namísto fyzického serveru. Cloud může být několika typů:

IaaS Infrastructure as a Service umožňuje vlastní konfiguraci infrastruktury systému, správce systému dostane k dispozici například virtualizovaný server, na který si může nainstalovat libovolné aplikace.

PaaS Platform as a Service znamená, že vývojář dostane již připravené prostředí potřebné pro spuštění vyvíjené aplikace. Ekosystém může znamenat například běhové prostředí Python nebo nainstalované některé systémové knihovny.

SaaS Software as a Service typ cloudu představuje už konkrétní službu, jako například Gmail. [15]

Výhoda tohoto přístupu nasazování je, že často stačí pouze v administraci nastavit, jaké zdroje (paměť, velikost disku, počet jader CPU...) aplikace vyžaduje a není potřeba nejprve kupovat a sestavovat hardware. Další výhodou je, že ve chvíli, kdy zdroje již nestačí, je možné je opět v administraci jednoduše navýšit. Často je možné škálovat velikosti parametrů dokonce automatizovaně podle aktuální zátěže na naše služby. [16]

2.4.1 Cloud native

„Cloud native je přístup k vývoji, nasazování a spravování moderních aplikací v cloud computing prostředí. Moderní společnosti chtějí vytvořit vysoce škálovatelné, flexibilní a pružné aplikace, které lze rychle aktualizovat pro potřeby uživatelů.“ ([17] překlad autora)

Tento přístup spočívá v tom, že se vyvíjený systém dělí do zmíněných mikroslužeb. Tyto služby mají definované API, pomocí kterého mohou spolu komunikovat. Využívá se serverless principu, což znamená, že nezáleží, na jaké infrastruktuře služba běží. To lze zajistit pomocí kontejnerizace. Díky těmto vlastnostem lze aplikace spouštět a aktualizovat nezávisle na sobě. [17]

2.5 Single Sign-On (SSO)

„SSO zpřístupňuje možnost uživatelům používat pouze jeden set hesla a přihlašovacího jména do různých aplikací.“ [18] Toho lze docílit použitím jednoho zdroje identit (IdP) pro více aplikací. Tomu se říká federace identit. Federace lze docílit pomocí několika protokolů. Tyto pojmy a protokoly jsou v následujících podsekcích popsány.

2.5.1 IdP

„IdP je systém, který vytváří, ukládá a spravuje digitální identity. Může buď přímo autentizovat uživatele nebo poskytovat autentizační služby poskytovatelům služeb třetích stran (aplikace, webové stránky, jiné digitální služby).“ ([19] překlad autora)

2.5.2 OAuth 2.0

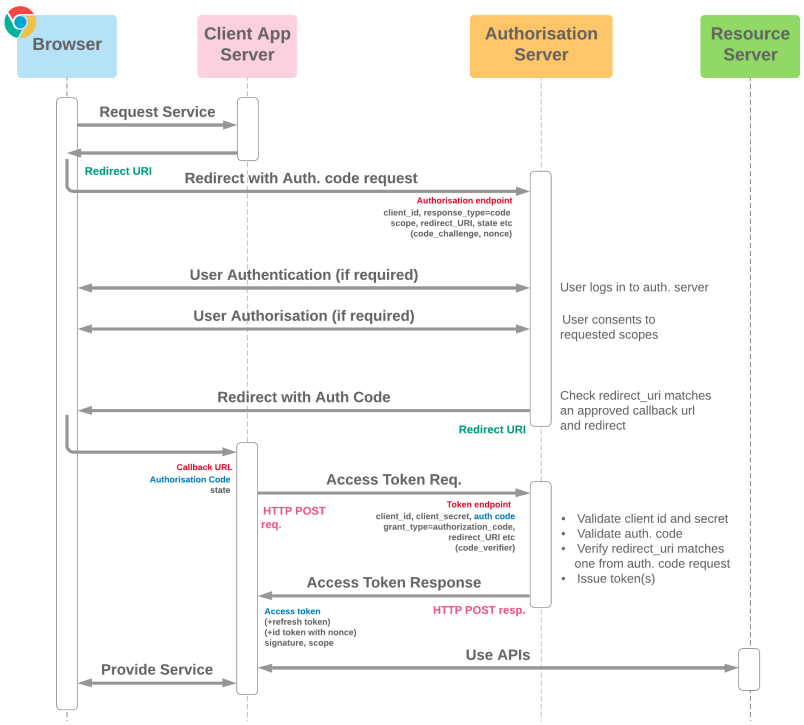
„OAuth 2.0 je protokol pro autorizaci. Zaměřuje se na jednoduchost pro vývojáře klientských aplikací. Poskytuje specifické autorizační procesy pro webové aplikace, desktopové aplikace, mobilní telefony a chytrá domácí zařízení. Specifikace a její rozšíření jsou vytvářeny v rámci IETF Oauth Working Group.“ ([20] překlad autora)

Tento protokol neslouží k autentizaci uživatelů. Jeho účel je primárně autorizovat službu k tomu, aby mohla přistupovat k nějaké jiné službě jménem přihlášeného uživatele.

Tuto autorizaci zajišťují access tokeny. Není definovaný konkrétní formát tokenu, ale obvykle se používá JWT (2.2.1). Token může obsahovat data o uživateli a z bezpečnostních důvodů obvykle obsahuje i čas expirace. [21]
Ve chvíli, kdy se uživatel pokusí provést akci, ke které je potřeba autorizace, nastane následující proces, který je znázorněn na diagramu 2.1:

1. Klientská aplikace požádá autorizační server o autorizaci.
2. Autorizační server požádá uživatele o schválení potřebných oprávnění.
3. Autorizační server po schválení poskytne klientské aplikaci autorizační kód.
4. Autorizační kód si klientská aplikace vymění za access token (a refresh token).
5. Klientská aplikace může provést původní požadovanou akci.

[21]



Obrázek 2.1: OAuth 2.0 [1]

2.5.3 Federated Identity

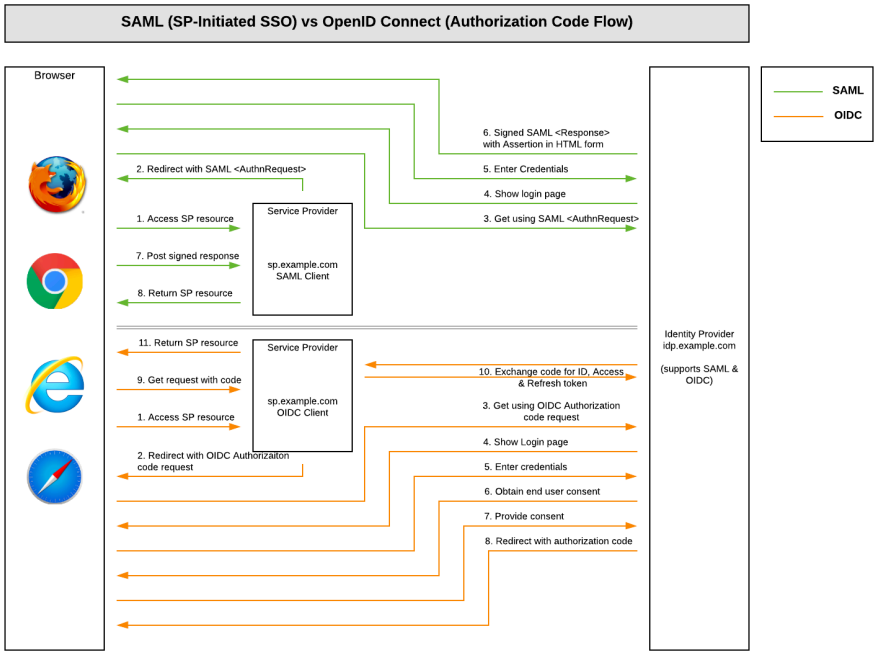
Tento pojem označuje spojení identit napříč více domén. To znamená, že pokud jsou 2 domény federované, může se uživatel identifikovat jednou identitou v obou doménách. Často je možné se setkat například s přihlášením pomocí sociálních sítí. [22]

Provider je strana, která potvrzuje identitu uživatele.

Relying party je strana mající zájem o potvrzení identity koncového uživatele. [23]

2.5.4 OIDC

OIDC je autentizační protokol postavený nad protokolem OAuth 2.0 popsaném v podsekci 2.5.2. Rozšíření spočívá v tom, že kromě access tokenu obdrží klientská aplikace po přihlášení uživatelem i ID token, který obsahuje informace o uživateli. Tento ID token figuruje jako důkaz autentizace uživatele. Rozdíl mezi ID tokenem a access tokenem je ten, že access token může být získán více způsoby, ale ID token může být získán jen v případě interakce uživatele v podobě přihlášení, tedy jsou potřeba jeho přihlašovací údaje. [22]



Obrázek 2.2: OIDC vs SAML [2]

2.5.5 SAML 2.0

SAML protokol slouží k autentizaci uživatele v různých aplikacích napříč doménami a je oblíbený hlavně v B2B a B2E aplikacích. Používá XML pro reprezentaci uživatelské identity. [24]

SAML funguje tak, že ve chvíli, kdy se uživatel pokusí přistoupit ke službě, služba vygeneruje SAML dotaz a přesměruje prohlížeč na autentizační server, který SAML dotaz zpracuje. Uživatel se následně přihlásí. Po úspěšném přihlášení autentizační server vytvoří SAML odpověď, kterou vrátí prohlížeči. Prohlížeč tuto SAML odpověď přepošle původní službě, která autentizuje uživatele. [25]

Srovnání průběhu autentizace pomocí OIDC a SAML je vidět na obrázku 2.2.

2.6 Oprávnění

Autorizační služba zajišťuje ověření, zda má uživatel oprávnění přistupovat ke konkrétnímu zdroji. Ověření přístupu se může zakládat na různých fakto-

rech. Například příslušnost ke skupině nebo uživatelova role. V této sekci jsou popsány modely řízení přístupu.

2.6.1 RBAC

Řízení přístupu na základě rolí (zkráceně RBAC) spočívá v rozdělení zaměstnanců ve firmě podle jejich zodpovědností. Na základě tohoto rozdělení se vytvoří role (vedoucí výroby, účetní...), které jsou pak zaměstnancům (uživatelům systému) přiřazovány. Jedné osobě je možné přiřadit i více rolí. [26] Výhoda tohoto způsobu řízení přístupu je jednoduchost. Avšak tato jednoduchost je podmíněná nalezením vhodného rozdělení zodpovědností zaměstnanců do několika skupin. Všichni v jedné skupině musí mít stejná oprávnění. Skupiny se mohou překrývat. Tyto pomyslené skupiny, což mohou být v praxi firemní oddělení, představují v systému role.

Pokud by ale každému uživateli byla přiřazena jiná role s unikátní množinou oprávnění, vznikne velmi mnoho rolí a ztrácí tento způsob smysl. Je tedy potřeba nejprve rozmyslet, jak jsou rozděleny oprávnění v organizaci a zdali mezi uživateli vzniknou některé shody.

2.6.2 ABAC

Řízení přístupu na základě vlastností oproti RBAC funguje tak, že oprávnění přístupu je vyhodnocováno při každém dotazu na základě vlastností uživatele (oddělení, pozice...) nebo aktuálního prostředí (čas, zdrojová adresa/aplikace požadavku...). Díky tomu je ABAC flexibilnější, než RBAC, ale zároveň je složitější na vyhodnocení. V RBAC stačí ověřit, že uživatel má potřebnou roli. V ABAC je potřeba zjistit a vyhodnotit všechny parametry, což může být výpočetně náročnější. [27]

2.6.3 ACL

Access Control List spočívá v tom, že namísto oprávnění přidělených uživateli jsou specifikována pravidla pro jednotlivé uživatele přímo na zdroj dat. Na základě seznamu pravidel je následně vyhodnoceno, jestli daný uživatel může ke zdroji přistupovat, případně jakým způsobem. ACL je lepší pro případy, kdy potřebujeme oprávnění definovat jednotlivě. Naopak s sebou nese více administrativní práce při nastavování a spravování oprávnění. [28]

Kapitola 3

Analýza vlastností nástrojů

V této kapitole jsou vypsané vlastnosti, které nástroje pro správu uživatelů mohou mít. Následující rešerše se na ně zaměřuje. Poté jsou vypsané nalezené nástroje a u každého z nich popsáno, které vlastnosti mají, případně jakým způsobem tyto vlastnosti u konkrétního nástroje fungují. Nakonec jsou tyto vlastnosti shrnuty v interaktivních tabulkách s nastavitelnou váhou. Díky tomu je možné nástroje mezi sebou srovnávat a přizpůsobit hodnocení podle vlastních potřeb.

3.1 Vhodné vlastnosti

Na základě konzultací s oběma firmami bylo zjištěno, že některé požadavky se překrývají. Těmito požadavky jsou *vhodnost pro mikroservisní architekturu* a *možnost spustit a spravovat hledaný nástroj svépomocí (self-hosting)*. Z toho důvodu se rešerše zabývá pouze takovými nástroji. Dále jsou pro alespoň jednu z firem významné následující vlastnosti:

1. Open-source
2. Federace identity
   a) SPNEGO Kerberos
   b) OIDC
   c) SAML
3. Oprávnění uživatelů
   a) RBAC/ABAC/ACL
   b) Skupiny
4. Cloud native

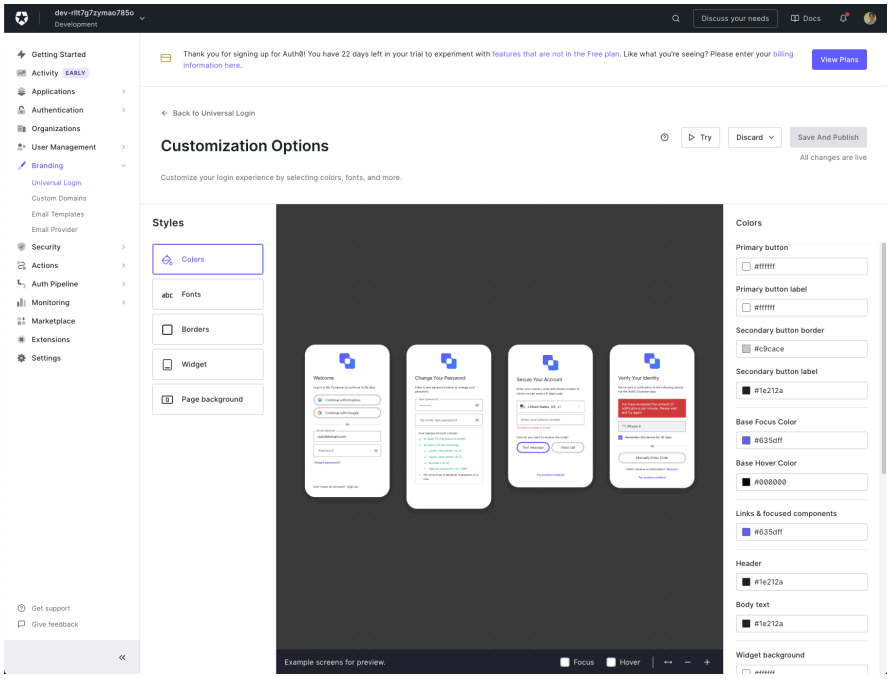
   a) Bezestavovost (vydávání tokenů)
   b) Možnost horizontálního škálování
   c) Serverless
5. Přizpůsobení UI login page

3.2 Výběr zkoumaných nástrojů

V této části jsou vyhledány a vybrány nástroje pro rešerši. Vyhledání proběhlo mimo jiné na portálu AlternativeTo (`https://alternativeto.net/category/networking-and-admin/identity-management/`), který umí vyhledávat alternativní software. Dále bylo využito fórum reddit, kde existuje kanál „selfhosted“ (`https://www.reddit.com/r/selfhosted`), na kterém se diskutuje o nástrojích, které lze spustit svépomocí a není nutné je využívat jako SaaS službu.

Při hledání nástrojů na zmíněných portálech se může ve výsledcích vyhledávání zobrazit i nesprávně zařazený software, který se správou uživatelů nemá nic společného. Z toho důvodu byly pro rešerši vybrány pouze nástroje, které se v popisu na portálu nebo jejich webových stránkách prezentují jako „autentizační a autorizační služba“, „správa identit“ nebo „správa uživatelů“.

1. Auth0 `https://auth0.com/`
2. Authorizer `https://authorizer.dev/`
3. FusionAuth `https://fusionauth.io`
4. Keycloak `https://www.keycloak.org/`
5. Pomerium `https://www.pomerium.com/`
6. Authelia `https://www.authelia.com/`
7. Ory `https://www.ory.sh/`
8. Ory Kratos `https://www.ory.sh/kratos`
9. ZITADEL `https://zitadel.com/`
10. Janssen Project `https://github.com/JanssenProject/jans`
11. SuperTokens `https://supertokens.com/`
12. Logto `https://logto.io/`
13. LoginRadius `https://www.loginradius.com/`
14. authentik `https://goauthentik.io/`



Obrázek 3.1: Auth0 administrace – přizpůsobení přihlašovací stránky

3.3 Rešerše vlastností vybraných nástrojů

V této části je zejména z oficiálních webových stránek a dokumentací nástrojů zjištěno, zda jsou opravdu nástroji, které umožňují přihlašování a splňují alespoň část požadovaných vlastností. V případě že nástroj lze jednoduše spustit, jsou tyto vlastnosti prozkoumány i v administraci služby.

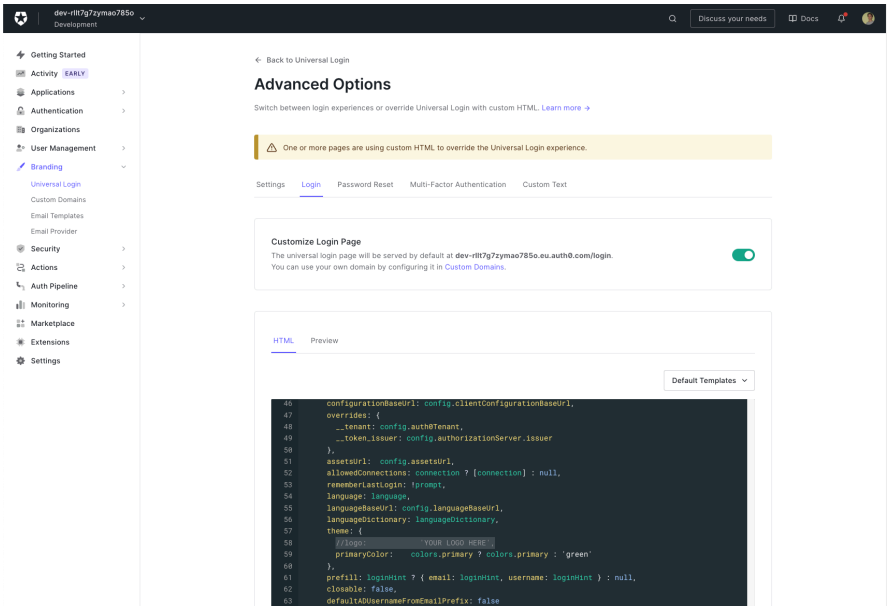
3.3.1 Auth0

Auth0 je služba umožňující velmi rychle nasadit přihlášení do aplikace. Po registraci a prvním přihlášení do jejich administrátorské aplikace se zobrazí průvodce, který pomůže s prvním nastavením a po výběru, do jakého typu aplikace bude vývojář přihlášení přidávat (např. SPA v React knihovně), poradí, jak přihlášení integrovat do aplikace.

Zároveň průvodce umožní grafické přizpůsobení přihlašovací stránky (barvy, okraje, zarovnání textu, písmo, pozadí, pořadí prvků ve formuláři), jak je vidět na obrázku 3.1. Upravit lze i logo zobrazené nad formulářem, jeho nastavení je ovšem trochu skryté v sekci „Pokročilé nastavení“ -> „Login“, kde je potřeba URL adresu obrázku upravit přímo v JavaScript kódu viz obrázek 3.2.

x

Nicméně Auth0 je proprietární služba typu SaaS, tedy není možné aplikaci nasadit tzv. self-hosted. Z toho důvodu služba nevyhovuje potřebě obou firem,



Obrázek 3.2: Auth0 administrace – přizpůsobení loga

a to možnost spustit a spravovat nástroj svépomocí.

3.3.2 Authorizer

Přehled vlastností k této aplikaci je dostupný v tabulce: 3.1.

Aplikace Authorizer má na první pohled hned několik kladných vlastností. V první řadě na webových stránkách zmiňují, že je poskytována zdarma. Dále je zřejmé, že vyzdvihují možnost okamžitého nasazení na PaaS prostředí Railway, Heroku a Render a naznačují, že službu lze zprovoznit ve 3 krocích: spuštění instance, její konfigurace a následná integrace do aplikace [29].

Kromě nasazení na PaaS je možné službu spustit self-hosted dvěma cestami. Authorizer nabízí buď předkompilované binární soubory, nebo zdrojový kód s návodem na kompilaci a spuštění. Kompilace a spuštění bylo vyzkoušeno a proběhlo bez obtíží.

Ve výchozím nastavení Authorizer používá databázi SQLite, ale podporuje napojení na mnoho jiných databázových systémů (MongoDB, Cassandra, PostgreSQL, Couchbase…) [29].

Authorizer server v sobě má zabudovanou přihlašovací stránku, u které lze změnit logo a název organizace viz obrázek 3.3. Dále poskytuje i knihovnu `@authorizerdev/authorizer-js` a variantu speciálně pro React a Svelte. Díky těmto knihovnám je integrace do aplikace relativně jednoduchá a navíc nabízí možnost velkého přizpůsobení, protože si vývojář může naprogramovat celý vlastní formulář.

Tabulka 3.1: Authorizer – přehled vlastností

vlastnost	stav
web	`https://authorizer.dev/`
dokumentace	`https://docs.authorizer.dev/`
sandbox	nenalezeno
self-hosted	ANO
cloud-native	nenalezeno
nasazení	Railway, Heroku, Render, binární aplikace, Kubernetes
role	ANO
skupiny	NE
přizpůsobení UI	ANO (logo, název organizace)
protokoly	nenalezeno
cena	zdarma
jazyk implementace	Go
open-source	ANO
podporované db	MongoDB, Cassandra, PostgreSQL, Couchbase...

Samotný server podporuje vytváření a přidělování rolí, což je jedna z požadovaných vlastností. Dále se Authorizer na svých webových stránkách zmiňuje o podpoře Kubernetes, což je také velmi příhodné.

Aktivita projektu vypadá oproti ostatním projektům průměrně až podprůměrně. Kromě posledního čtvrtletí je počet příspěvků do projektu konzistentní. Poslední čtvrtletí je aktivita nižší. Za poslední rok do projektu přispělo 13 lidí, z čehož 2 výrazně více. Graf příspěvků je na obrázku 3.4. Projekt vznikl v roce 2021, takže je možné, že se ještě prosadí. V tuto chvíli to ale nevypadá, že by byl naprosto opuštěn, ale ani že by měl velkou komunitu. [3]

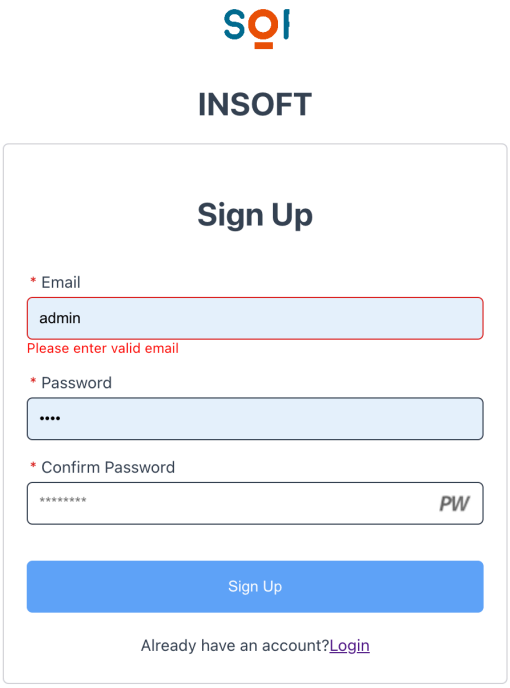
3.3.3 FusionAuth

Přehled vlastností k této aplikaci je dostupný v tabulce: 3.2.

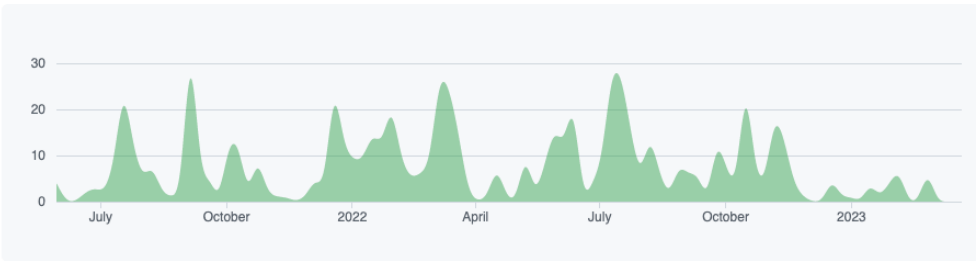
FusionAuth nabízí jak cloud řešení, tak možnost self-hostingu. Ačkoliv je tento produkt placený, pro self-hosting nabízí komunitní edici, která neobsahuje veškeré funkcionality, ale je zcela zdarma.

Nasazení je možné udělat několika připravenými způsoby: spuštění z připraveného Docker obrazu, instalace pomocí deb, rpm, homebrew balíčků, nebo instalace na Windows pomocí PowerShell. Zároveň je na webových stránkách zmíněná podpora Kubernetes. [32]

Spuštění bylo vyzkoušeno pomocí připravené varianty s Docker compose. Bohužel spuštění neproběhlo tak snadno, jako u Authorizer. Instance databáze i aplikace sice naběhla, ale aplikaci se nepodařilo k databázi připojit.



Obrázek 3.3: Authorizer – přizpůsobené logo a název organizace



Obrázek 3.4: Authorizer – graf počtu commitů [3]

Tabulka 3.2: FusionAuth – přehled vlastností

vlastnost	stav
web	`https://fusionauth.io/`
dokumentace	`https://fusionauth.io/docs/`
sandbox	`https://sandbox.fusionauth.io/`
self-hosted	ANO
nasazení	deb, rpm, Docker, Kubernetes
role	ANO
skupiny	ANO
přizpůsobení UI	šablony, v administraci
protokoly	SAML, OIDC
cena	zdarma CE
jazyk implementace	Java [30]
open-source	NE
podporované db	MySQL, Postgres [31]

Kvůli chybě při spuštění byla tedy administrace prozkoumána v nabízeném sandboxu.

FusionAuth, podobně jako Authorizer, podporuje role. Navíc umožňuje přidávat skupiny, ke kterým lze stejně jako k uživatelům přiřadit vytvořené role. Neumožňuje ale hierarchii skupin. Po přihlášení do administrace FusionAuth je patrné, že nabízí mnohem více různého nastavení oproti Authorizer. Přizpůsobení UI je možné pomocí „Témat“, které umožňují vytvářet HTML šablony formulářů (obr. 3.6). Dále administrace nabízí jednoduchý monitoring, který zobrazuje například počet přihlášení za den. Seznam uživatelů je možné vidět na obrázku 3.5.

Jelikož FusionAuth nemá otevřený zdrojový kód, není možné zjistit informace o počtu příspěvků. Na StackOverflow je ale 312 příspěvků zmiňujících FusionAuth, což je oproti konkurenčním nástrojům relativně hodně.

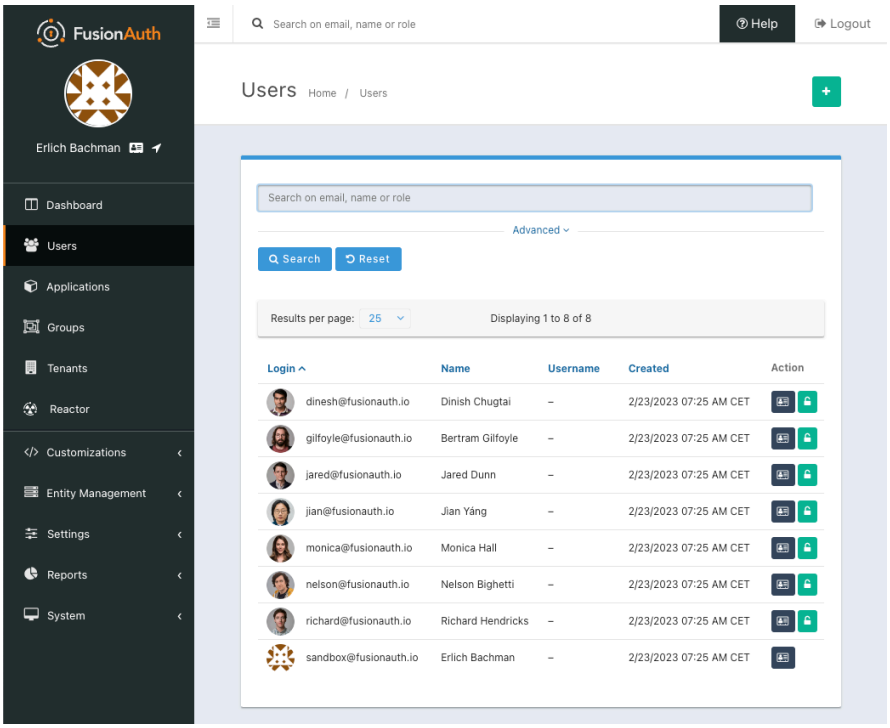
3.3.4 Keycloak

Přehled vlastností k této aplikaci je dostupný v tabulce: 3.3.

Keycloak je open-source řešení poskytované zcela zdarma. Neposkytuje oficiální cloud hostování. Soustředí se na self-hosting.

Na svém webu nabízejí ke stažení buď sestavenou JAR aplikaci, nebo Docker obraz vhodný pro Docker, Podman, Kubernetes a OpenShift. [35] Spuštění jedné instance pro úvodní vyzkoušení pomocí Docker obrazu proběhlo bez jakýchkoliv problémů.

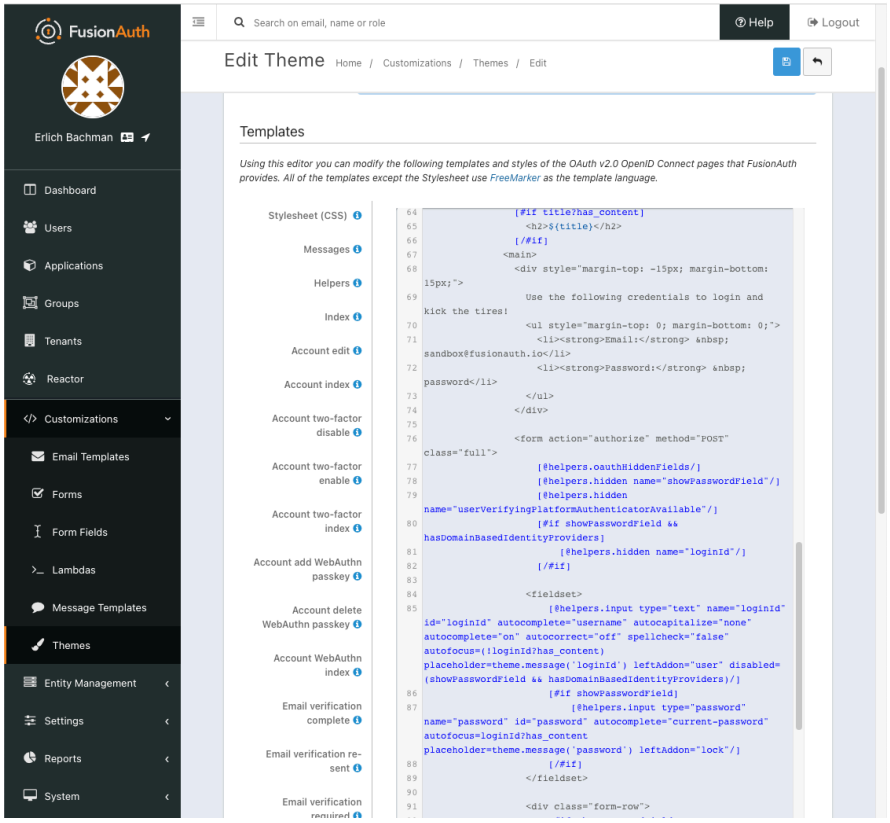
Keycloak podle webových stránek podporuje protokoly OIDC, SAML 2.0 a Kerberos, díky kterým je možné se přihlašovat pomocí jiných autentizačních serverů. [33] Oproti FusionAuth nástroj Keycloak podporuje hierarchické sku-



Obrázek 3.5: FusionAuth – přehled uživatelů

Tabulka 3.3: Keycloak – přehled vlastností

vlastnost	stav
web	`https://www.keycloak.org`
dokumentace	`https://www.keycloak.org/documentation`
sandbox	nenalezeno
self-hosted	ANO
nasazení	JAR, Docker, Podman, Kubernetes
role	ANO, možnost přiřadit vlastnosti
skupiny	hierarchické
přizpůsobení UI	ANO, pomocí šablonových souborů
protokoly	OIDC, SAML 2.0, Kerberos [33]
cena	zdarma
jazyk implementace	Java
open-source	ANO
podporované db	MariaDB, MSSQL, MySQL, Oracle, Postgres [34]

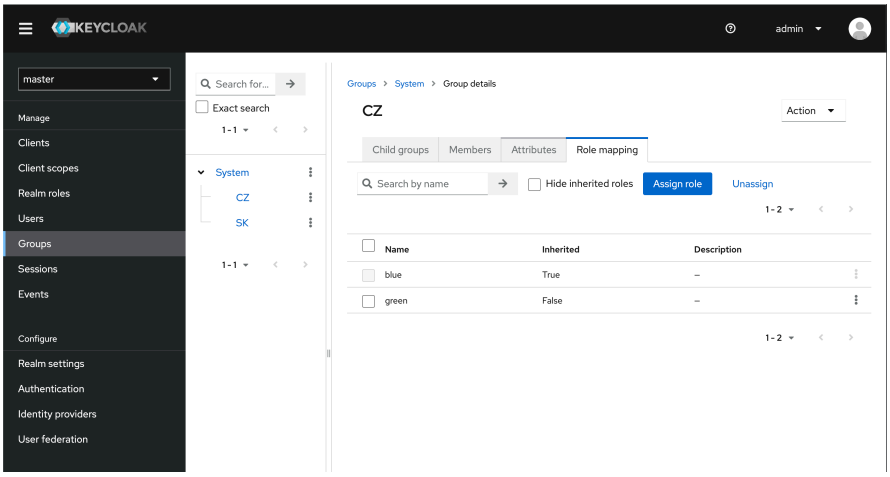


Obrázek 3.6: FusionAuth – úprava šablon

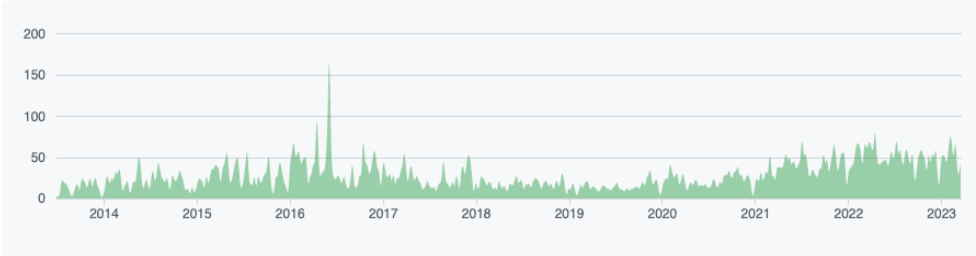
piny, kde každé skupině lze přidělit role, jak může být vidět na obrázku 3.7. Role přiřazené vyšším skupinám jsou zděděny do podskupin, nicméně nelze je již v nějaké podtřídě zneplatnit. To na jednu stranu přináší jednoduchost a přehlednost, na druhou stranu to ale snižuje flexibilitu. Role mohou mít přiřazené vlastnosti v podobě klíče a hodnoty. Stejně lze přiřadit vlastnosti uživatelům.

Přizpůsobení UI je oproti předchozím nástrojům komplikovanější, jelikož je potřeba do specifické složky nahrát soubory šablon. [36] To je nepřívětivé pro horizontální škálování, jelikož se šablona špatně instaluje do jednotlivých instancí. Tento způsob přizpůsobení přihlašovací stránky byl konzultován s firmou Stratox Cloud Native s.r.o., která má s Keycloak již zkušennosti a toto řešení považuje za nevýhodu. Pro srovnání ve FusionAuth a Authorizer bylo možné UI přizpůsobit přímo v administraci.

Aktivita projektu je velmi dobrá. Za poslední rok do projektu přispělo 58 lidí, z čehož zhruba u 20 je přispívání do projektu průběžné za celý rok. Jak je možné vidět na obrázku 3.8, množství commitů za celý život projektu je konzistentní a za poslední 2 roky dokonce lehce vyšší, nicméně je to pravděpo-



Obrázek 3.7: Keycloak – hierarchické skupiny

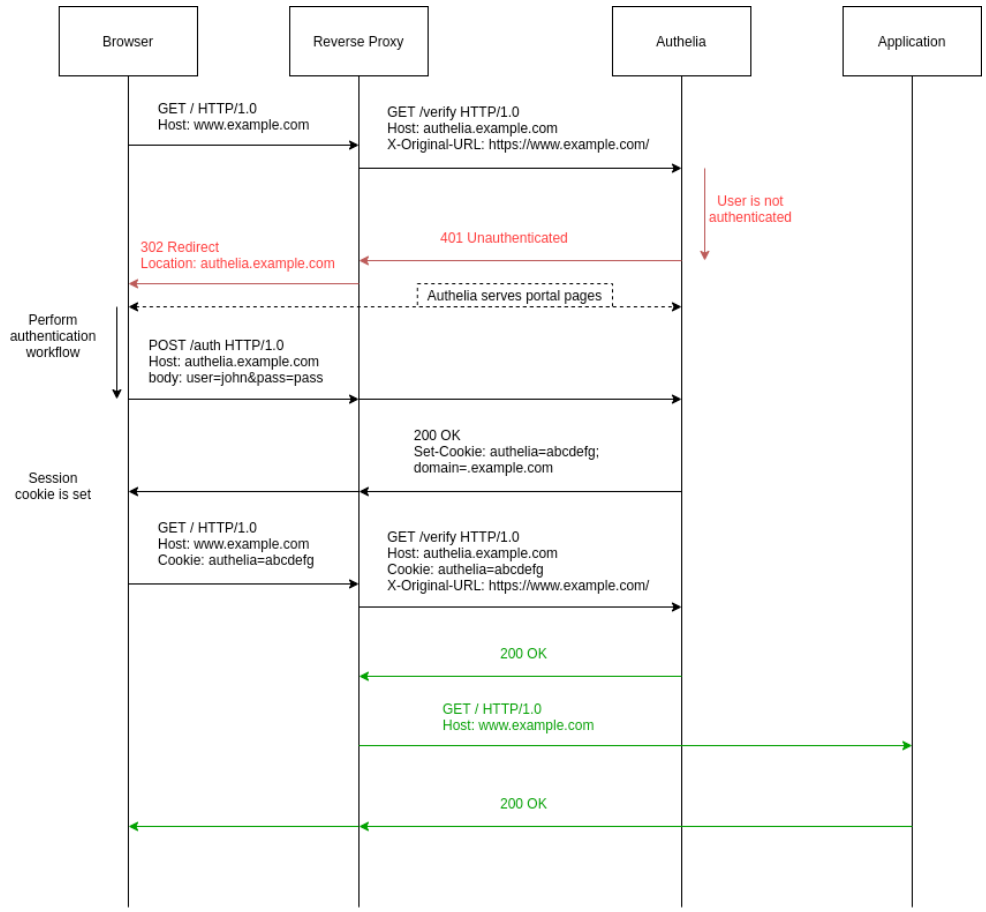


Obrázek 3.8: Keycloak – graf počtu commitů [4]

dobně způsobeno botem „dependabot[bot]“, který zhruba tou dobou začal do projektu přispívat. [4] Počet otázek na StackOverflow zmiňujících „Keycloak“ je přes 15 000, což je řádově více než u ostatních nástrojů. Zároveň je Keycloak nejstarší z projektů. Byl vytvořen již v roce 2013. To samozřejmě může znamenat klady i zápory. Na jednu stranu lze předpokládat, že je nástroj již dobře odladěn, na druhou stranu je možné, že si s sebou nese tzv. technický dluh.

3.3.5 Pomerium

Pomerium slouží podobně jako předchozí služby k autentizaci a autorizaci uživatelů, nicméně nezajišťuje samotné přihlášení uživatele. K tomu potřebuje jinou autentizační službu. Po přihlášení si ale dokáže zapamatovat identifikátor zařízení uživatele. Takto zapamatovaným zařízením pak věří a umožňuje jim komunikaci do systému. Pomerium funguje spíše jako alternativa k VPN. [37] Jelikož služba nezajišťuje přihlašování, nebude dále zkoumána v této rešerši.



Obrázek 3.9: Authelia – průběh dotazu [5]

3.3.6 Authelia

Přehled vlastností k této aplikaci je dostupný v tabulce: 3.4.

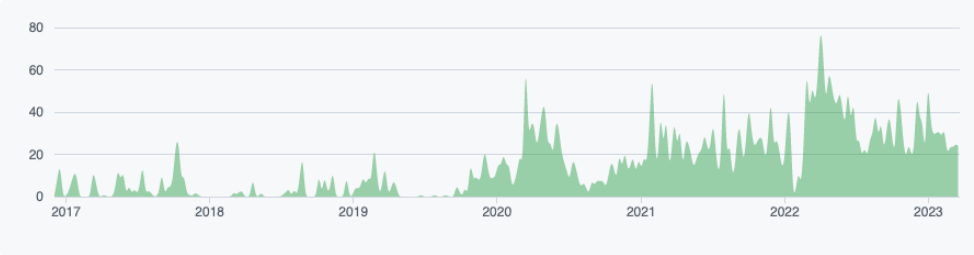
Authelia je autentizační služba spolupracující s proxy serverem. Dalo by se říct, že funguje jako rozšíření proxy poskytující autentizaci a přihlašovací stránku. [5] Průběh zpracování dotazu je možné vidět na obrázku 3.9, který ukazuje, že proxy každý dotaz nejprve ověří dotázáním se služby Authelia a až následně posílá originální dotaz požadované službě.

Podle dokumentace Authelia umožňuje vytváření skupin. [38] Na základě skupin a dalších kritérií, jako například subdomény nebo uživatelského jména, lze vytvářet autorizační pravidla, podle kterých Authelia rozhodne, zda je uživatel oprávněn provést požadovanou akci.

Authelia podporuje protokol OIDC pro federaci uživatelů, nicméně pouze v provider roli. Tedy není možné instanci Authelia napojit na jiné IdP. [39]

Tabulka 3.4: Authelia – přehled vlastností

vlastnost	stav
web	`https://www.authelia.com/`
dokumentace	`https://www.authelia.com/`
sandbox	nenalezeno
self-hosted	ANO
cloud-native	ANO, bezestavovost [40]
nasazení	binární soubory, Linux balíčky, Docker, Kubernetes [41]
role	nenalezeno
skupiny	ANO
přizpůsobení UI	nenalezeno
protokoly	OIDC (pouze provider), LDAP
cena	zdarma
jazyk implementace	Go
open-source	ANO
podporované db	MySQL, SQLite3, PostgreSQL [42]



Obrázek 3.10: Authelia – graf počtu commitů [6]

Služba Authelia vznikla v roce 2016 a statistiky na GitHub má srovnatelné s Keycloak. Navíc má velice dobrý poměr otevřených k vyřešeným problémům, a to 1:10. Podle grafu na obrázku 3.10 se aktivita za poslední 3 roky zvýšila. Za poslední rok přispělo 28 lidí, z čehož 4 přispívali průběžně celý rok. [6] StackOverflow nicméně nalezl pouze 29 výsledků obsahujících slovo „authelia“.

3.3.7 Ory

Ory je placená cloud služba nabízející řešení pro flexibilní autentizaci, autorizaci a řízení přístupu. [43] Jelikož placená SaaS řešení nejsou předmětem rešerše, nebude Ory hlouběji zkoumáno. Nicméně zdrojový kód aplikací, které Ory používá pro provozování SaaS, je otevřený a vydávaný pod Apache 2.0 licencí. V následující podsekci 3.3.8 bude prozkoumána jedna z těchto aplikací

Tabulka 3.5: Ory Kratos – přehled vlastností

vlastnost	stav
web	`https://www.ory.sh/kratos/`
dokumentace	`https://www.ory.sh/docs/kratos/ory-kratos-intro`
self-hosted	ANO
cloud-native	ANO
nasazení	Kubernetes, Google Cloud, AWS...
role	potřeba Ory Keto
skupiny	nenalezeno
přizpůsobení UI	neobsahuje integrované UI
protokoly	OIDC [44]
cena	zdarma, Apache 2.0
jazyk implementace	Go
open-source	ANO
podporované db	SQLite, PostgreSQL, MySQL, CockroachDB

zajišťující správu uživatelů.

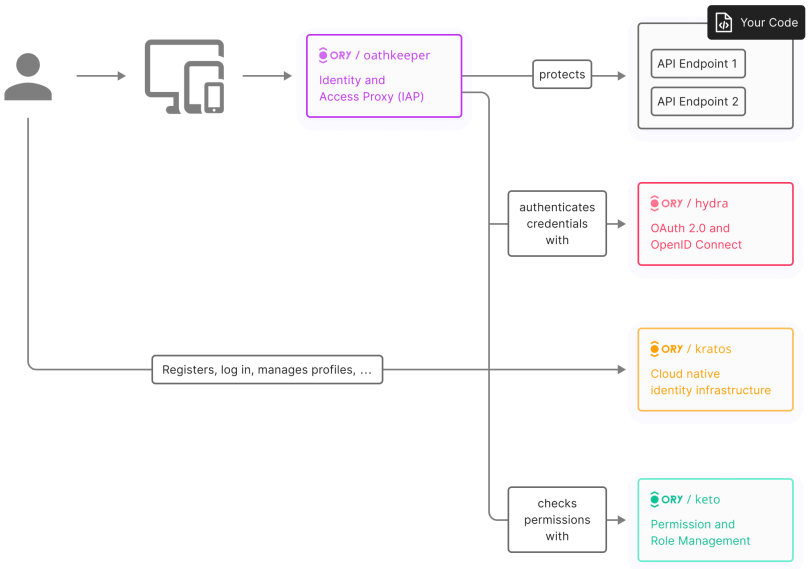
3.3.8 Ory Kratos

Přehled vlastností k této aplikaci je dostupný v tabulce: 3.5.

Ory Kratos je konfigurovatelná služba pro autentizaci a správu uživatelů bez integrovaného UI. Podporuje vícefázové ověření, přihlášení pomocí sociálních sítí, vlastní identity a více. Navíc je připravená pro nasazení do cloudového prostředí. [45] V dokumentaci této služby se jako výhoda uvádí nezávislost na dalších službách a knihovnách. Podle dokumentace je jediná závislost některý z podporovaných RDBMS (SQLite, PostgreSQL, MySQL, CockroachDB) [46] a knihovna libc/libmusl [47]. Díky tomu by mělo být nasazení velmi snadné a vytvářet malé Docker obrazy (< 20 MB). [46] Dokumentace také uvádí dobrou podporu horizontálního škálování v cloudovém prostředí díky bezestavovosti aplikace a rychlému spuštění. [47]

Dokumentace aplikace je obsáhlá a přehledná. Kratos se snaží být „lightweight“ a UI lze naprosto libovolně přizpůsobit, jelikož samotná služba je na UI nezávislá, což poskytuje flexibilitu. Buď může být přihlašovací stránka implementována přímo do konkrétní aplikace [48], nebo lze vytvořit službu poskytující jen danou přihlašovací stránku, což je pravděpodobně vhodnější při vytváření mikroservisního systému. URL přihlašovací stránky se následně nastaví v konfiguračním souboru *config.yml* [49]. Neintegrované UI může znamenat i nevýhodu. Tento přístup totiž nemusí být vhodný ve všech případech a opět záleží na konkrétním systému, pro který má být služba použita.

Samotný Kratos se orientuje pouze na správu identit a proces přihlášení.



Obrázek 3.11: Ory ekosystém [7]

Pro podporu rolí je potřeba přidat spolupracující službu Ory Keto nebo pro podporu JWT je potřeba Ory Oathkeeper. Spolupráce mezi těmito službami je znázorněna na obrázku 3.11.

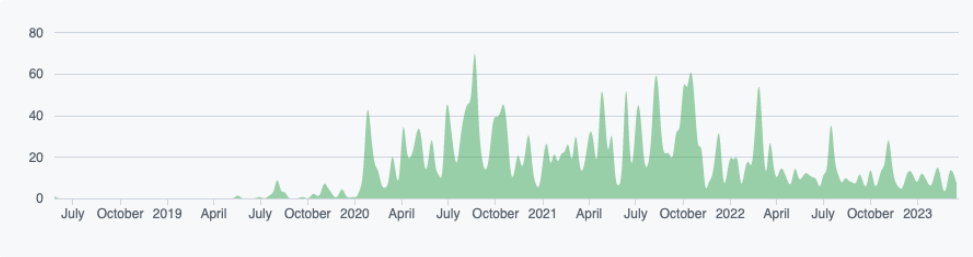
Z tohoto důvodu Ory Kratos není příliš vhodný pro rychlé a jednoduché nasazení kompletní autentizace a autorizace. Vhodný je spíše v případech potřeby flexibility, jelikož lze použít pouze některé služby z ekosystému a ostatní buď vynechat, nebo nahradit alternativou.

Ory Kratos získávalo největší množství příspěvků mezi lety 2020 až 2022. Za poslední rok se aktivita relativně snížila, nicméně ve srovnání s ostatními nástroji je i tak relativně v pořádku. Za poslední rok má Kratos 29 přispěvatelů, z čehož 5 přispívalo průběžně. Při zkoumání příspěvků do projektu za celý jeho život bylo zjištěno, že opravdu velkou část projektu vytvořil jeden člověk. [8] Jedná se o nevýhodu, jelikož projekt je velmi závislý pouze na jedné osobě. V případě, že tento vývojář projekt opustí, může tím skončit vývoj a podpora celého projektu.

3.3.9 ZITADEL

Přehled vlastností k této aplikaci je dostupný v tabulce: 3.6.

ZITADEL je autentizační a autorizační služba nabízející jak placené SaaS ře-



Obrázek 3.12: Ory Kratos – graf počtu commitů [8]

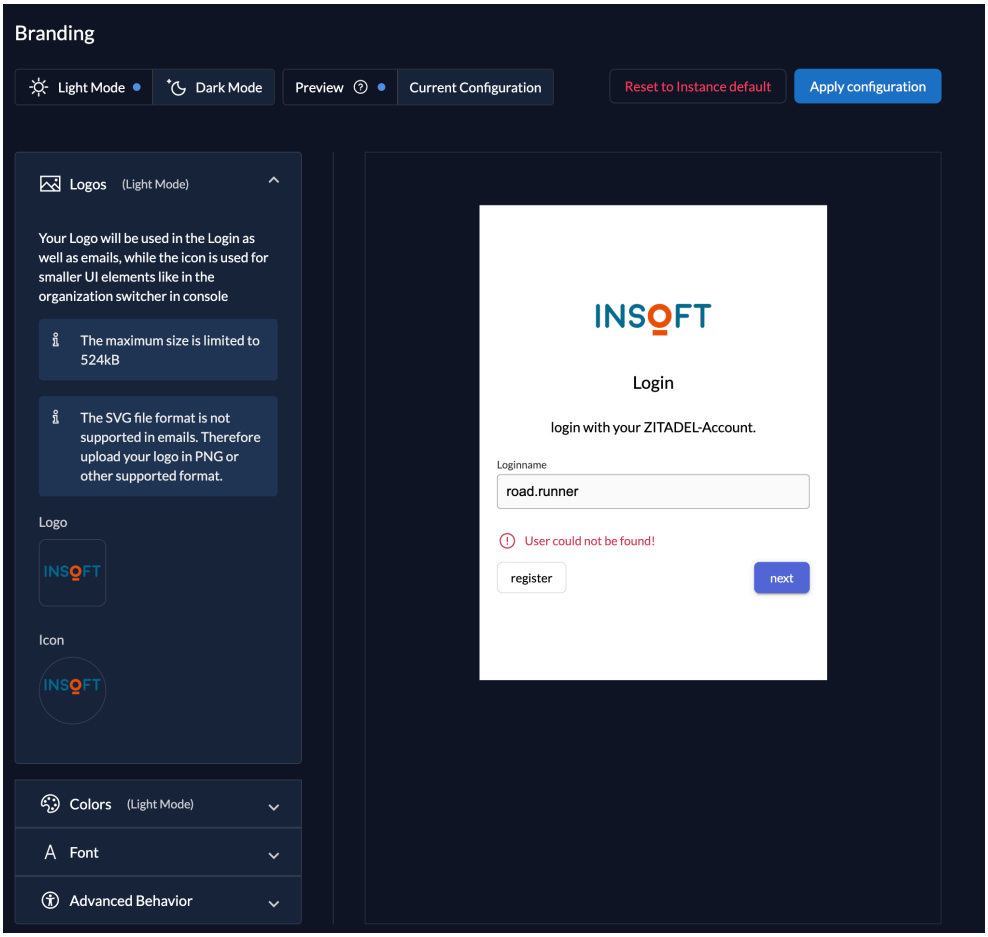
Tabulka 3.6: ZITADEL – přehled vlastností

vlastnost	stav
web	`https://zitadel.com/`
dokumentace	`https://zitadel.com/docs`
sandbox	nenalezeno
self-hosted	ANO [50]
cloud-native	ANO [51]
nasazení	Linux, Docker compose, Knative, Kubernetes [52]
role	ANO
skupiny	NE
přizpůsobení UI	ANO, per organizace
protokoly	OIDC, OAuth 2.0, JWT IDP [53]
cena	zdarma, Apache 2.0
jazyk implementace	Go, Angular
open-source	ANO
podporované db	CockroachDB, PostgreSQL (BETA)

šení, tak možnost self-hosted aplikace kompletně zdarma. Podle webových stránek je služba vhodná pro ty, kteří chtějí jednoduché nastavení jako Auth0 a open-source univerzální produkt jako Keycloak [50].

ZITADEL cílí na cloud-native podporu, což zahrnuje škálovatelnost, dynamické prostředí a jednoduchou změnu poskytovatele hostitelských služeb. [51] Službu je možné spustit v „highly available“ režimu, díky čemuž bude aplikace bezestavová a je možné spustit několik instancí paralelně. [54]

Dále ZITADEL poskytuje přívětivé podmínky pro B2B integraci. ZITADEL systém poskytuje možnost založit více organizací s různým nastavením ověřováním identity. Díky tomu je možné zaměstnance různých firem ověřovat různými způsoby. Zároveň je možné každé organizaci nastavit vlastní vzhled přihlašovací stránky a nechat zákazníky spravovat vlastní zaměstnance v dané organizaci (např. přidělování rolí). [55] Přihlašovací stránku je možné přizpůsobit pro každou organizaci zvlášť. Náhled lze pozorovat na obrázku 3.13.

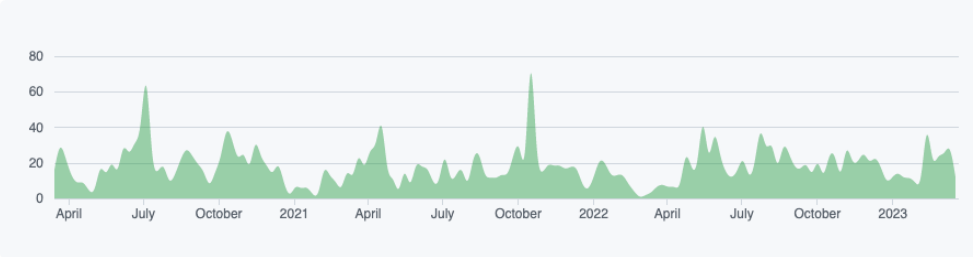


Obrázek 3.13: ZITADEL – přizpůsobení přihlašovací stránky

Role jsou podporovány pouze v jednoduché formě. Není možné rolím přiřazovat žádná metadata, oprávnění ani vlastnosti. Jediná informace navíc, kterou je možné rolím přiřadit, je název skupiny. Nicméně tato skupina má význam pouze pro seskupení rolí. Například je možné přiřadit uživateli všechny role z nějaké skupiny. [56]

V administraci ZITADEL je možné uživatele deaktivovat a znemožnit mu tím přihlášení. Revokace tokenů je možná pomocí API. [57]

Projekt ZITADEL vznikl v roce 2020. GitHub statistiky jsou s přihlédnutím na rok vzniku lehce podprůměrné, nicméně podle grafu příspěvků na obrázku 3.14 si projekt po celou dobu života udržuje konzistentní počty příspěvků. Za poslední rok do vývoje nástroje přispělo 37 lidí, z čehož 8 přispívalo průběžně, což je dobrý poměr. Zároveň žádný z největších 4 přispěvatelů výrazně nedominuje. [9] Z toho důvodu je hodnocení komunity ZITADEL spíše



Obrázek 3.14: ZITADEL – graf počtu commitů [9]

kladné.

3.3.10 Janssen Project (dříve oxAuth od Gluu)

Přehled vlastností k této aplikaci je dostupný v tabulce: 3.7.

Nástroj Janssen Project se původně jmenoval oxAuth. Byl součástí Gluu Server a lze často narazit na odkazy do původní dokumentace pod původním názvem. [58]
Janssen Project si stanovuje 3 cíle: [58]

1. horizontální škálovatelnost,
2. vysoká dostupnost,
3. flexibilita upgradování.

Mnohé části dokumentace nejsou zatím dokončené a nebylo možné aplikaci jednoduše spustit, jelikož Docker obraz není připravený pro ARM architekturu. Z těchto důvodů nebylo možné zjistit některé oficiální informace.

Janssen Project má výrazně nižší množství GitHub hvězd, nicméně graf commitů 3.15 vypadá velmi dobře. I když se průměrné množství příspěvků za poslední rok snížilo zhruba o polovinu, stále je to ve srovnání s jinými projekty dost. Za poslední rok přispělo 38 lidí, z čehož 13 přispívalo průběžně. Projekt je tedy aktivní. [63]

3.3.11 SuperTokens

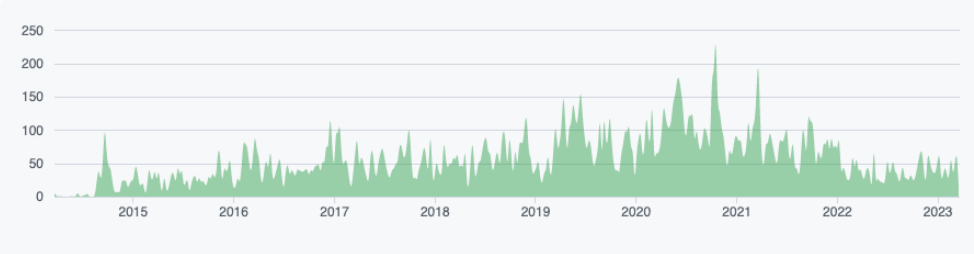
Přehled vlastností k této aplikaci je dostupný v tabulce: 3.8.

SuperTokens opět nabízí jak SaaS placené řešení, tak self-hosted open-source aplikaci zcela zdarma. [65]

Služba působí velmi moderně a nabízí rychlou a jednoduchou možnost spuštění testovací instance, čímž umožňují vyzkoušení přihlášení. [66] Nicméně není možné si touto cestou prohlédnout administraci.

Tabulka 3.7: Janssen – přehled vlastností

vlastnost	stav
web	`https://jans.io/`
dokumentace	`https://docs.jans.io/v1.0.8/`
sandbox	nenalezeno
self-hosted	ANO
cloud-native	ANO
nasazení	Linux balíčky, Docker, Kubernetes, Amazon EKS... [59]
role	nenalezeno
skupiny	nenalezeno
přizpůsobení UI	nahrazení výchozích šablon [60]
protokoly	OAuth, OIDC, SCIM, FIDO [61]
cena	zdarma, Apache 2.0
jazyk implementace	Java
open-source	ANO
podporované db	RDBMS Erwin Table, LDAP, Couchbase, MySQL, Spanner [62]



Obrázek 3.15: Janssen Project – graf počtu commitů

Služba SuperTokens udržuje přihlášení pomocí access a rotujících refresh tokenů a umožňuje jejich revokaci. [67]

Je podporováno přidělování rolí uživatelům a navíc je možné ke každé roli přiřadit seznam oprávnění, což umožňuje vytvořit přehlednější a flexibilnější systém práv. [64]

Služba SuperTokens používá architekturu znázorněnou na obrázku 3.16. Systém se skládá ze 3 aplikací: frontend, backend a jádro.

Frontend zobrazuje přihlašovací a registrační stránku. Lze vytvořit v React, Vue, Angular...

Backend fungující jako most mezi jádrem a frontend částí. Podpora NodeJS, Python, Go.

Tabulka 3.8: SuperTokens – přehled vlastností

vlastnost	stav
web	`https://supertokens.com/`
dokumentace	`https://supertokens.com/docs/guides`
sandbox	nenalezeno
self-hosted	ANO
cloud-native	nenalezeno
nasazení	binární aplikace, Docker, Kubernetes
role	ANO (+ oprávnění) [64]
skupiny	NE
přizpůsobení UI	NE, pouze pokud napíšeme vlastní FE
protokoly	OAuth, OIDC, SAML
cena	zdarma
jazyk implementace	Java
open-source	ANO
podporované db	MySQL, PostgreSQL

Jádro je hlavní část spustitelná pomocí Docker nebo předsestavené binární aplikace.

Pro fungování systému je potřeba vytvořit frontend a backend, s čímž velmi pomohou připravené knihovny, a následně všechny 3 části spustit.

Návod k přizpůsobení přihlašovací stránky není nikde k nalezení, nicméně na úvodní stránce „https://supertokens.com/“ jsou zobrazeny přihlašovací stránky společností, které SuperTokens používají, a některé z nich mají například vlastní logo. Podle mého názoru je to možné pokud frontend napíšeme vlastní bez použití některých částí připravených knihoven. To nám umožní libovolný vzhled, ale bude to samozřejmě pracnější.

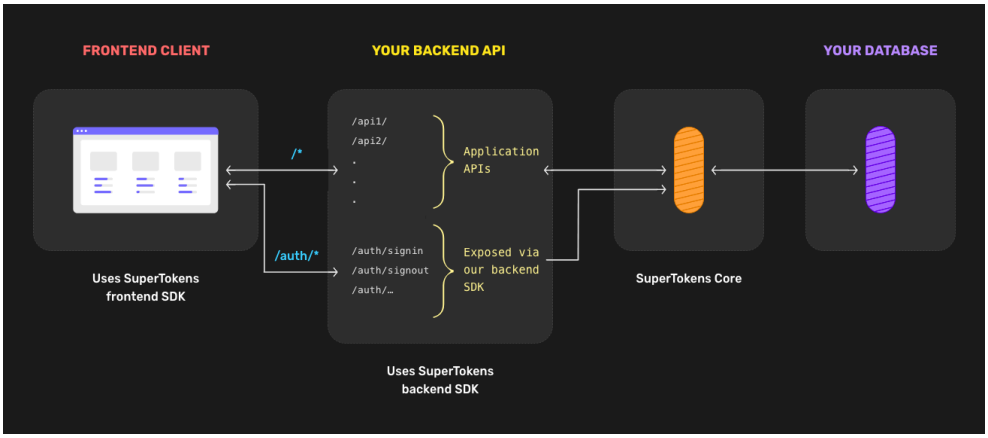
Ačkoliv SuperTokens je stále aktivní, množství příspěvků je výrazně nižší než u jiných nástrojů. Zároveň jak celkově, tak za poslední rok dominuje pouze jeden přispěvatel. Celkově za poslední rok přispělo 12 lidí, z čehož 10 pouze jedním nebo dvěma commity. [68] Na druhou stranu má projekt SuperTokens hodně GitHub hvězd.

3.3.12 Logto

Přehled vlastností k této aplikaci je dostupný v tabulce: 3.9.

Logto se prezentuje jako open-source alternativa k Auth0. Hned v úvodu je zmíněno, že poskytují bezpečnou a flexibilní infrastrukturu, možnost přizpůsobení přihlašovací stránky a robustní administraci. [69]

Dále je interaktivně ukázáno, jak je možné aplikaci jednoduše spustit pomocí Docker compose viz 1. Spuštění bylo vyzkoušeno a proběhlo bez problému.



Obrázek 3.16: SuperTokens – Architektura

Obrázek 3.17: SuperTokens – graf počtu commitů

Administrace na úvod zobrazí uživateli několik bodů, které by měl splnit pro inicializaci autorizační služby. Jako jeden z bodů je zmíněné přizpůsobení přihlašovací stránky. Grafické přizpůsobení umožňuje upravit logo, přidat krátký text a změnit barvu tlačítka. Takové nastavení je kompromis mezi jednoduchostí a přizpůsobitelností. Oproti ZITADEL ale Logto neumožňuje uzpůsobit přihlašovací stránku různým organizacím.

Dalším bodem průvodce je přidání první aplikace, která bude instanci Logto používat pro přihlašování uživatelů. I samotné přidání aplikace je interaktivní. Vývojář si vybere technologie, ve kterých vytváří aplikaci, a průvodce ho provede jak nastavením, tak i integrací.

Logto podobně jako SuperTokens podporuje role společně s oprávněními. Rozdíl je ovšem v tom, že v SuperTokens bylo oprávnění reprezentováno pouze textem. Oproti tomu v Logto je potřeba si nejprve založit API zdroj, k němu vytvořit oprávnění a ty následně přidělit k rolím. Tento způsob je přehlednější, jelikož v případě více různých API je možné rozlišit, které oprávnění patří ke kterému zdroji.

```
curl -fsSL \
https://raw.githubusercontent.com/logto-io/logto/HEAD/docker-compose.yml \
| TAG=prerelease docker compose -p logto -f - up
```

Listing 1: Logto quickstart [69]

Tabulka 3.9: Logto – přehled vlastností

vlastnost	stav
web	`https://logto.io/`
dokumentace	`https://docs.logto.io/`
sandbox	nenalezeno
self-hosted	ANO
cloud-native	nenalezeno
nasazení	Docker
role	ANO (+ oprávnění) [70]
skupiny	NE
přizpůsobení UI	ANO [69]
protokoly	OAuth 2.0, OIDC, SAML [71]
cena	zdarma, MPL 2.0
jazyk implementace	TypeScript
open-source	ANO
podporované db	PostgreSQL

Kromě rolí můžeme uživatelům přiřadit další metadata ve formátu JSON, což je flexibilnější než metadata například u Keycloak. Skupiny podporovány nejsou, ale pravděpodobně by šlo do určité míry použít právě metadata.

Nepodařilo se najít žádné informace na oficiálních ani neoficiálních zdrojích, které by potvrdily, zda je produkt připraven k horizontálnímu škálování. Nicméně je použitý refresh a access JWT token, což nasvědčuje, že server nepotřebuje udržovat v paměti informace o přihlášených uživatelích.

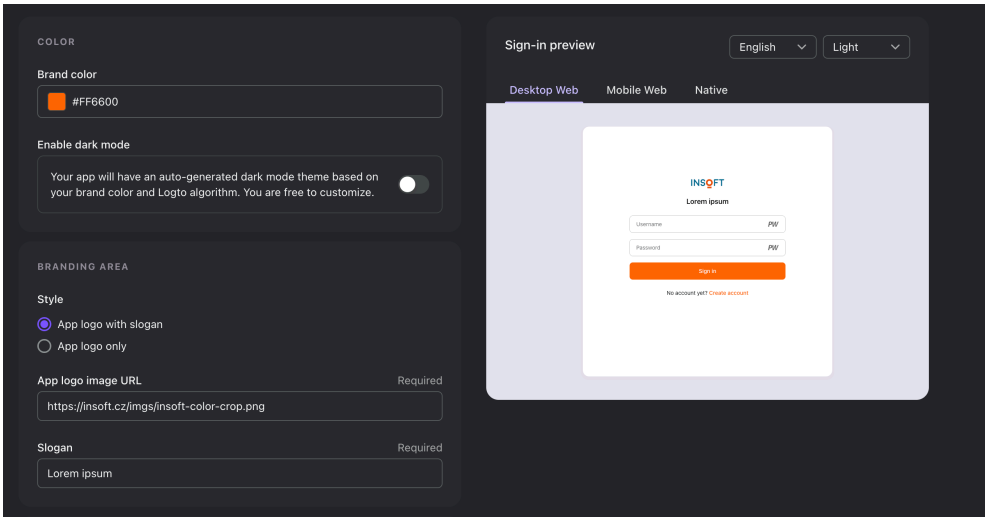
Jako méně podstatná výhoda Logto je přehledná a jednoduchá dokumentace a administrace.

Logto je jeden z nejmladších projektů. Vznikl v roce 2021 a za poslední rok množství příspěvků výrazně vzrostlo. Přispělo 33 lidí, z čehož 9 přispívalo průběžně. [10] Ačkoliv je projekt velmi mladý, má ve srovnání s konkurenčními nástroji velké množství hvězdiček.

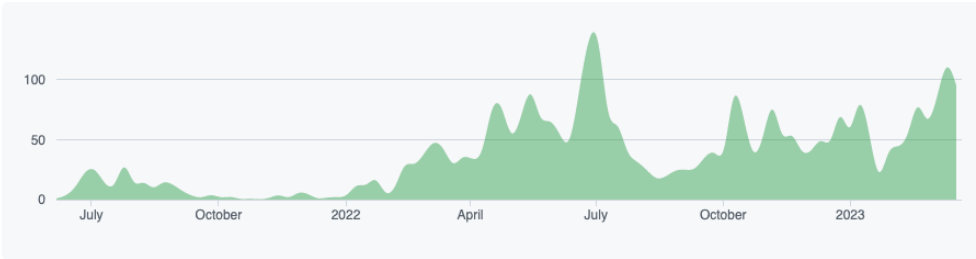
3.3.13 LoginRadius

Přehled vlastností k této aplikaci je dostupný v tabulce: 3.10.

LoginRadius je proprietární produkt. Nasazení je možné buď jako cloud řešení nebo „on premise“. Cena se odvíjí od velikosti instalace a je nutné se na ní ptát ad hoc obchodního oddělení. Tedy službu není možné nasadit jako



Obrázek 3.18: Logto – přizpůsobení přihlašovací stránky



Obrázek 3.19: Logto – graf počtu commitů [10]

self-hosted aplikaci bez podpory poskytovatele. Z toho důvodu nevyhovuje základním požadavkům a nebude hlouběji zkoumána.

3.3.14 authentik

Přehled vlastností k této aplikaci je dostupný v tabulce: 3.11.

Služba authentik je opět bezplatná a open-source. Oproti Logto není dokumentace a administrace tak přehledná a intuitivní. Spuštění testovací instance pomocí Docker compose proběhlo bez obtíží. Docker compose spustí 4 služby: PostgreSQL databázi, Redis databázi, authentik server a authentik worker, který slouží pro vykonání dlouhotrvajících úloh. [72]

Je možné vytvářet skupiny a přiřazovat do nich uživatele. Skupinám i uživatelům je možné přiřazovat metadata ve formátu YAML nebo JSON. Nejsou ale podporovány role.

Tabulka 3.10: LoginRadius – přehled vlastností

vlastnost	stav
web	`https://www.loginradius.com/`
dokumentace	`https://www.loginradius.com/docs/`
sandbox	nenalezeno
self-hosted	nenalezeno
cloud-native	ANO
nasazení	nenalezeno
role	nenalezeno
skupiny	nenalezeno
přizpůsobení UI	nenalezeno
protokoly	nenalezeno
cena	dle nabídky obchodního oddělení
jazyk implementace	nenalezeno
open-source	nenalezeno
podporované db	nenalezeno

Authentik umožňuje velmi flexibilně nastavit jak budou probíhat jednotlivé kroky různých procesů jako například přihlášení. Toto nastavení se dělá pomocí tzv. flows. [73] Příklad flow pro smazání uživatele může být vidět na ukázce kódu 2.

V oficiálních zdrojích nebyly nalezeny žádné informace o horizontálním škálování. Podpora bude potřebovat případné ověření ve kapitole ověřování vlastností.

authentik – zajímavé funkcionality:

1. U každé aplikace je možné v administraci otestovat, zdali do ní má uživatel přístup podle aktuálního nastavení.

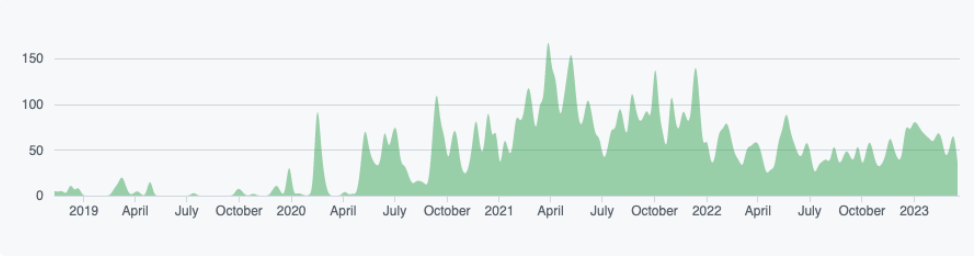
Authentik má od roku 2020 velké množství příspěvků a není pozorován výrazný pokles. Velkou většinu commitů ale vytvořil pouze jeden člověk. Z toho je patrné, že celý projekt silně závisí pouze na jednom vývojáři. To je velká nevýhoda. Celkový počet přispěvatelů za poslední rok je 79.

3.4 Kvantifikace

V této části budou informace seskupeny a ohodnoceny v tabulkovém editoru Google Sheets s cílem porovnat nástroje mezi sebou. Vzhledem k tomu, že každá organizace může mít odlišné požadavky na funkcionality, jsou parametrům přiděleny váhy. V případě použití tabulek pro vlastní účely je doporučené nastavit váhy a ohodnocení dle svých preferencí. V takovém případě je nutné si vytvořit kopii tabulky.

```yaml
version: 1
metadata:
  labels:
    blueprints.goauthentik.io/instantiate: "false"
  name: Example - User deletion
entries:
  - identifiers:
      slug: default-unenrollment-flow
    model: authentik_flows.flow
    id: flow
    attrs:
      name: Default unenrollment flow
      title: Delete your account
      designation: unenrollment
      authentication: require_authenticated
  - identifiers:
      name: default-unenrollment-user-delete
    id: default-unenrollment-user-delete
    model: authentik_stages_user_delete.userdeletestage
    attrs: {}
  - identifiers:
      target: !KeyOf flow
      stage: !KeyOf default-unenrollment-user-delete
      order: 10
    model: authentik_flows.flowstagebinding
```

Listing 2: authentik – Flow pro smazání uživatele



Obrázek 3.20: authentik – graf počtu commitů

Tabulka 3.11: authentik – přehled vlastností

vlastnost	stav
web	`https://goauthentik.io/`
dokumentace	`https://goauthentik.io/docs/`
sandbox	nenalezeno
self-hosted	ANO [72]
cloud-native	nenalezeno
nasazení	Docker compose, Kubernetes
role	NE
skupiny	ANO, hierarchické
přizpůsobení UI	ANO, v administraci, per tenant [74]
protokoly	OAuth 2.0, OIDC, SAML 2.0, LDAP [75]
cena	zdarma, MIT Licence
jazyk implementace	Python [75]
open-source	ANO
podporované db	PostgreSQL

Tabulky v následujících obrázcích jsou dostupné v interaktivní podobě na odkazu `https://tinyurl.com/2p9x23ak`.

První tabulka 3.21 zobrazuje souhrn všech nástrojů a stručný výpis jejich vlastností. Zaznamenána je cena a otevřenost kódu, možnosti nasazení a škálování, dále podpora protokolů pro federaci identit, možnost přizpůsobení vzhledu přihlašovací stránky, jazyk implementace a podporované databáze. Informace o vlastnostech jsou kvantifikovány v sumarizační tabulce 3.23.

	zdarma	open source	self hosted	cloud-native	LDAP	OIDC (relying party)	Kerberos	SAML (relying party)	role	skupiny	integrované přizpůsobení UI	deploy	jazyk	databáze
Auth0	NE	NE	NE								ANO v administraci	SaaS		
Authorizer	ANO	ANO	ANO		NE	NE	NE	NE	ANO	NE	ANO v administraci (logo + text) nebo vlastní form pomocí lib	Railway Heroku, Render, binární applikace, Kubernetes	Go	MongoDB, Cassandra, PostgreSQL, Couchbase, ...
FusionAuth	ANO (CE)	NE	ANO		NE	ANO	NE	ANO	ANO	ANO	ANO v administraci (šablony)	deb, rpm, docker, Kubernetes	Java	MySQL, Postgres,
KeyCloak	ANO	ANO	ANO		NE	ANO	ANO	ANO	ANO (+ vlastnosti)	ANO (hierarchické)	NE pouze ručně pomocí šablonových souborů	JAR, Docker, Podman, Kubernetes,	Java	mariadb mssql mysql oracle postgres
Pomerium														
Authelia	ANO	ANO	ANO	ANO bezestavovost	ANO	NE (pouze provider)	NE	NE	NE	ANO		binárky, Linux balíčky, Docker, Kubernetes	Go	MySQL, SQLite3, Postgres,
Ory	NE											SaaS		
Ory Kratos	ANO	ANO	ANO	ANO horizontální škálování bezestavovost	NE	ANO	NE	NE	NE (pouze s Ory Keto)	NE (pouze s Ory Keto)	NE neobsahuje UI	Kubernetes, Google Cloud, AWS,	Go	SQLite, Postgres, MySQL, CockroachDB,
ZITADEL	ANO	ANO	ANO	ANO high availability bezestavovost	NE	ANO	NE	NE	ANO	NE	ANO v administraci každá organizace zvlášť logo, text, barvy, fonty	Linux, MacOS, Docker compose, Knative,	Go Angular	cockroachdb postgres (BETA)
Janssen Project	ANO	ANO	ANO	ANO horizontální škálování hight availability	NE	ANO	NE	NE			NE pouze ručně pomocí šablonových souborů	Linux balíčky, Docker, Kubernetes, Amazon EKS,	Java	RDMS Erwin Table, LDAP, Couchbase, MySQL, Spanne,
SuperTokens	ANO	ANO	ANO		NE	ANO	NE	ANO	ANO (+ oprávnění)	NE	NE pouze s vlastní FE částí systému	binárky, Docker, Kubernetes	Java	MySQL, Postgres,
Logto	ANO	ANO	ANO		NE	ANO	NE	ANO	ANO (+ oprávnění)	NE	ANO v administraci	Docker	TypeScript	Postgres
LoginRadius	NE		NE											
authentik	ANO	ANO	ANO		ANO	ANO	NE	ANO	NE	ANO (hierarchické)	ANO v administraci každá organizace zvlášť logo, favicon, krásný text	Docker compose, Kubernetes	Python	Postgres

Obrázek 3.21: Vlastnosti prozkoumaných nástrojů (`https://tinyurl.com/2p9x23ak`)

Tabulka 3.22 zobrazuje kvalitu komunity kolem jednotlivých nástrojů. O některých nástrojích chybí informace, jelikož nemají otevřený zdrojový kód.

V horní části jsou zobrazeny konkrétní údaje o GitHub repositářích a počet otázek na StackOverflow. Spodní část tyto hodnoty normalizuje podle maxima v daném sloupci. Sloupec „poslední push“ je vyloučen, jelikož je vidět, že všechny nástroje jsou stále aktivní a tudíž nemá smysl je na základě této informace srovnávat.

	Data ke dni 23.3.2023									
	hvězdy	**fork**	**sledující**	**poslední push**	**open issues**	**vytvořeno**	**commitů poslední rok**	**GitHub příspěvky**	**SO questions**	
Authorizer	945	73	945	2023/03/07	58	2021/06/02	470	3,5		
FusionAuth									312	
Keycloak	15346	5234	15346	2023/03/23	1820	2013/07/02	2885	1	15722	
Authelia	15837	894	15837	2023/03/23	107	2016/12/07	1534	1	29	
Ory Kratos	8322	747	8322	2023/03/23	251	2018/05/29	606	3	144	
ZITADEL	2890	159	2890	2023/03/23	366	2020/03/16	693	2	3	
Janssen Project	177	47	177	2023/03/23	325	2020/11/03	2209	2	0	
SuperTokens	8293	293	8293	2023/03/23	98	2020/01/05	108	4	33	
Logto	5436	200	5436	2023/03/23	70	2021/06/19	3332	2,5		
LoginRadius										
authentik	3419	274	3419	2023/03/23	288	2019/12/30	2687	3	11	
										Celkem
Authorizer	0,05	0,01	0,05		0,00	0,01	0,89	0,17	0,00	1,16
FusionAuth									0,02	0,01984480346
Keycloak	0,97	1,00	0,97		1,00	1,00	0,14	1,00	1,00	7,08
Authelia	1,00	0,16	1,00		0,03	0,57	0,56	1,00	0,00	4,32
Ory Kratos	0,52	0,13	0,52		0,11	0,38	0,85	0,33	0,01	2,86
ZITADEL	0,17	0,02	0,17		0,17	0,16	0,82	0,67	0,00	2,19
Janssen Project	0,00	0,00	0,00		0,15	0,08	0,35	0,67	0,00	1,24
SuperTokens	0,52	0,05	0,52		0,02	0,18	1,00	0,00	0,00	2,29
Logto	0,34	0,03	0,34		0,01	0,00	0,00	0,50	0,00	1,21
authentik	0,21	0,04	0,21		0,13	0,18	0,20	0,33	0,00	1,31

Obrázek 3.22: Hodnocení komunity prozkoumaných nástrojů (`https://tinyurl.com/2p9x23ak`)

Poslední souhrnná tabulka 3.23 kombinuje informace z předchozích tabulek. Z první tabulky jsou vlastnosti převedeny na číslo. Obvykle 1 pro ANO a 0 pro NE, v případech sloupců „deploy“, „jazyk implementace“ a „databáze“ je hodnocení vysvětleno v poznámce buňky a v popisu jednotlivých sloupců níže. Sloupec „komunita“ je importovaný z tabulky 3.22.

free Produkt je dostupný zdarma.

open-source Produkt má otevřený zdrojový kód.

self hosted Produkt lze provozovat svépomocí.

cloud-native Produkt o sobě tvrdí, že má podporu pro cloud prostředí.

LDAP Podpora protokolu LDAP.

OIDC (relying party) Podpora protokolu OIDC jako „relying party“.

Kerberos Podpora protokolu Kerberos.

SAML (relying party) Podpora protokolu SAML jako „relying party“.

role Podpora rolí:

0 – nepodporuje,

1 – podporuje,

2 – umožňuje rolím přiřadit vlastnosti/oprávnění.

skupiny Podpora skupin.

integrované přizpůsobení UI Možnost změnit vzhled přihlašovací stránky:

0 – UI není integrované nebo nelze upravit pomocí administrace,

1 – UI lze upravit,

2 – UI lze upravit každé organizaci zvlášť.

deploy Možnosti spuštění aplikace. Jako metriku použiji součet obodovaní jednotlivých způsobů nasazení, které služba nabízí:

0 - SaaS

1 - PaaS

1 - zdrojové soubory

2 - sestavená aplikace

3 - Linux instalační balíčky

4 - Docker obraz

5 - Kubernetes

jazyk implementace Pro každého může být jinak významný jiný jazyk. Hodnotu je možné změnit dle vlastních preferencí.

V tomto případě je pro ukázku vyplněna hodnota *1* pokud je aplikaci buď zkompilovaná nebo spustitelná v běhovém prostředí pro Python. (Dle požadavků INSOFT s.r.o.)

databáze Pro každého může být jinak významná jiná databáze. Hodnotu je možné změnit dle vlastních preferencí.

V tomto případě reprezentuje hodnota počet podporovaných databází.

přehlednost dokumentace Subjektivní hodnocení webových stránkách a dokumentace. Přehlednost dokumentace nástrojů je důležitá pro vývojáře a administrátory při instalaci a správě.

komunita Importovaná hodnota z listu „Komunita“.

	Σ bodů	free	open source	self hosted	cloud-native	LDAP	OIDC (relying party)	Kerberos	SAML (relying party)	role	skupiny	integrované přizpůsobení UI	deploy	jazyk implementace	databáze	přehlednost dokumentace	komunita
váha		10	10	10	10	10	10	10	10	10	10	10	10	10	1	2	1
Keycloak	228	1	1	1		0	1	1	1	2	2	0	11	0	5	3	7,08
Authelia	227	1	1	1	1	1	0	0	0	0	1		14	1	3	5	4,32
ZITADEL	222	1	1	1	1	0	1	0	0	1	0	2	11	1	2	9	2,19
authentik	216	1	1	1		1	1	0	1	0	2	2	9	1	1	7	1,31
SuperTokens	200	1	1	1		0	1	0	1	2	0	0	11	0	2	8	2,29
FusionAuth	193	0,5	0	1		0	1	0	1	1	1	1	12	0	2	3	0,02
Authorizer	167	1	1	1		0	0	0	0	1	0	1	8	1	14	6	1,16
Janssen Project	166	1	1	1	1	0	1	0	0			0	11	0	5	0	1,24
Logto	147	1	1	1		0	1	0	1	2	0,5	1	4	0	1	10	1,21
Ory Kratos	131	1	1	1	1	0	1	0	0	0	0	0	5	1	4	7	2,86
Auth0	10	0	0	0								1	0				
Pomerium	0																
Ory	0	0											0				
LoginRadius	0	0		0													

Obrázek 3.23: Souhrnná kvantifikační tabulka vlastností (`https://tinyurl.com/2p9x23ak`)

3.5 Závěr analýzy

V první fázi analýzy byly vyhledány nástroje podle klíčových slov „správa identity, autorizace, autentizace“. Pokud byly dle jejich popisu vhodné pro tuto rešerši, byly hlouběji prozkoumány zejména na oficiálních webových stránkách a v dokumentacích. Byl obvykle zjištěn koncept, cíl a vlastnosti jednotlivých nástrojů.

Nástroje a jejich vlastnosti byly následně vypsány do interaktivních tabulek s cílem umožnit jejich porovnání. Vlastnosti byly také kvantifikovány a na základě této kvantifikace byla zjištěna dominance nástrojů FusionAuth, Keycloak, Authelia, ZITADEL, SuperTokens a authentik.

Kapitola 4

Přizpůsobení srovnávacích tabulek pro potřeby firem

V této kapitole jsou použity srovnávací tabulky z provedené rešerše v předchozí kapitole. Tyto tabulky mají upravené váhy a hodnocení jednotlivých parametrů na základě potřeb obou firem. Po upravení bodování je pro každou firmu vybrán vhodný zástupce, který bude v další kapitole experimentálně integrován do ukázkové aplikace pro ověření nalezených vlastností.

4.0.1 Požadavky INSOFT s.r.o.

Firma INSOFT s.r.o. vyvíjí softwarové řešení pro call centra. Tento systém, pojmenovaný UCS, je zpravidla nasazen „on-premise“, k čemuž INSOFT nabízí i správu. Každý zákazník má vlastní instanci systému UCS a není ovlivňován zátěží jiných zákazníků. Cílovým klientem jsou středně velká call centra.

Jelikož je UCS složené z několika klientských aplikací a služeb, vznikla potřeba sjednocení přihlášení jak z technických důvodů, tak kvůli uživatelské přívětivosti.

Na základě informací o systému UCS a konzultací vzniklo několik následujících požadavků na hledanou IdP službu:

Podpora mikroservisní architektury Jak už bylo zmíněno, UCS se skládá z několika služeb, proto je potřeba, aby IdP umělo zajistit jednotné přihlášení pro všechny.

Spustitelnost na OS Ubuntu bez závislostí na prostředí Jelikož se INSOFT snaží minimalizovat množství závislostí, preferuje možnost spustit aplikaci buď jako zkompilovanou aplikaci pro OS Linux nebo v runtime prostředí pro Python, které INSOFT již používá.

Bezestavová autentizace a autorizace Tato vlastnost umožňuje autorizaci kontrolovat přímo na koncových službách bez nutnosti autorizační proxy. Z hlediska architektury UCS je toto řešení vhodnější.

Revokace refresh tokenu V případě rozhodnutí firmy o zneplatnění přístupu zaměstnance k systému je díky této funkcionalitě společně s nastavením délky životnosti access tokenu znemožnit přístup obvykle v řádu jednotek minut.

Podpora RBAC, ACL, skupiny INSOFT v tuto chvíli používá komplexní mechanismus oprávnění. Každý uživatel má nastavenou roli, která má různá nastavení včetně možnosti nastavení přístupů pro každou uživatelskou skupinu zvlášť. Navíc jsou skupiny hierarchické a umožňují dědičnost. Ze zkoumaných nástrojů v předchozí kapitole neměl žádný z nástrojů totožný systém oprávnění. Z toho důvodu se tento požadavek zjednoduší na role, které jsou systému nejblíže.

Podpora přihlášení pomocí jiných IdP (federace) Systém UCS lze propojit s jinými systémy jako například systém pro zpracování kontaktů. Zároveň je potřeba umožnit zákazníkům přihlašování pomocí jejich již existujících účtů v jejich systému. Pro tyto účely je potřebná podpora federace identit. V aktuálním stavu podporuje UCS protokol Kerberos a umožňuje uživatelům přihlášení pomocí účtu, na který jsou přihlášení na svém počítači. Po konzultaci s INSOFT je možné použít i cestu přes protokol OIDC.

Podpora PostgreSQL V této chvíli systém UCS používá databázový systém PostgreSQL. Je vhodné aby autentizační a autorizační služba uměla využít této již nasazené databáze.

4.0.2 Požadavky Stratox Cloud Native s.r.o.

Firma Stratox Cloud Native s.r.o. vytváří platformu CodeNOW pro nasazování a vydávání softwaru, která vývojářům umožňuje soustředit se pouze na vývoj. [76]

V této platformě si uživatel může připravit prostředí pro nasazení aplikace (např. produkce a test), tzv. tech stack pro kompilaci a sestavení jednotlivých komponent aplikace (např. React + npm nebo php + composer) a spouštěcí konfiguraci. Po nasazení je možné aplikace i monitorovat.

Kromě vydání a nasazení svého produktu může uživatel platformy využít i nasazení připravené IdP služby, na kterou napojí svůj software. Taková služba by měla mít pro kompatibilitu s touto platformou následující vlastnosti:

Podpora mikroservisní architektury CodeNOW se soustředí na provozování systémů rozdělených do mikroslužeb. Jednotlivé služby pak umož-

ňuje škálovat. I IdP služba by měla být schopna v takové architektuře fungovat.

Cloud-native kompatibilita Aplikace by měla být navržena a připravena pro běh v distribuovaném cloudovém prostředí.

Bezvýpadkové škálování při nasazení v Kubernetes Platforma pojmenovaná CodeNOW je navržena pro umožnění horizontálního škálování služeb, proto by i IdP software měl podporovat škálování. Jedině tak dokáže CodeNOW zajistit slíbené funkcionality.

Snadnost/možnost přizpůsobení UI login page Jelikož výhodou nasazení IdP pomocí CodeNOW je, že vývojář má IdP tzv. „out of the box“, není pro CodeNOW vhodná aplikace, která pro přizpůsobení přihlašovací stránky vyžaduje zásah do zdrojových souborů jako například Keycloak. Stratox Cloud Native s.r.o. preferuje možnost přizpůsobit UI pomocí administrace IdP služby.

4.0.3 Přizpůsobení kvantifikace pro Stratox Cloud Native s.r.o.

Jelikož firma Stratox Cloud Native s.r.o. vyžaduje zejména podporu pro běh v cloudovém prostředí a integrované přizpůsobení UI přihlašovací stránky, nebylo třeba manipulovat konkrétním ohodnocením. Stačilo v tabulce 4.1 zvednout váhy u odpovídajících sloupců „cloud-native“ a „integrované přizpůsobení UI“. Váha byla postupně zvyšována, dokud se na neprojevila na výsledcích. Zvyšování ustalo na hodnotě 30. Pokud by byla nastavena příliš vysoká váha, mohlo by dojít ke ztrátě významu ostatních parametrů. Tím by se nástroj, který splňuje požadavek, mohl dostat dopředu, i když zaostává v ostatních vlastnostech.

Po nastavení vah byla tabulka seřazena podle výsledného počtu bodů ve sloupci „Σ bodů“. Po seřazení je možné vidět, že první čtyři pozice obsadily nástroje ZITADEL, authentik, Authelia a Keycloak.

Ačkoli nástroj Keycloak získal mnoho bodů díky ostatním vlastnostem, nedostatečně splňuje požadavek „Snadnost/možnost přizpůsobení UI login page“ 4.0.2. Nevhodnost tohoto nástroje byla i potvrzena na konzultaci s firmou.

U nástroje Authelia není uvedeno, zda je možné přizpůsobit vzhled přihlašovací stránky. To znamená, že nelze potvrdit ani vyvrátit tuto vlastnost na základě dokumentace. Nicméně i to je důvod k vyloučení tohoto nástroje, protože jiné nástroje tuto možnost nabízejí a mají ji zdokumentovanou.

Nástroj Authentik podporuje úpravu vzhledu, ale nemá zdokumentovanou podporu pro běh v cloudovém prostředí. Z toho důvodu také nesplňuje požadavky.

Na prvním místě se umístil nástroj ZITADEL, který v dokumentaci popisuje možnost spustit jej v Kubernetes. Zároveň umožňuje upravovat vzhled přihlášení dokonce pro každou organizaci zvlášť. Navíc je nástroj zdarma s otevřeným zdrojovým kódem, což je na základě konzultace se Stratox pozitivní vlastnost pro tuto firmu.

Vybraný nástroj: ZITADEL

	Σ bodů	free	open source	self hosted	cloud-native	LDAP	OIDC (relying party)	Kerberos	SAML (relying party)	role	skupiny	integrované přizpůsobení UI	deploy	jazyk implementace	databáze	přehlednost dokumentace	komunita
váha		10	10	10	30	10	10	10	10	10	10	30	10	0	1	2	1
ZITADEL	272	1	1	1	1	0	1	0	0	1	0	2	11	1	2	9	2,19
authentik	246	1	1	1		1	1	0	1	0	2	2	9	1	1	7	1,31
Authelia	237	1	1	1	1	1	0	0	0	0	1		14	1	3	5	4,32
Keycloak	228	1	1	1		0	1	1	1	2	2	0	11	0	5	3	7,08
FusionAuth	213	0,5	0	1		0	1	0	1	1	1	1	12	0	2	3	0,02
SuperTokens	200	1	1	1		0	1	0	1	2	0	0	11	0	2	8	2,29
Janssen Project	186	1	1	1	1	0	1	0	0			0	11	0	5	0	1,24
Authorizer	177	1	1	1		0	0	0	0	1	0	1	8	1	14	6	1,16
Logto	167	1	1	1		0	1	0	1	2	0,5	1	4	0	1	10	1,21
Ory Kratos	141	1	1	1	1	0	1	0	0	0	0	0	5	1	4	7	2,86
Auth0	30	0	0	0								1	0				
Pomerium	0																
Ory	0	0											0				
LoginRadius	0	0		0													

Obrázek 4.1: Souhrnná kvantifikační tabulka vlastností - Stratox Cloud Native s.r.o. (`https://tinyurl.com/2p9x23ak`)

4.0.4 Přizpůsobení kvantifikace pro INSOFT s.r.o.

Pro zdůraznění potřeb byla upravena souhrnná tabulka 3.23. Byla zvýšena váha u sloupců „OIDC“, „role“, „skupiny“ a „jazyk implementace“ a snížena u sloupců „cloud-native“, „LDAP“, „Kerberos“, „SAML“, „integrované přizpůsobení UI“. Dále bylo změněno hodnocení sloupce „databáze“, kde namísto počtu podporovaných databází mají bod takové nástroje, které podporují databázi PostgreSQL, kterou INSOFT s.r.o. používá. Dále byl upraven sloupec „deploy“, kde byly odebrány body za kontejnerizaci a možnost provozování v PaaS, jelikož tyto možnosti nasazení nejsou pro INSOFT s.r.o. významné.

Poté byla tabulka seřazena podle výsledného počtu bodů ve sloupci „Σ bodů“. Po seřazení se opět v prvních 4 pozicích objevily nástroje Keycloak, authentik, a ZITADEL. Namísto Authelia se ale tentokrát na 4. pozici dostala aplikace Logto.

Služba ZITADEL už byla vybrána pro Stratox. Zároveň se dobře umístila i pro INSOFT. Nicméně nepodporuje skupiny, které jsou pro INSOFT potřeba.

Služba authentik podporuje skupiny, ale naopak nepodporuje role, které jsou pro firmu důležitější než skupiny.

Keycloak a Logto se liší v podpoře skupin. Logto nemá vestavěnou podporu, ale je možné použít uživatelská metadata pro přiřazení uživatelů do skupin. Logto naopak oproti Keycloak umožňuje přizpůsobit vzhled pomocí administrace. Výběr mezi těmito službami byl konzultován s firmou INSOFT a nakonec bylo rozhodnuto pro nástroj Logto, který je mladší a modernější. Zároveň je pro firmu vhodnější programovací jazyk TypeScript, ve kterém je nástroj Logto napsán, v případě potřeby rozšíření nástroje.

Vybraný nástroj: Logto

	Σ bodů	free	open source	self hosted	cloud-native	LDAP	OIDC (relying party)	Kerberos	SAML (relying party)	role	skupiny	integrované přizpůsobení UI	deploy	jazyk implementace	databáze	přehlednost dokumentace	komunita
váha		10	10	10	1	1	30	1	1	30	30	1	10	30	1	2	1
Keycloak	206	1	1	1		0	1	1	1	2	2	0	1	0	1,0	3	7,08
authentik	180	1	1	1		1	1	0	1	0	2	2	1	1	1,0	7	1,31
ZITADEL	174	1	1	1	1	0	1	0	0	1	0	2	3	1	0,5	9	2,19
Logto	169	1	1	1		0	1	0	1	2	0,5	1	1	0	1,0	10	1,21
Authelia	167	1	1	1	1	1	0	0	0	0	1		6	1	1,0	5	4,32
SuperTokens	160	1	1	1		0	1	0	1	2	0	0	2	0	1,0	8	2,29
FusionAuth	144	0,5	0	1		0	1	0	1	1	1	1	3	0	1,0	3	0,02
Authorizer	135	1	1	1		0	0	0	0	1	0	1	3	1	1,0	6	1,16
Ory Kratos	119	1	1	1	1	0	1	0	0	0	0	0	1	1	1,0	7	2,86
Janssen Project	102	1	1	1	1	0	1	0	0			0	4	0	0,0	0	1,24
Auth0	1	0	0	0								1	0				
Pomerium	0																
Ory	0	0											0				
LoginRadius	0	0		0													

Obrázek 4.2: Souhrnná kvantifikační tabulka vlastností - INSOFT s.r.o. (`https://tinyurl.com/2p9x23ak`)

4.0.5 Vyřazené nástroje

LoginRadius Není self-hostingu možnost.

Ory Není self-hostingu možnost.

Pomerium Soustředí se na autentizaci pomocí zapamatovaného zařízení, ze kterého se uživatel přihlašuje a neumožňuje běžný způsob přihlášení. Je tedy nevhodný vzhledem k požadavkům.

Auth0 Není self-hostingu možnost.

Ory Kratos Neposkytuje jednotné řešení pro autentizaci a autorizaci, je potřeba použít i spolupracující služby.

Janssen Project Nepodporuje systém autorizování uživatelů (role/skupiny), ani integrované přizpůsobení UI.

Authorizer Nepodporuje ani jednu z možností federace identit.

FusionAuth Získal průměrně bodů, nemá otevřený zdrojový kód a oproti konkurenci nemá přehlednou dokumentaci.

SuperTokens Nástroj je podobný Logto. Rozdíl je v jazyku implementace a možnosti úpravy UI. Pro Stratox je nemožnost přizpůsobení přihlašovací stránky blokující. Pro INSOFT je nevhodný programovací jazyk, jelikož jeho vývojáři tento jazyk nepoužívají a nemohli by přispívat do projektu.

authentik Nepodporuje role, což je nevhodné pro INSOFT. Zároveň nepodporuje běh v distribuovaném prostředí, což je nevhodné pro Stratox.

Authelia Nepodporuje integrované přizpůsobení přihlašovací stránky. Zároveň neposkytuje bezestavovou autorizaci. Funguje jako proxy modul. Dále nepodporuje protokol OIDC v roli „relying party“.

Keycloak je silná služba s největší historií a komunitou. Nicméně pro Stratox je nevhodná z důvodu nepodporování cloud prostředí a nemožnosti upravit vzhled přihlašovací stránky. Pro INSOFT by mohl nástroj vyhovovat, nicméně po konzultaci byla zvolena alternativa Logto.

4.0.6 Vybrané nástroje:

1. ZITADEL
2. Logto

Kapitola 5

Integrace vybraných služeb do ukázkové aplikace

Pro ověření vlastností nalezených v dokumentacích projektů je v této kapitole vytvořen ukázkový systém. Tento systém simuluje jednoduchou aplikaci využívající mikroservisní architekturu. Po jeho vytvoření jsou oba vybrané nástroje postupně integrovány do ukázkové aplikace. Integrací a následným ověřením funkčnosti celého systému jsou vyzkoušeny a ověřeny jednotlivé vlastnosti.

5.1 Ukázkový systém s mikroservisní architekturou

V této sekci je představeno vytvoření ukázkového systému. Tento systém sestává ze dvou backend mikroslužeb a jedné klientské webové aplikace. Klientská aplikace komunikuje s backend službami pomocí HTTP. Do tohoto systému budou následně integrovány vybrané služby pro ověření funkčnosti.

5.1.1 Služba 1 (Python)

Jako jazyk pro první ukázkovou backend službu byl vybrán Python s knihovnou FastAPI. Tento technologický stack je používaný nejen firmou INSOFT s.r.o. Podle statistik z PyPI je týdně zaznamenáno přibližně 700 tisíc stažení knihovny.

Pro demonstraci přístupu má služba tři endpointy. Jeden veřejný, ke kterému není potřeba přihlášení. Další 2 vyžadují přihlášení a roli. Každý z neveřejných endpointů vyžaduje jinou roli pro simulaci administrátorské sekce a sekce pro supervizora, což je manažerská pozice v kontaktním centru.

Náhled na zdrojový kód služby je na ukázce kódu 3. Třída *JWTBearer* je využita vždy před zpracováním dotazu. Z dotazu si získá hlavičku *Authorization*, ze které získá token. Ten následně ověří veřejným klíčem autorizačního serveru. Zároveň ověří, že v tokenu se nachází role předaná v konstruktoru jako

```python
from fastapi import FastAPI, Depends
from fastapi.middleware.cors import CORSMiddleware
from jwt_bearer_dep import JWTBearer

app = FastAPI()

app.add_middleware(CORSMiddleware, allow_origins=["*"],
    allow_credentials=True, allow_methods=["*"],
    allow_headers=["*"],
)

@app.get("/")
def read_root():
    return {"response": "Hello world"}


@app.get("/restricted", dependencies=[Depends(JWTBearer(
    required_role="supervisor",
    role_key="urn:zitadel:iam:org:project:roles"
))])
def add_item():
    return {"response": "Hello supervisor!"}


@app.get("/admin/", dependencies=[Depends(JWTBearer(
    required_role="admin",
    role_key="urn:zitadel:iam:org:project:roles"
))])
def admin():
    return {"response": "Hello ADMIN!"}
```

Listing 3: Služba 1 – endpointy

parametr *required_role*. V případě, že tato třída nevyhodí výjimku, je dotaz předán samotné funkci na zpracování a odeslána odpověď.

5.1.2 Služba 2 (Go)

Pro další ukázkovou službu byl vybrán jazyk Go. Zvolení jiného jazyku není běžné v praxi, ale v tomto případě se zvýší diverzifikace ukázkových technologií. Jazyk Go získává na popularitě. Podle průzkumu by se ho rádo naučilo 16,41 % respondentů, čímž se v žebříčku umístil na čtvrtém místě. [77]. Zároveň je vhodným jazykem pro vývoj webových stránek. [78]

Tato služba simuluje obdobné prostředí jako služba v jazyce Python. Poskytuje 3 endpointy. Každý vyžaduje jinou potřebnou úroveň oprávnění. Hlavní část implementace je možné vidět na ukázce kódu 4.

5.1.3 Uživatelské rozhraní (React)

Pro implementaci ukázkové Frontend aplikace byla vybrána knihovna React, jelikož je opět využívána firmou INSOFT a bude tedy simulovat nasazení do této firmy. Zároveň je hojně využívána i obecně [79], a tak je vhodnou volbou i pro Stratox.

Jelikož je jeden z vybraných nástrojů ZITADEL, byl v dokumentaci vyhledán návod na propojení React aplikace s touto službou. Přímo dokumentace ZITADEL vysvětluje, jak aplikaci s přihlašováním pomocí ZITADEL zprovoznit od založení projektu po integraci s autentizačním serverem.

Po inicializaci projektu je potřeba si stáhnout knihovnu „oidc-react“. Ta se pak v aplikaci inicializuje tak, že se vytvoří konfigurační konstanta a předá se do komponenty AuthProvider, kterou se obalí zbytek aplikace. Tuto integraci je možné vidět na ukázce kódu 5. V konfiguraci je potřeba zadat url autentizační služby, clientId vygenerované v autentizační službě, způsob předání tokenu, url, kam má autentizační služba přesměrovat po přihlášení a nakonec scope. [80]

Po navštívení této webové aplikace je uživatel ihned přesměrován na přihlašovací server, kde je vybídnut k vyplnění přihlašovacích údajů. Pokud uživatel zadá správné heslo, je přesměrován zpět na URL adresu, která je vyplněna v konfiguraci. Knihovna *oidc-react* se následně postará o dokončení přihlášení získáním tokenů. Z tokenů získá informace o uživateli, které je možné následně zobrazit v aplikaci. Grafické rozhraní aplikace je vidět na obrázku 5.1

5.2 Integrace ZITADEL

Jako první je vyzkoušena služba ZITADEL. Nejprve je spuštěna v Kubernetes, následně nakonfigurována pomocí administrace. Poté je vysvětleno napojení ukázkového systému. Nakonec je ověřeno, že je možné se pomocí služby přihlašovat a služba splňuje požadavky potřebné zejména pro firmu Stratox.

5.2.1 Spuštění v Kubernetes

Pro ověření schopnosti aplikace provádět bezvýpadkové horizontální škálování je vyzkoušeno nasazení do Kubernetes. K tomuto otestování je využito „Docker Desktop“ se zapnutým Kubernetes modulem.

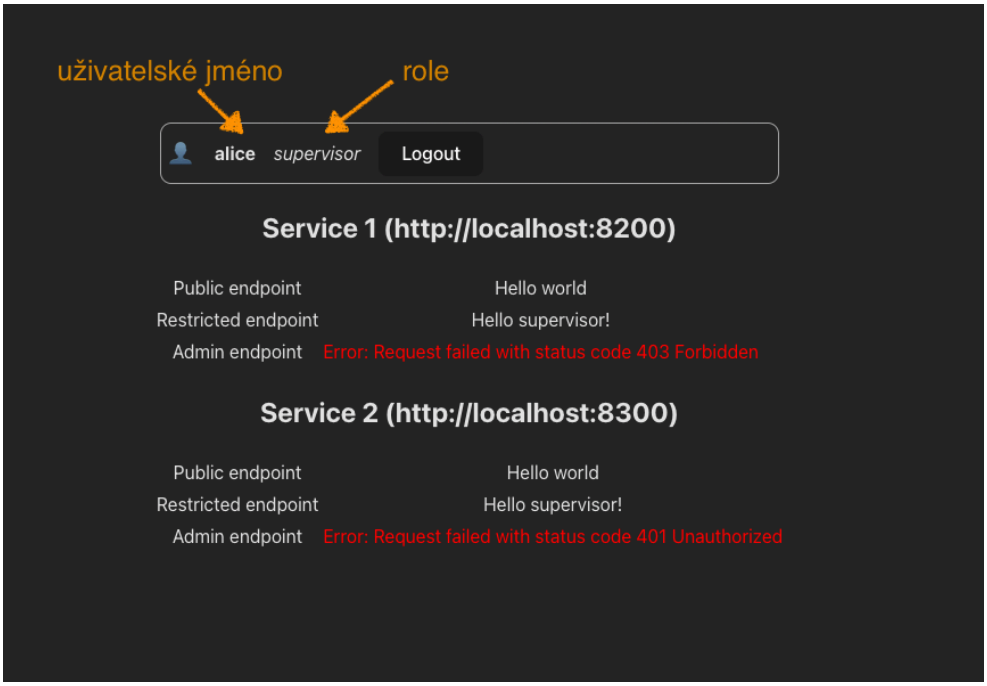
V následujících je postupováno dle dokumentace ZITADEL. Jako první je připravena databáze, kterou aplikace využije pro ukládání nastavení a uživatelů.

```
type OkResponse struct {
    Response string `json:"response"`
}

func main() {
    e := echo.New()
    e.Use(middleware.CORSWithConfig(middleware.CORSConfig{
        AllowOrigins: []string{"*"},
        AllowHeaders: []string{
            echo.HeaderOrigin, echo.HeaderContentType,
            echo.HeaderAccept, echo.HeaderAuthorization},
    }))
    e.GET("/", func(c echo.Context) error {
        return c.JSON(http.StatusOK, OkResponse{
            Response: "Hello world"})
    })
    config := echojwt.Config{ KeyFunc: getKey }
    r := e.Group("/restricted")
    {
        r.Use(echojwt.WithConfig(config))
        r.Use(roleValidator("supervisor"))
        r.GET("", func(c echo.Context) error {
            return c.JSON(http.StatusOK, OkResponse{
                "Hello supervisor!"})
        })
    }
    a := e.Group("/admin")
    {
        a.Use(echojwt.WithConfig(config))
        a.Use(roleValidator("admin"))
        a.GET("", func(c echo.Context) error {
            return c.JSON(http.StatusOK, OkResponse{
                "Hello ADMIN!"})
        })
    }
    e.Logger.Fatal(e.Start(":8300"))
}
```

Listing 4: Služba 2 – Ukázka hlavní části kódu

```
import { AuthProvider } from "oidc-react";

const oidcConfig: AuthProviderProps = {
  authority: "http://localhost:8080",
  clientId: "208924526216544400@ucs",
  responseType: "code",
  redirectUri: "http://localhost:3000",
  scope: "openid profile email offline_access metadata",
};


const App = () => (
  <AuthProvider {...oidcConfig}>
    <div className="App">
      ...
    </div>
  </AuthProvider>
)
```

Listing 5: Integrace přihlášení pomocí OIDC do React aplikace [80]



Obrázek 5.1: Ukázková aplikace – přihlášený uživatel s rolí „supervizor“

```
helm repo add cockroachdb https://charts.cockroachdb.com/
helm repo add zitadel https://charts.zitadel.com

# CockroachDB
helm install crdb cockroachdb/cockroachdb \
  --set fullnameOverride=crdb \
  --set single-node=true \
  --set statefulset.replicas=1

# ZITADEL
helm install my-zitadel zitadel/zitadel \
  --set zitadel.masterkey="MasterkeyNeedsToHave32Characters" \
  --set zitadel.configmapConfig.ExternalSecure=false \
  --set zitadel.configmapConfig.TLS.Enabled=false \
  --set zitadel.secretConfig.Database.cockroach.User.Password="..." \
  --set replicaCount=1
```

Listing 6: Spuštění ZITADEL v Kubernetes s CockroachDB [81]

ZITADEL podporuje databázi CockroachDB a PostgreSQL. CockroachDB je doporučená jelikož je vhodná pro horizontální škálování. Veškeré návody na spuštění ZITADEL v Kubernetes obsahují tuto databázi. Aplikace je tedy spuštěna v doporučené konfiguraci podle návodu 5.2.1.

Následující ukázka kódu 5.2.1 ukazuje nejprve přidání CockroachDB a ZITADEL repositářů do správce balíčků pro Kubernetes – Helm.

Následně je pomocí Helm nainstalována databáze CockroachDB. Název nainstalované služby je „crdb“ a počáteční počet replikací 1.

Nakonec je inicializována služba ZITADEL s vypnutým šifrováním. Vzhledem k tomu, že vše běží na jednom stroji a žádná ze služeb není veřejná, není šifrování potřeba. Počet replikací je nejprve nastaven také na hodnotu 1. Aplikace bude škálována později.

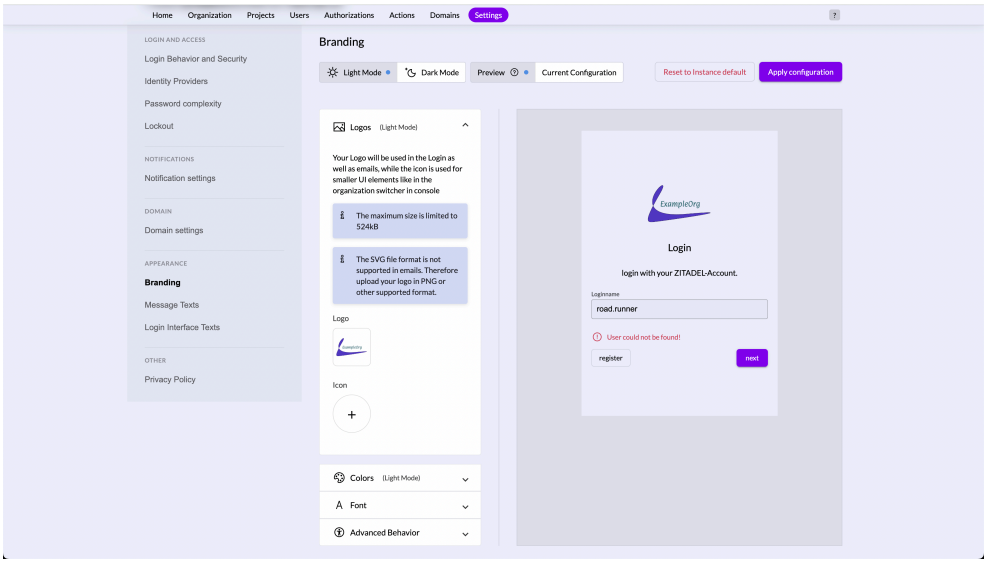
5.2.2 Konfigurace

Založení organizace *Instance → Organizations → New*

V první řadě je provedena konfiguraci organizací. Po instalaci ZITADEL již obsahuje výchozí organizaci „ZITADEL“, která obsahuje administrátorský účet. Jelikož je jeden z požadavků přizpůsobení vzhledu přihlašovací stránky a ZITADEL to umožňuje pro každou organizaci zvlášť, je založena ještě jedna organizace s názvem „ExampleOrg“.

Přizpůsobení vzhledu *Organizations → ExampleOrg → Settings → Branding*

Následně je přizpůsoben vzhled přihlašovací stránky. Pokud se bude přihlašovat uživatel z této organizace, zobrazí se na přihlašovací stránce nastavené



Obrázek 5.2: Přizpůsobení vzhledu UI organizace ExampleOrg

logo a upravené barvy. Nastavení vzhledu je vidět na obrázku 5.2. Totožné nastavení lze provést i na úrovni celé instance služby. To se pak použije jako výchozí nastavení.

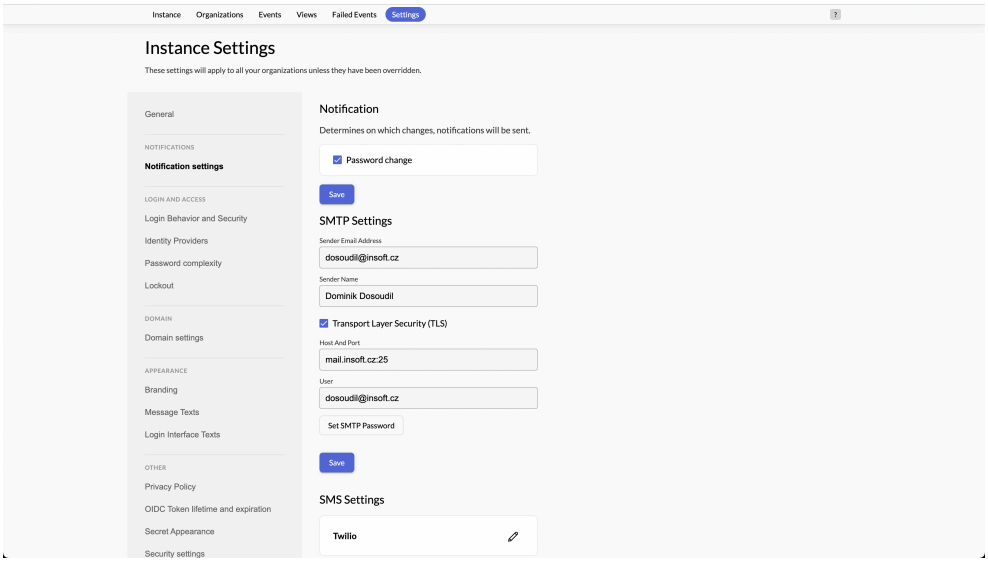
Kromě loga firmy lze nastavit i text zobrazený na různých obrazovkách procesu přihlášení/registrace/...Toto nastavení lze provést v sekci „Login Interface Texts“.

Ve výchozím nastavení funguje zvolení organizace tak, že uživatel nejprve vyplní uživatelské jméno načež ZITADEL dohledá, do které organizace uživatel patří. Na základě toho je zobrazeno přizpůsobené grafické rozhraní. Nicméně v případě, že je nějaká aplikace určená pouze pro jednu organizaci, je možné v přihlašovacím požadavku zaslat i doménu/ID organizace. Na základě toho ZITADEL zobrazí rovnou přizpůsobení vzhledu pro zaslanou organizaci a navíc umožní přihlášení pouze uživatelům z této organizace. [82]

Založení projektu *ExampleOrg → Projects → Create New Project*

Alternativní způsob přizpůsobení vzhledu již při vyplňování přihlašovacího jména je zvolení vzhledu v konfiguraci projektu. V rámci organizace „ExampleOrg“ je proto založen projekt a pojmenován „ExampleProject“. Na hlavní obrazovce projektu je konfigurační parametr *Branding Setting*, který má následující možnosti:

Nespecifikované přizpůsobení Stránka pro vyplnění uživatelského jména je bez přizpůsobení. Po vyplnění uživatelského jména se stránka přizpůsobí dle organizace, ve které je uživatel.



Obrázek 5.3: ZITADEL – nastavení SMTP

Přizpůsobení dle organizace vlastnící projekt Všechny obrazovky přihlášení budou přizpůsobené dle organizace, ve které je projekt. Na uživateli nezáleží.

Přizpůsobení dle organizace uživatele Kombinace předchozích – dokud není uživatel identifikován, je použito přizpůsobení dle organizace vlastnící projekt. Následně je přizpůsobení přepnuto na organizaci vlastnící uživatele.

Nastavení SMTP *Instance → Notification settings*

Dalším krokem konfigurace je mailový server. Při registraci nového uživatele je potřeba potvrdit jeho e-mail. Zároveň je potřeba mít dostupnou cestu například pro obnovení hesla v případě jeho zapomenutí. K tomu slouží mailový server, pomocí kterého se odešle automatický e-mail uživateli na jeho nakonfigurovanou e-mailovou adresu.

Toto nastavení nelze provést pro každou organizaci zvlášť. Tedy není možné různým organizacím odesílat e-maily z různých adres. Nicméně alespoň lze následně změnit text e-mail zprávy. V nastavení je potřeba vyplnit e-mail a jméno odesílatele, adresu a port SMTP serveru, zda se má použít TLS a nakonec uživatelské jméno a heslo pro autentizaci. Nastavení lze vidět na obrázku 5.3.

5.2.3 Uživatelé a role

Vytvoření rolí v projektu *ExampleOrg → ExampleProject → Roles*

Pro testování integrace je potřeba několik uživatelů s různými oprávněními. V první řadě jsou vytvořeny v projektu „ExampleProject“ 3 role: „user“, „supervisor“, „admin“.

Následně jsou založeni v organizaci s názvem „ExampleOrg“ tři noví uživatelé: „alice.example“, „bob.example“ a „charlie.example“. Po založení uživatelů všem přijde na vyplněnou e-mailovou adresu žádost o potvrzení registrace. Zároveň je uživatel požádán o inicializační přihlášení a nastavení hesla. Díky tomuto procesu zná heslo jen samotný uživatel. Je ale možné uživateli vyplnit i počáteční heslo administrátorem při zakládání uživatele. Uživatel je nicméně i tak následně požádán o změnu hesla.

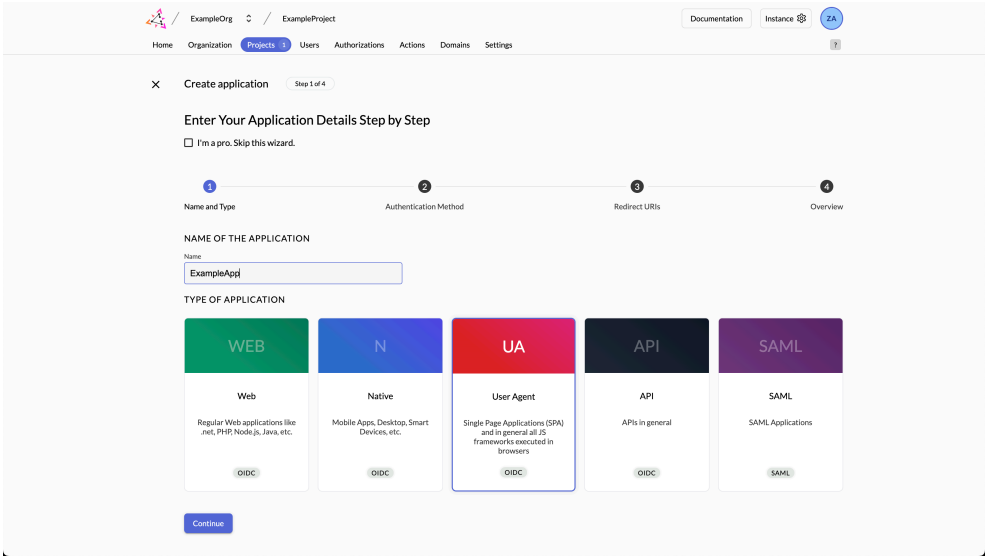
5.2.4 Registrace ukázkové aplikace

Založení aplikace *ExampleProject → New*

Jelikož je ukázková aplikace webová SPA komunikující se službami, je v ZITADEL založena aplikace typu „User Agent“ a pojmenována „ExampleApp“ viz obrázek 5.4. Jako autentizační metodu je zvolena doporučená metoda PKCE. Doporučená je z důvodu zvýšení bezpečnosti [83]. Třetí krok je nastavení URL adresy, na kterou má ZITADEL přesměrovat po dokončení přihlášení. Nakonfigurována je adresa ukázkové aplikace *http://localhost:3000*. Po potvrzení ZITADEL vygeneruje unikátní identifikátor „ClientId“, který se vyplní do dříve zmíněné konfigurace v ukázkové webové aplikaci. Po dokončení registrace je potřeba na záložce „Redirect settings“ zaškrtnout „Development mode“, který umožní provoz bez šifrovaného HTTP. V produkci by k tomuto povolení nemělo dojít, ale jelikož je toto pouze testovací instance na lokálním stroji, je to v pořádku. Dále je potřeba na záložce „Token Settings“ vybrat v parametru „Auth Token Type“ možnost „JWT“ a zaškrtnout „Add user roles to the access token“ a „User roles inside ID Token“, díky čemuž se uživatelské role vloží do tokenu. Dále je možné zaškrtnout i poslední možnost „User Info inside ID Token“, díky čemuž bude možné z ID tokenu získat základní informace o uživateli.

Nakonec je potřeba v ukázkové aplikaci představené v podsekci 5.1.3 vyplnit zmíněné „ClientId“ identifikující aplikaci (*209684131565600912exampleproject*), URL adresu autentizačního serveru (*http://localhost:8080*), typ odpovědi (*code*), adresu přesměrování po přihlášení (*http://localhost:3000*) a „scope“, který říká, k jakým informacím o uživateli je vyžadován přístup (*openid profile email offline_access urn:zitadel:iam:user:metadata*).

V obou službách 5.1.1 a 5.1.2 je potřeba vyplnit pouze adresu, na které se nachází klíče k tokenům (*http://localhost:8080/oauth/v2/keys*), kde se v tokenu nachází role (*urn:zitadel:iam:org:project:roles*) a nakonec správně syn-



Obrázek 5.4: ZITADEL – založení aplikace – krok 1

chronizovat pojmenování rolí v aplikaci a v ZITADEL (*user, supervisor, admin*).

5.2.5 Registrace Google jako IdP pro ZITADEL

V této podsekci je nastaveno a povoleno přihlášení pomocí Google účtu. Tím je ověřena možnost federace identit.

Konfigurace IdP *Instance → Settings → Identity Providers*

Pro ověření možnosti napojení na externí IdP službu a umožnění přihlášení pomocí ní je potřeba v ZITADEL nakonfigurovat externí přihlášení. Pro ukázku je možné použít například Google účet. Nejprve je potřeba ZITADEL zaregistrovat v Google Cloud konzoli, která se nachází na adrese `https://console.cloud.google.com/apis/credentials`. Při registraci je pro správné fungování potřeba vyplnit adresy přesměrování:

1. http://localhost:8080/ui/login/logint/externalidp/callback,
2. http://localhost:8080/ui/register/logint/externalidp/callback.

Po registraci aplikace se zobrazí vygenerované údaje („Client ID“ a „Client secret“), které je potřeba následně vyplnit v ZITADEL konfiguraci IdP. [84]

V konfiguraci je nutné vyplnit několik parametrů. Název je libovolný text, hodnota „Issuer“ je v případě Google přihlášení *https://accounts.google.com*.

„Client ID“ a „Client secret“ se opíše z registrace v Google Cloud konzoli jak bylo zmíněno v předešlém odstavci.

Po této konfiguraci je již možné využívat Google jako IdP a přihlašovat se pomocí Google účtu. ZITADEL tedy podporuje protokol OIDC.

5.2.6 Ověření revokace refresh tokenu

V administraci ZITADEL je možné uživatele deaktivovat, což by mělo zneplatnit i jeho refresh tokeny, aby po expiraci vydaných access tokenů odhlášen.

Pro otestování této funkcionality lze nastavit dobu platnosti access tokenu na několik desítek sekund, přihlásit se a následně pomocí administrace deaktivovat uživatele. Následující požadavek o obnovení access tokenu by měl selhat.

Po provedení tohoto testu se ukázkové aplikaci nepodařilo obnovit token a aplikace uživatele přesměrovala na přihlašovací stránku. Toto chování potvrzuje schopnost zakázání přístupu uživateli k systému.

5.2.7 Závěr

Dokumentace aplikace ZITADEL přehledně popisuje, jak aplikaci spustit pomocí Helm v Kubernetes. Toto schéma ve výchozím nastavení připraví instanci ZITADEL v režimu vysoké dostupnosti. [85]

Nastavení celé služby ZITADEL je přehledné a intuitivní. Umožňuje přizpůsobit vzhled přihlašovací stránky jak pro celou instanci služby, tak pro jednotlivé organizace (tenant). V případě umožnění přístupu do projektů z jiných organizací umožňuje nastavit, zda se přihlášení přizpůsobí na základě organizace uživatele nebo projektu. Díky tomuto rozsáhlému nastavení aplikace splňuje požadavek na integrované přizpůsobení UI.

Nakonec byl ukázkový systém nastaven na používání ZITADEL instance jako autentizační a autorizační služby, čímž byla demonstrována podpora mikroservisní architektury.

Ukázková aplikace po autentizaci získala přístupové tokeny ve formátu JWT, které ukázkové služby dokáží ověřovat pomocí veřejných klíčů i v případě dočasného výpadku ZITADEL, a tedy je splněn požadavek na bezestavovost autorizace.

Pro ukázku bylo nastaveno několik rolí a ověřeno, že je možné je zahrnout do tokenu. Díky tomu mohly ukázkové služby ověřit autorizaci požadavku.

5.3 Integrace Logto

Dle dokumentace je možné Logto spustit lokálně třemi způsoby: pomocí Docker Compose, Docker nebo npm-init. [86] Pro následné ověřování schopnosti škálování je nicméně potřeba spustit aplikaci v Kubernetes, které škálování umožňuje. Dokumentace obsahuje i informace o spuštění v Kubernetes. Je vysvět-

```
helm repo add binami https://charts.bitnami.com/bitnami
helm install postgres \
    --set auth.postgresPassword=postgres \
    binami/postgresql
```

Listing 7: Instalace PostgreSQL do Kubernetes clusteru pomocí Helm

```
kubectl run postgres-postgresql-client \
    --rm --tty -i \
    --restart='Never' \
    --namespace default \
    --image docker.io/bitnami/postgresql:15.2.0-debian-11-r21 \
    --env="PGPASSWORD=postgres" \
    --command -- psql --host postgres-postgresql -U postgres \
        -d postgres -p 5432
```

Listing 8: Připojení do PostgreSQL v Kubernetes clusteru pomocí CLI

leno, jaké je potřeba nastavit konfigurační proměnné, možnosti upgradování aplikace a nakonec podpora kontejnerizace v Kubernetes včetně informace, že je potřeba kontejnerům nastavit sdílené úložiště pro ukládání modulů pro federaci identit. Zároveň dokumentace poskytuje minimální potřebnou konfiguraci pro spuštění aplikace. [87] Nicméně pro účely závěrečné práce je potřeba provést úpravy.

5.3.1 Spuštění v Kubernetes

V první řadě je potřeba připravit databázi. Logto je kompatibilní s databázovým systémem PostgreSQL verze alespoň 14. [86]. Pro instalaci PostgreSQL do Kubernetes je použit Helm. Helm schéma je dostupné v repozitáři od organizace „Bitnami“. Pro instalaci je obdobně jako u CockroachDB potřeba přidat repozitář a následně z něho nainstalovat službu *postgres* viz ukázka kódu 7. [88].

Po instalaci je možné ověřit funkčnost přihlášením do databáze pomocí příkazu v ukázce kódu 8.

Nakonec je potřeba v databázovém systému vytvořit databázi *logto*, kterou služba využije. Databáze se vytvoří příkazem `CREATE DATABASE logto`.

Po spuštění databáze přichází na řadu spuštění Logto. V dokumentaci je připravený soubor se specifikací pro Kubernetes, nicméně tuto konfiguraci je potřeba upravit. Jelikož byla vytvořena vlastní databáze, kontejnery musí obsahovat proměnnou *DB_URL* obsahující adresu, port, přihlašovací údaje a jméno databáze. Dále byl do specifikace přidán kontejner inicializující databázi[1]. Celá specifikace je dostupná v ukázce kódu 9.

[1]Kontejner určený pro inicializaci databáze není ideální při použití v produkci, jelikož je

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: logto
  namespace: default
  labels:
    app: logto
spec:
  replicas: 1
  selector:
    matchLabels:
      app: logto
  template:
    metadata:
      labels:
        app: logto
    spec:
      volumes:
        - name: connectors
          emptyDir: {}
      initContainers:
        - image: ghcr.io/logto-io/logto
          command:
            - /bin/sh
          args:
            - '-c'
            - 'npm run cli connector add -- --official'
          name: init
          volumeMounts:
            - name: connectors
              mountPath: /etc/logto/packages/core/connectors
        - image: ghcr.io/logto-io/logto

          command:
            - /bin/sh
          args:
            - '-c'
            - 'npm run cli db seed'
          name: init-db
          env:
          - name: DB_URL
            value: "postgres://postgres:postgres@10.96.42.213:5432/logto"
      containers:
        - image: ghcr.io/logto-io/logto
          name: logto
          volumeMounts:
            - name: connectors
              mountPath: /etc/logto/packages/core/connectors
          env:
          - name: DB_URL
            value: "postgres://postgres:postgres@10.96.42.213:5432/logto"
```

Listing 9: Specifikace Kubernetes objektu pro nasazení Logto

```yaml
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  name: logto-lb
spec:
  ports:
  - port: 3001
    name: core
    protocol: TCP
    targetPort: 3001
  - port: 3002
    name: admin
    protocol: TCP
    targetPort: 3002
  selector:
    app: logto
  type: LoadBalancer
status:
  loadBalancer: {}
```

Listing 10: Specifikace Kubernetes Objektu – Logto služba

Následně je potřeba vytvořit objekt typu „služba“ pro umožnění rozesílání požadavků instancím Logto. Popis objektu typu „služba“ je dostupný v ukázce kódu 10. Jelikož Logto poslouchá na dvou portech (jádro na portu 3001 a administrace na portu 3002) [86], obsahuje specifikace přesměrování těchto portů. Dále je nastaveno, že služba bude typu „LoadBalancer“, díky čemuž bude dostupná ze vně Kubernetes [89].

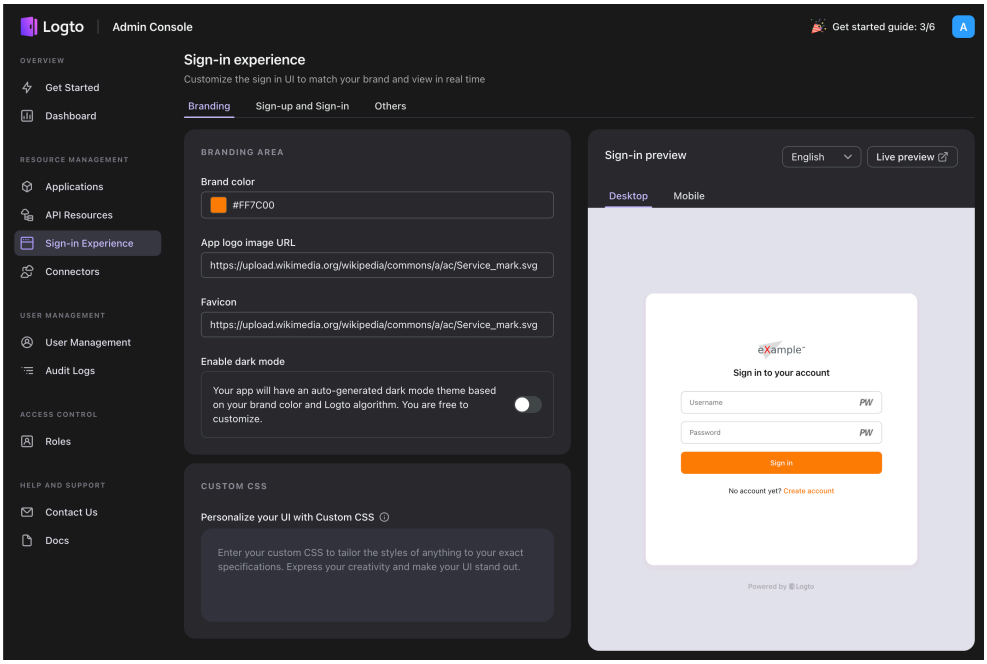
5.3.2 Konfigurace

Logto hned na úvodní stránce administrace nabízí některé možnosti konfigurace, které je možné provést pro inicializaci služby. Například integrace aplikace, přizpůsobení UI nebo přidání vlastního konektoru. Kromě těchto kroků je vhodné nastavit i SMTP server pro možnost ověření e-mailu při registraci nových uživatelů.

Přizpůsobení vzhledu *Sign-in experience → Get started / Branding*

Logto nabízí možnost úpravy hlavní barvy UI. Touto nakonfigurovanou barvou se obarví tlačítko „Přihlásit se“ a odkaz na registraci. Dále je možné vložit odkaz na obrázek, který se zobrazuje nad přihlašovacím formulářem. Způsob konfigurace obrázku pomocí URL má oproti nahrání souboru výhodu v tom, že pokud tu samou URL adresu používají například webové stránky

potřeba ho spustit při spouštění každé replikace, ale pro účely závěrečné práce je toto řešení dostatečné.



Obrázek 5.5: Logto přizpůsobení přihlašovací stránky

firmy a firma se rozhodne pro změnu loga, stačí změnit obrázek na jednom místě. Změna se promítne se i do přihlašovací služby. Nevýhoda ovšem spočívá v tom, že při mazání souboru s logem je potřeba myslet na to, že tento soubor nepoužívá pouze webová stránka, ale i autentizační služba. Dále je možné v konfiguraci zapnout tmavý režim a nakonfigurovat mu také vhodnou barvu. Kromě těchto vestavěných možností je dále umožněno vložit vlastní CSS, pomocí kterého lze přihlašovací stránku komplexněji designovat. Tím je zvýšena flexibilita úprav, nicméně je potřeba znalost této technologie.

Nastavení SMTP *Connectors* → *Email connector* → *SMTP*

Pro umožnění potvrzení e-mailu při registraci, obnově hesla nebo zaslání jednorázového verifikačního kódu je potřeba nakonfigurovat konektor k e-mail serveru. [90] V konfiguraci je potřeba nastavit adresu SMTP serveru, přihlašovací údaje, e-mailová adresa, ze které budou e-maily odesílány uživatelům a časové limity pro spojení se serverem. Dále je volitelně možné zapnout šifrování a změnit text, který bude odeslán e-mailové zprávě. Po dokončení konfigurace je možné otestovat korektnost konfigurace odesláním testovacího e-mailu.

5.3.3 Uživatelé, role a oprávnění

Pro ověření podpory autorizace pomocí rolí je potřeba v instanci Logto založit ukázkové uživatele a přiřadit jim role. Logto systém oprávnění pomocí rolí rozšiřuje o oprávnění, čímž se liší od ZITADEL. Tyto role se následně přiřadí k rolím. Pro vytvoření ukázkových oprávnění je potřeba nejprve vytvořit API zdroj.

Vytvoření API zdroje *API Resources → Create API Resource*

Podobně jako v ZITADEL existoval projekt, který měl role, je možné v Logto založit tzv. „API zdroj“. K tomuto zdroji lze vytvořit možná oprávnění. Oprávnění mají představovat, co uživatel v systému smí provést za akci nad jakým typem dat (například čtení/zápis objednávek) [91].

Pro testovací účely je v Logto založen API zdroj „ExampleResource“ identifikovaný adresou „http://localhost“. K tomuto zdroji jsou vytvořeny obecné oprávnění „read“, „write“, „delete“.

Vytvoření role *Roles → Create Role*

Po přípravě oprávnění je možné v systému vytvořit role. Stejně jako v ZITADEL jsou vytvořeny role „user“, „supervisor“ a „admin“. K těmto rolím jsou přiřazeny následující oprávnění.

user read

supervisor read write

admin read write delete

Vytvoření uživatele *User Management → Add User*

Dále jsou vytvoření uživatelé „aliceexample“, „bobexample“ a „charlieexample“. Při vytvoření uživatel je nutné zadat přihlašovací jméno a e-mail. Obě informace musí být unikátní v rámci instance Logto. Po založení uživatele se administrátorovi zobrazí nastavené parametry a navíc vygenerované heslo. Toto heslo má předat vytvořenému uživateli.

Pro srovnání v ZITADEL tento krok funguje sofistikovaněji, jelikož místo zobrazení hesla administrátorovi je uživateli odeslán e-mail s odkazem pro nastavení prvního hesla. Pokud administrátor první heslo nastaví, je uživatel pak vyzván k jeho změně při prvním přihlášení. Díky tomu je v ZITADEL předcházeno tomu, aby osoba zakládající uživatelský účet znala hesla uživatelů. Tento mechanismus Logto postrádá.

Po vytvoření uživatel je možné nastavit jeho celé jméno a přidat odkaz na fotku. Dále je možné uživateli nastavit metadata v podobě JSON. Pro ověření možností získání těchto dat ve službách byly všem uživatelům nastaveny metadata `{"group": "prague"}`. Na další záložce uživatelského profilu se konfigurují role, které byly uživatelům nastaveny následovně:

aliceexample user

bobexample user supervisor

charlieexample user supervisor admin

5.3.4 Registrace ukázkové aplikace

Založení aplikace *Applications → Create Application*

Po přípravě uživatelů přichází na řadu založení aplikace v přihlašovací službě a následná integrace do ukázkového systému. Obdobně jako ZITADEL dává i Logto na výběr mezi několika typy aplikací (nativní aplikace, SPA, webová aplikace nebo backend služba). Pro účely integrace do ukázkového systému je vybrána SPA aplikace. Ihned po vyplnění názvu aplikace a potvrzení se zobrazí návod o 5 krocích jak přihlášení integrovat do aplikace napsané v knihovně React (Vue/Vanilla). Kromě návodu je při integraci možné vycházet i z demo integrace ve zdrojových kódech Logto dostupných na GitHub `https://github.com/logto-io/logto/blob/master/packages/demo-app/`, která oproti návodu nepoužívá *react-router*.

Oproti ZITADEL, kde stačilo pro integraci využít podporovanou knihovnu *react-oidc*, je tentokrát potřeba použít knihovnu od Logto, která má připravené přihlašovací procesy.

Po instalaci knihovny *@logto/react* je v první řadě potřeba vyměnit *AuthProvider* za *LogtoProvider* a upravit konfiguraci viz ukázka kódu 5.3.4. V konfiguraci je třeba nastavit adresu služby Logto, id aplikace „ExampleApp“, zdroje, ke kterým bude ukázková aplikace přistupovat a oprávnění, která bude potřebovat. Uživatel se bude moci přihlásit i v případě, že oprávnění nemá, nicméně do tokenu budou přidány pouze oprávnění, která má. Tato oprávnění si následně z tokenu mohou přečíst jak klientská aplikace, tak backend služba. V Logto není možné získat z tokenu i uživatelovi role, nicméně tato funkcionalita je plánu na rok 2023 [92].

Pomocí nastavení *custom_data* v parametru *scope* je možné vyžádat si přístup k metadatům. Nicméně tyto metadata následně nejsou zahrnuty v žádném tokenu a tedy není možné v backend službě ověřit, že s nimi nikdo nemanipuloval.

Při vytváření požadavků na ukázkové služby je potřeba pomocí Logto knihovny získat token určený přímo pro daný zdroj. K tomu slouží funkce *getAccessToken*, které se jako parametr předá identifikátor zdroje *http://localhost* viz ukázka kódu 12.

Logto vkládá udělené oprávnění do tokenu pod klíč *scope*. Z toho důvodu je potřeba upravit backend služby tak, aby namísto rolí využívaly právě tuto hodnotu. Dále je potřeba upravit potřebná oprávnění u endpointů API služeb tak, aby místo rolí využívaly právě nastavená oprávnění. V této ukázce je

```
const config: LogtoConfig = {
  endpoint: "http://localhost:3001/",
  appId: "o22js2qc81b9kwcg7cp89",
  resources: ["http://localhost"],
  scopes: ["custom_data",
    /* permissions: */ "read", "write", "delete"],
};

function App() {
  const isInCallback = !!(
    new URL(window.location.href).searchParams.get("code")
  );
  return (
    <div className="App">
      <LogtoProvider config={config}>
        {isInCallback ? (
          <Callback />
        ) : (
          ...
        )}
      </LogtoProvider>
    </div>
  );
}
```

Listing 11: Integrace Logto do ukázkové aplikace

tato změna triviální, jelikož stačí zaměnit „supervisor“ za „write“ a „admin“ za „delete“.

Kompletní zdrojové kódy jsou dostupné v příloze.

5.3.5 Registrace Google jako IdP pro ZITADEL

Opět jako u ZITADEL se podpora federace identity ověří připojením k přihlašovací službě Google. V Google Cloud konzoli se jako adresa pro přesměrování po přihlášení vyplní *http://localhost:3001/callback/<id>*, kde *<id>* se vygeneruje při konfiguraci IdP (konektoru) v Logto.

Konfigurace IdP *Connectors* → *Social connectors* → *Add* → *OIDC*

Vyplnění konfiguračních parametrů:

Identity provider name Google

Authorization Endpoint https://accounts.google.com/o/oauth2/v2/auth

```
const { getAccessToken } = useLogto();
const supervisorEndpointData = useQuery({
  queryKey: ["endpoint", "restricted", port],
  queryFn: async () => {
    const token = await auth.getAccessToken("http://localhost");
    const response = await ky.get(
      `http://localhost:${port}/restricted`,
      {
        headers: {
          Authorization: token ? `Bearer ${token}` : undefined
        },
      }
    ).json();
    return publicResponseParser.parse(response);
  },
});
```

Listing 12: Integrace Logto do ukázkové aplikace – získání tokenu pro zdroj

Token Endpoint https://oauth2.googleapis.com/token

Client ID dle hodnoty vygenerovné v Google Cloud konzoli

Client secret dle hodnoty vygenerovné v Google Cloud konzoli

Scope openid profile

jwksUri https://www.googleapis.com/oauth2/v3/certs

Po dokončení nastavení je možné se přihlásit pomocí Google účtu přes protokol OIDC.

5.3.6 Ověření revokace refresh tokenu

Uživatele je v administraci Logto možné suspendovat, čímž se dle dokumentace i zneplatní jeho refresh token. Při dalším pokusu o získání nového access tokenu bude požadavek zamítnut i přesto, že refresh token stále neexpiroval. [93].

Tuto funkcionalitu lze ověřit buď počkáním na expiraci životnosti access tokenu nebo ruční změnou jeho expirační doby, díky čemuž knihovna logto požádá o obnovu tokenu. Očekává se, že po suspendování uživatele skončí požadavek o obnovu access tokenu chybou.

Po vyzkoušení tohoto scénáře se ukázalo, že navzdory dokumentaci je uživateli i po suspendování přístupový token k API zdroji opět vydán. Uživateli je znemožněn přístup až po jeho odhlášení a pokusu o opětovné přihlášení.

5.3.7 Závěr

Ačkoliv to není jedna z doporučených cest jak Logto spustit, tak jej bylo možné s podporou dokumentace spustit v systému Kubernetes.

Nastavení celé služby je ve srovnání se ZITADEL méně rozsáhlé, ale za to mnohem přehlednější. Bylo možné nastavit přizpůsobený vzhled, SMTP server pro odesílání e-mailů, vytvořit role, uživatele a ukázkovou aplikaci.

Ve srovnání se ZITADEL nemá Logto tak propracované vytváření uživatelů, jelikož ZITADEL po vytvoření uživatele odešle uživateli e-mail pro potvrzení registrace a e-mail pro vytvoření hesla. Oproti tomu Logto (v případě založení uživatele administrátorem) rovnou e-mail považuje za ověřený a heslo pouze zobrazí administrátorovi, který ho má předat uživateli. To nepředchází předání hesla nezabezpečenou cestou.

Administrace umožňuje úpravu vzhledu přihlašovací stránky. Konkrétně lze přímo v administraci změnit barva a logo. Nicméně oproti ZITADEL nelze mít více tenantů a nastavit vzhled pro každého zvlášť.

Integrace do ukázkové aplikace vyžaduje několik kroků a je dostatečně popsána jak v dokumentaci, tak v administraci Logto. Služby sice po provedení dotazu nemají dostupné role uživatelů, ale mají dostupné jejich oprávnění. V krajním případě mohou oprávnění plně nahradit role, jelikož je možné oprávnění pojmenovat stejně jako roli a každé roli nastavit stejnojmenné oprávnění. Tím lze docílit stavu, jakoby celý mikroservisní systém používal pouze role.

Po přihlášení uživatel je uživateli umožněno získat JWT token přímo pro zdroj, který chce dotazovat. Zároveň je uživateli vydán token pro obnovení přístupového tokenu. Přístupový token má nastavitelnou životnost [94] s výchozí hodnotou 3600 sekund. Po integraci bylo ověřeno, že se přístupový token po hodině automaticky obnoví a není vyžadováno opětovné přihlášení. Při ověřování vlastností byly nalezeny dva nedostatky. Oproti ZITADEL nejsou uživatelská metadata zahrnuta v tokenu, a tak si je mikroslužby nemohou ověřit. Dalším nedostatkem je nefunkčnost revokace refresh tokenu při suspendování uživatele jak bylo deklarováno v dokumentaci.

Kapitola 6

Testování bezvýpadkového škálování

Po integraci autentizační a autorizační služby do ukázkové mikroservisní aplikace je možné otestovat schopnost služby škálovat bez výpadků. Test by měl ověřit následující:

1. Aplikaci je možné spustit v několika replikacích.
2. Aplikace nepřestane fungovat pokud se ukončí některá replikace.
3. Množství výpadků požadavků na službu při změně počtu replikací.

Pro ověření těchto vlastností je nastaven počet replikací na hodnotu 3. Následně bude ověřena funkčnost ukázkové mikroservisní aplikace tím, že bude několikrát provedeno přihlášení pod různými uživateli s různými rolemi a zkontrolována bezchybnost provedení požadavků na mirkoslužby.

V případě funkčnosti aplikace bude spuštěno automatizované zasílání požadavků na službu zatímco se v průběhu zasílání požadavků bude měnit počet replikací. To ověří, že služba nepřestane fungovat při změně počtu replikací. V případě výpadků se naměří jejich počet.

6.1 k6 – nástroj pro zátěžové testy

Na základě konzultace s vedoucím práce je pro testování vybrán nástroj k6. Nástroj k6 slouží k realizaci zátěžových testů. Zaměřuje se na vývojáře a rozšiřitelnost. Umožňuje otestovat výkon a spolehlivost systému a pomoct tím vytvářet škálovatelné aplikace. [95] Testování pomocí tohoto nástroje se provádí pomocí skriptů napsaných v jazyce JavaScript a použití k6 knihoven. Skript zpravidla obsahuje funkci exportovanou pomocí `export default`, kterou k6 následně spustí. Tato funkce představuje jednu iteraci testu pro jednoho uživatele. Tedy pokud je simulováno více uživatelů, k6 tuto funkci spustí vícekrát

```
import http from 'k6/http';
import { sleep } from 'k6';

export default function () {
  http.get('https://test.k6.io');
  sleep(1);
```

Listing 13: k6 jednoduchý příklad testu

Tabulka 6.1: Parametry testovacího stroje

Operační systém	macOS 12.5.1
Počet jader	8
Maximální frekvence procesoru	3,2 GHz [97]
Architektura	ARM
Čip	Apple M1 Pro
Paměť	16 GB LPDDR5

paralelně. Jednoduchý příklad testu je vidět na ukázce kódu 13. Tento kód lze následně spustit příkazem `k6 run --vus 10 --duration 30s script.js`. Parametr *vus* je počet virtuálních uživatelů a parametr *duration* je doba, po kterou má test běžet. Tedy tento příkaz říká, že se má funkce v ukázce kódu 13 spustit paralelně desetkrát a ve chvíli, kdy se jedno spuštění dokončí, spustí se opakovaně pokud doba od zapnutí testu nepřekročila 30 sekund.

Ačkoliv cílem tohoto tetu není otestovat maximální možnou zátěž nebo výkon, nástroj k6 je vhodný, jelikož díky němu lze nepřetržitě vytvářet požadavky na aplikaci a udržet tím replikace ve stavu zpracovávání požadavku.

6.2 Zdroje dostupné ukázkové aplikaci a testům

Všechny aplikace běží na osobním notebooku MacBook Pro. Detailní parametry jsou dostupné v tabulce 6.1. Z toho důvodu je očekávatelné, že aplikace, které jsou určené pro běh na serverových strojích, mohou mít výkonnostní problémy. Nicméně v tomto testu není významný výkon aplikace, ale její schopnost škálovat (měnit počet běžících instancí) bez výpadků.

Na notebooku je spuštěna aplikace Docker Desktop, která slouží ke spouštění a správě kontejnerů a obrazů. Obsahuje zároveň nástroj pro správu a škálování kontejnerů Kubernetes. Kubernetes je potřeba zapnout v nastavení. [96] Aplikace Docker Desktop má nastavené limity na využití zdrojů. Tyto limity lze najít v tabulce 6.2.

Tabulka 6.2: Docker Desktop limity

Počet CPU	4
Paměť	8 GB
Swap	1 GB
Limit virtuálního úložiště	64 GB

```
#!/bin/bash

k6 run --vus 20 --duration 5m load-test.js &

sleep 10
kubectl scale --replicas 5 deployment my-zitadel
sleep 10
kubectl scale --replicas 3 deployment my-zitadel
sleep 10
kubectl scale --replicas 2 deployment my-zitadel

sleep 30
...
```

Listing 14: Ukázková část skriptu pro spuštění testu bezvýpadkového škálování

6.3 Průběh testu

V průběhu testu bude horizontálně škálována aplikace a zaznamenáno případné selhání dotazů. Manuální škálování služby se v Kubernetes provádí pomocí příkazu `kubectl scale deployment (TYPE NAME) --replicas=1`.

Test spuštěný pomocí nástroje k6 začne spouštět dotazy na různé endpointy autentizační a autorizační služby a ověřovat odpovědi. Nástroj k6 simuluje 20 uživatelů a požadavky bude vytvářet po dobu 5 minut.

Zatímco se budou provádět požadavky, automatizovaný test po 10 vteřinách změní počet replikací na hodnotu 5. Po dalších 10 vteřinách sníží počet replikací na 3 a po dalších 10 vteřinách na 2. Následně ponechá po dobu 30 vteřin počet replikací beze změny. Tento proces změny počtu replikací se provede dohromady třikrát. Skript ke spuštění testu je možné si prohlédnout na ukázce kódu 14.

6.4 ZITADEL

Služba ZITADEL byla při integraci s ukázkovou aplikací spuštěna v Kubernetes a je možné spustit více replikací, čímž je možné ji horizontálně škálovat. V následujícím testu je ověřeno, že při škálování nenastanou významné výpadky nebo autentizační a autorizační služba nespadne kompletně.

6.4.1 Testované endpointy

V ukázkové aplikaci bylo využito mechanismu přihlašování pomocí OIDC protokolu a následně získávání veřejných klíčů použitelných k ověření validity tokenů. Alternativní způsob ověření tokenu je využití endpointu v ZITADEL API. Test tedy provede následující reprezentativní požadavky: přihlášení na uživatele, získání veřejných klíčů, ověření tokenu pomocí API a získání profilu uživatele pomocí tokenu získaného po přihlášení. Tyto požadavky budou vytvářeny opakovaně a paralelně pro simulaci několika uživatelů.

přihlášení /oauth/v2/token

veřejné klíče /oauth/v2/keys

ověření tokenu /oauth/v2/introspect

získání profilu /oidc/v1/userinfo

V první iteraci se nejprve provede přihlášení a uloží token. V následujících iteracích se náhodně vybere a provede jeden ze zmíněných požadavků.

6.4.2 Výsledky testu

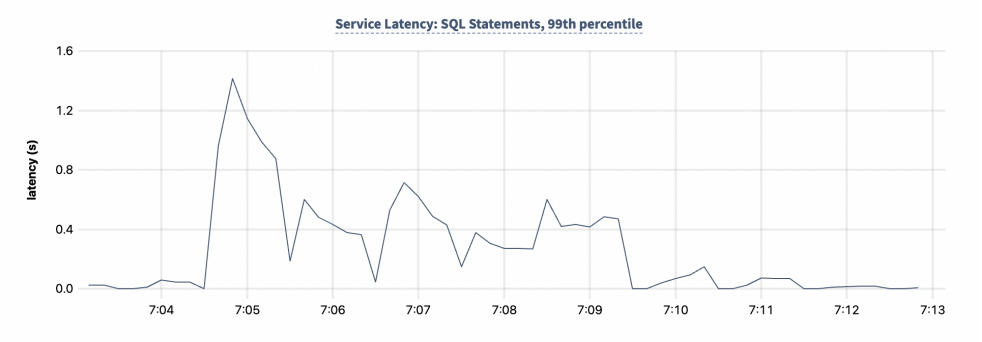
Výsledek testu je možné vidět na obrázku 6.1. Na začátku reportu jsou pouze úvodní informace o nastaveném počtu virtuálních uživatelů a době běhu. Dále jsou záznamy o provedení škálování. Následně je možné vidět, že všechny kontroly odpovědí na požadavky byly úspěšné. V případě selhání některých kontrol by nástroj k6 zobrazil počet selhání a relativní úspěšnost v procentech. Nakonec jsou detailněji popsány informace například o počtu provedených kontrol nebo doby požadavků.

Ačkoliv je primárně testováno, že nedojde k výpadkům požadavků, ve statistikách je vidět, že průměrná doba zpracování požadavku byla 1,7s. To je způsobeno velkou latencí databázových požadavků. Na grafu 6.2 je vidět, že v průběhu testu (čas 7:04:30 - 7:09:30) mělo po většinu času 99 % databázových dotazů dobu vykonání delší než 0,4s. Nejvyšší hodnota byla dokonce 1,6s. Je možné, že databáze nemá přiděleno dostatek zdrojů, nicméně pro tento test to není takový problém. Naopak je díky tomu větší šance, že se služba ukončí při škálování právě v okamžiku, kdy je zpracováván požadavek. Prodleva databáze způsobuje větší okno tohoto zpracování.

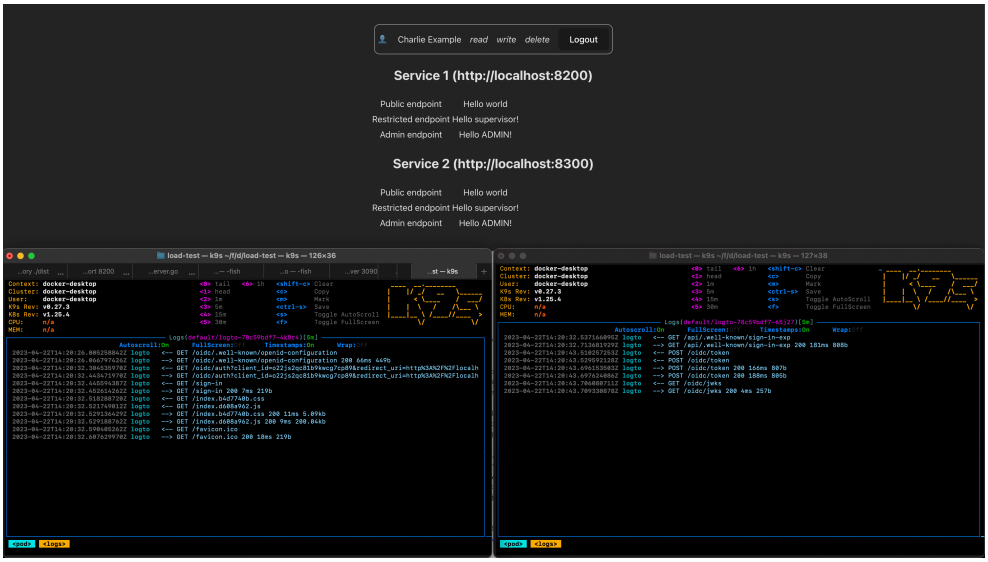
Test ověřil, že nedošlo k výpadku požadavků i při změně počtu replikací a tedy ZITADEL splňuje požadavek na schopnost bezvýpadkového horizontálního škálování.

```
[dominikdosoudil@Dominik-MacBook-Pro-2 ~/f/d/load-test [1]> ./run-test.sh

  execution: local
     script: load-test.js
     output: -

  scenarios: (100.00%) 1 scenario, 20 max VUs, 5m30s max duration (incl. graceful stop):
           * default: 20 looping VUs for 5m0s (gracefulStop: 30s)

deployment.apps/my-zitadel scaled
deployment.apps/my-zitadel scaled
deployment.apps/my-zitadel scaled
deployment.apps/my-zitadel scaled
deployment.apps/my-zitadel scaled
deployment.apps/my-zitadel scaled
deployment.apps/my-zitadel scaled
deployment.apps/my-zitadel scaled
deployment.apps/my-zitadel scaled

     ✓ [login] status was 200
     ✓ [login] token exists
     ✓ [loadKeys] status was 200
     ✓ [userInfo] status was 200
     ✓ [userInfo] username exists
     ✓ [checkToken] status was 200

     checks.........................: 100.00% ✓ 5354      ✗ 0
     data_received..................: 2.6 MB  8.5 kB/s
     data_sent......................: 683 kB  2.3 kB/s
     http_req_blocked...............: avg=8.33µs  min=0s     med=4µs      max=4.26ms  p(90)=6µs    p(95)=7µs
     http_req_connecting............: avg=1.85µs  min=0s     med=0s       max=452µs   p(90)=0s     p(95)=0s
     http_req_duration..............: avg=1.7s    min=6.2ms  med=979.42ms max=12.81s  p(90)=5.21s  p(95)=6.29s
       { expected_response:true }...: avg=1.7s    min=6.2ms  med=979.42ms max=12.81s  p(90)=5.21s  p(95)=6.29s
     http_req_failed................: 0.00%   ✓ 0         ✗ 3529
     http_req_receiving.............: avg=151.8µs min=12µs   med=96µs     max=13.06ms p(90)=169µs  p(95)=241.59µs
     http_req_sending...............: avg=32.25µs min=3µs    med=21µs     max=8.93ms  p(90)=35µs   p(95)=43µs
     http_req_tls_handshaking.......: avg=0s      min=0s     med=0s       max=0s      p(90)=0s     p(95)=0s
     http_req_waiting...............: avg=1.7s    min=6ms    med=976.53ms max=12.81s  p(90)=5.21s  p(95)=6.29s
     http_reqs......................: 3529    11.670784/s
     iteration_duration.............: avg=1.7s    min=6.46ms med=979.68ms max=13.35s  p(90)=5.21s  p(95)=6.29s
     iterations.....................: 3529    11.670784/s
     vus............................: 4       min=4       max=20
     vus_max........................: 20      min=20      max=20


running (5m02.4s), 00/20 VUs, 3529 complete and 0 interrupted iterations
default ✓ [======================================] 20 VUs  5m0s
```

Obrázek 6.1: Výsledky testu bezvýpadkového škálování ZITADEL



Obrázek 6.2: Prodleva databáze při testu bezvýpadkového škálování



Obrázek 6.3: Logto přihlášení pomocí několika instancí

6.5 Logto

Služba Logto byla spuštěna v Kubernetes a podle dokumentace je možné spustit několik instancí zároveň, ale je potřeba mezi všemi instancemi sdílet složku obsahující konektory. [87] Tato podmínka byla zohledněna při spouštění. V ukázce kódu 9, která zobrazuje specifikaci pro Kubernetes, je nadefinována sdílená složka s názvem „connectors“. Ověření funkčnosti lze provést zvýšením počtu replikací a vyzkoušením funkčnosti přihlašování v ukázkové aplikaci. Obrázek 6.3 ukazuje log 2 instancí po úspěšném přihlášení do ukázkového mikroservisního systému. Tím je ověřeno, že aplikace umí poskytnout službu přihlášení i pokud běží v několika replikacích.

Následně lze vyzkoušet, že přihlášení lze provést i po ukončení jedné z replikací a ověřit tím, že služba bude dále dostupná. I tento test proběhl v pořádku i po několika opakování.

6.5.1 Testované endpointy

Pro test je potřeba simulovat provoz. Pro tuto simulaci testovací nástroj provede přihlášení a stažení veřejných klíčů sloužících pro ověření tokenů. Jelikož samotný proces přihlášení je delší a probíhá přes několik přesměrování a v některých krocích je potřeba spustit JavaScript kód, nebude přihlášení kompletní. K testování ovšem stačí ověřit, že server neodpoví s chybou na požadavek s přihlašovacími údaji. Testovací nástroj vytváří dotazy na následující tři endpointy:

inicializace přihlášení /oidc/auth,

přihlášení uživatelským jménem a heslem /api/interaction,

potvrzení přihlášení /api/interaction/submit,

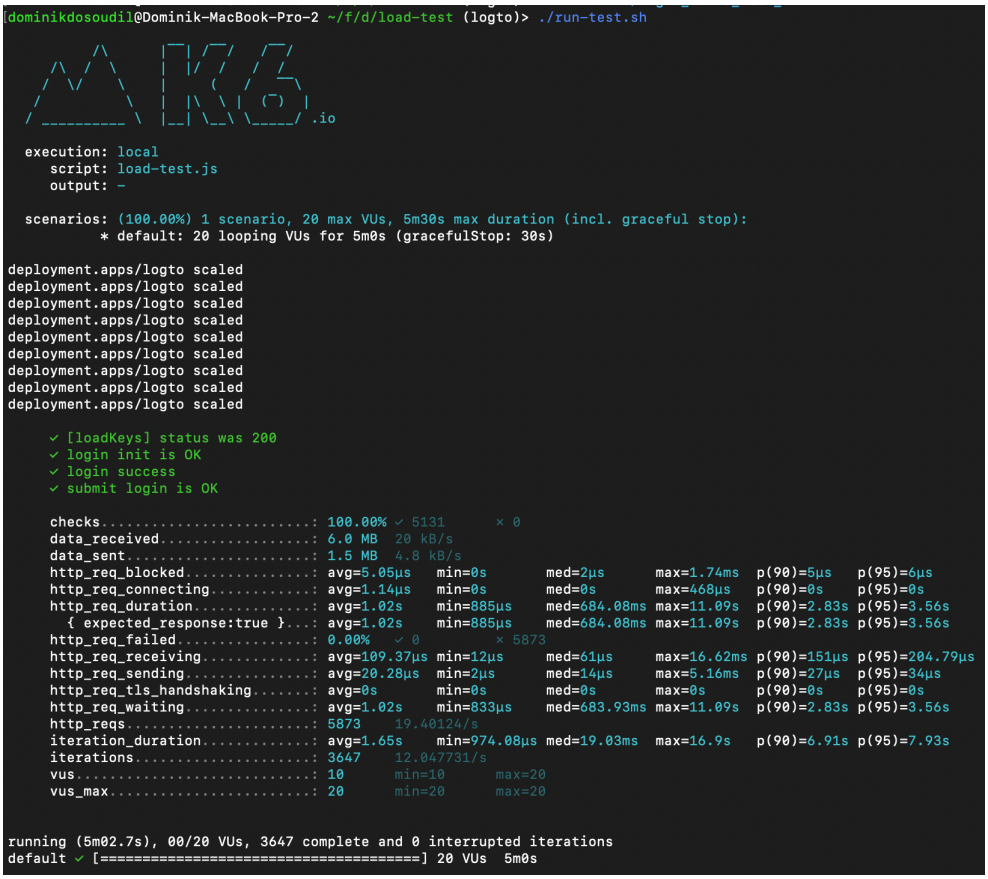
veřejné klíče /oidc/jwks.

V každé iteraci testu se náhodně vybere buď přihlášení uživatele zahrnující postupné zavolání prvních tří endpointů nebo získání klíčů simulující požadavek z backend služby.

6.5.2 Výsledky testu

Test byl opakován třikrát a ani v jednom z pokusů nedošlo k výpadkům jak je možné vidět podle hodnoty „checks“, která říká, že všech 5131 ověření odpovědi proběhlo v pořádku. Jeden z výsledků testů je možné vidět na obrázku 6.4. Ostatní jsou dostupné v přiloženém archivu.

Tímto testem byla ověřena schopnost provozu Logto v několika replikacích a možnost horizontálního škálování.



Obrázek 6.4: Výsledky testu bezvýpadkového škálování Logto

Kapitola 7

Vyhodnocení splnění požadavků

V rešerši byla provedena analýza vlastností nástrojů určených pro přihlašování uživatelů. Následně byly s firmami Stratox Cloud Native s.r.o. a INSOFT s.r.o diskutovány a specifikovány požadavky. Na základě požadavků a konzultace byl pro každou firmu vybrán jeden nástroj, který tyto požadavky splňuje dle rešerše.

Po výběru nástrojů byla vytvořena ukázková aplikace s mikroservisní architekturou pro demonstraci a ověření splnění vlastností.

V této kapitole je vyhodnoceno splnění požadavků obou firem. Na základě tohoto vyhodnocení je oběma firmám doporučen jeden z nástrojů.

7.1 ZITADEL

Tabulka 7.1 ukazuje, které požadavky služba splňuje.

Tabulka 7.1: ZITADEL – splnění požadavků

Požadavky Stratox Cloud Native s.r.o.	
Podpora mikroservisní architektury	ANO
Cloud-native kompatibilita	ANO
Bezvýpadkové škálování při nasazení v Kubernetes	ANO
Snadnost/možnost přizpůsobení UI login page	ANO
Požadavky INSOFT s.r.o.	
Spustitelnost na OS Ubuntu bez závislostí na prostředí	ANO
Bezestavová autentizace a autorizace	ANO
Revokace refresh tokenu	ANO
Podpora RBAC, ACL, skupiny	role, metadata
Podpora přihlášení pomocí jiných IdP (federace)	ANO
Podpora PostgreSQL	částečně

7.1.1 Podpora mikroservisní architektury

Tato vlastnost byla ověřena integrací do ukázkové mikroservisní aplikace, ve které bylo vyzkoušeno přihlášení i komunikace s mikroslužbami.

7.1.2 Cloud-native kompatibilita

Služba je připravená a schopná běžet jako kontejner v Docker a být spravována systémem Kubernetes. Tedy je kompatibilní s cloud prostředím.

7.1.3 Bezvýpadkové škálování při nasazení v Kubernetes

Škálování bez výpadku bylo ověřeno v sekci 6.4. Test ukázal, že je služba schopna škálovat bez výpadku.

7.1.4 Snadnost/možnost přizpůsobení UI login page

Integrace do ukázkové aplikace ukázala, že tato vlastnost je v ZITADEL flexibilně konfigurovatelná a dovoluje zvolit různé grafické přizpůsobení pro různé organizace. Konfigurace je popsána v podsekci 5.2.2. Takto nastavitelné přizpůsobení přihlašovací stránky plně vyhovuje požadavku firmy.

7.1.5 Spustitelnost na OS Ubuntu bez závislostí na prostředí

Aplikace je vytvořená v kompilovaném jazyku Go, díky čemuž ji lze spustit nativně na OS Linux.

7.1.6 Bezestavová autentizace a autorizace

ZITADEL aplikaci poskytne JWT, které je možné ověřit pomocí veřejných klíčů, čímž je tento bod splněn. Tento JWT token lze v ukázkové aplikaci po přihlášení nalézt v tzv. „Session Storage“, které lze zobrazit pomocí Chrome DevTools. [98] Získáním tokenu z „Session Storage“ a dekódováním jej pomocí online nástroje na stránce `https://jwt.io` byl ověřen formát tokenu.

7.1.7 Revokace refresh tokenu

Uživatele je možné v administraci deaktivovat, čímž proběhne revokace refresh tokenu. Tato funkcionalita byla ověřena experimentálně podle následujícího scénáře. Nejprve byla nastavena životnost access tokenu na 30 vteřin. Díky tomu není nutné dlouho čekat pro ověření. Následně bylo provedeno přihlášení do ukázkové aplikace. Po přihlášení byl v administraci deaktivován uživatel. Následně bylo pozorováno, zda se do nastavených 30 vteřin nepodaří obnovit token. Skutečně po vypršení tokenu došlo k odhlášení uživatele a zobrazil se formulář pro přihlášení.

7.1.8 Podpora RBAC, ACL, skupiny

ZITADEL umožňuje v každém projektu vytvořit role a ty následně přiřadit uživatelům. Tyto role jsou následně zahrnuty v JWT, kde je může přečíst a ověřit mikroslužba. Oproti Logto ZITADEL neumí oprávnění, která zabezpečení dělají flexibilnější. Skupiny ani ACL ZITADEL nepodporuje. Podporuje ale metadata, která lze využít pro uložení libovolných informací. Metadata lze oproti Logto vložit do ID tokenu. Informaci o skupině lze uložit do těchto metadat. Ověření vložení metadat bylo ověřeno experimentálně získáním tokenu a dekódováním jej pomocí nástroje na stránce `https://jwt.io`.

7.1.9 Podpora přihlášení pomocí jiných IdP (federace)

V podsekci 5.2.5 byla služba napojena na Google a ověřena možnost přihlášení se Google účtem. Po přihlášení se v ZITADEL vytvořil nový uživatel podle informací dostupných z Google účtu.

7.1.10 Podpora PostgreSQL

ZITADEL podporuje tento databázový systém, ale zatím pouze v BETA provozu. Tedy podpora v tuto chvíli není připravena na produkční provoz.

7.1.11 Shrnutí

Jelikož všechny vlastnosti byly ověřeny a splněny viz tabulka 7.1, ZITADEL je vhodný nástroj pro potřeby firmy Stratox Cloud Native s.r.o.

Zároveň služba vyhovuje téměř všem požadavkům firmu INSOFT s.r.o. Nekompletně vyhovující požadavek je podpora rolí, skupin a ACL. Z toho důvodu by firma INSOFT s.r.o. musela jinak navrhnout způsob ověřování oprávnění. Jelikož firma tento úkon v budoucnu plánuje pro zvýšení přehlednosti, je možné návrh směřovat k dosažení kompatibility se ZITADEL. Dalším bodem je částečná podpora PostgreSQL, ale protože kompletní přechod na jiný způsob autentizace a autorizace je potřeba dále diskutovat a testovat uvnitř firmy, může se firma podílet na testování a dokončení podpory tohoto databázového systému.

7.2 Logto

Tabulka 7.2 ukazuje, které požadavky služba splňuje.

7.2.1 Podpora mikroservisní architektury

Služba byla integrována do ukázkové mikroservisní aplikace a byla ověřena schopnost prokázání oprávnění přistupovat k mikroslužbám.

Tabulka 7.2: Logto – splnění požadavků

Požadavky INSOFT s.r.o.	
Podpora mikroservisní architektury	ANO
Spustitelnost na OS Ubuntu bez závislostí na prostředí	NE
Bezestavová autentizace a autorizace	ANO
Revokace refresh tokenu	NE
Podpora RBAC, ACL, skupiny	role
Podpora přihlášení pomocí jiných IdP (federace)	ANO
Podpora PostgreSQL	ANO
Požadavky Stratox Cloud Native s.r.o.	
Cloud-native kompatibilita	ANO
Bezvýpadkové škálování při nasazení v Kubernetes	ANO
Snadnost/možnost přizpůsobení UI login page	ANO

7.2.2 Cloud-native kompatibilita

Bylo ověřeno, že službu je možné spustit jako Docker kontejner spravovaný systémem Kubernetes. Službu bylo možné libovolně vypínat, zapínat a replikovat bez výskytu chyb ve funkčnosti.

7.2.3 Bezvýpadkové škálování při nasazení v Kubernetes

V sekci 6.5 bylo ověřeno, že službu je možné spustit v několika replikacích bez způsobení chyb. Navíc bylo automatizovaným testem ověřeno, že při škálování za provozu nedojde k výpadkům požadavků simulujících běžné používání služby.

7.2.4 Snadnost/možnost přizpůsobení UI login page

Při nastavení služby došlo k přizpůsobení vzhledu přihlašovací stránky. Oproti ZITADEL lze přizpůsobení provést pouze pro celou instanci Logto, nicméně pro firmy jako je INSOFT s.r.o., které nevyžadují podporu více tenantů, je toto nastavení dostatečné.

7.2.5 Spustitelnost na OS Ubuntu bez závislostí na prostředí

Aplikace Logto je vytvořená v jazyce TypeScript. Ten sice vyžaduje běhové prostředí a nelze jej tedy spustit nativně jak bylo požadováno, nicméně firma INSOFT s.r.o. tento jazyk používá a nebude překážkou pro spuštění nebo rozšíření aplikace. Bod je splněn částečně.

7.2.6 Bezestavová autentizace a autorizace

Logto vydává po přihlášení JWT tokeny, které lze za pomoci veřejného klíče ověřit bez nutnosti dotazování autorizační služby. Nicméně je třeba si od autentizační služby vyžádat právě klíče. Ty si lze však následně zapamatovat a tím pádem tato synchronizace bude probíhat jen zřídka. Taková cesta je přijatelná.

7.2.7 Revokace refresh tokenu

V podsekci 5.3.6 bylo vyzkoušeno suspendování uživatele a následné vyžádání nového přístupového tokenu. Při ověřování se ukázalo, že nový token je vydán i po suspendování uživatele. Ačkoliv tedy dokumentace popisuje, že po suspendování by se měl refresh token zneplatnit [93], tato funkcionalita nefunguje.

7.2.8 Podpora RBAC, ACL, skupiny

Logto používá pro autorizaci uživatelů role a oprávnění. Oprávnění se přidělují k rolím a ty pak k uživatelům. Oprávnění jsou následně dostupná v přístupovém JWT tokenu a koncové služby je mohou přečíst. Tím je splněna podpora RBAC. Logto ovšem nepodporuje skupiny ani ACL. Metadata uživatele nelze použít pro autorizaci, jelikož oproti ZITADEL nejsou zahrnuta v tokenu. To bylo ověřeno dekódováním tokenu v nástroji na stránce `https://jwt.io`. V ZITADEL metadata v ID tokenu zahrnuta byla. Tento bod je tedy splněn pouze částečně, nicméně role byly prioritní.

7.2.9 Podpora přihlášení pomocí jiných IdP (federace)

Kromě různých sociálních sítí Logto podporuje protokoly OAuth 2.0, OIDC a SAML. Zároveň umožňuje vytváření vlastní modulů pro jiné protokoly. [99]. Protokol OIDC byl v ukázkové aplikaci použit pro propojení s Google a bylo otestování přihlášení pomocí Google účtu. Tento požadavek je splněn.

7.2.10 Podpora PostgreSQL

Databázový systém PostgreSQL je jediná podporovaná databáze. Z toho důvodu je požadavek splněn.

7.2.11 Shrnutí

Služba Logto nesplnila všechny požadované vlastnosti pro firmu INSOFT s.r.o., ale problémy nejsou pro firmu neřešitelné. Nemožnost spustit aplikaci bez nutnosti instalace dalších nástrojů je pro firmu pouze nevýhoda a ne překážka. Firma INSOFT s.r.o. v jiných projektech pracuje s programovacím jazykem

TypeScript a je schopná tento projekt spustit, spravovat a případně se podílet na jeho vývoji formou open-source. Dalším nedostatkem je, že refresh token není zneplatněn po suspendování uživatele, ale jelikož to tak popisuje dokumentace, je tento nedostatek chybou, kterou firma může opravit a požádat o zahrnutí opravy do aplikace. Posledním nedostatkem je podpora pouze rolí a oprávnění, což nepokrývá veškeré potřeby firmy, ale tento nedostatek je řešitelný revizí a navržením systému ověřování oprávnění, který bude kompatibilní s Logto.

Služba Logto splnila i požadavky firmy Stratox Cloud Native s.r.o.

7.3 Doporučení nástrojů

Nástroj ZITADEL splnil všechny požadavky firmy Stratox Cloud Native s.r.o. a oproti Logto podporuje více tenantů v jedné instanci. Umožňuje každému z nich jinak přizpůsobit přihlašovací stránku. Tyto funkcionality jsou pro firmu Stratox Cloud Native s.r.o. vhodnější než Logto.

Ani jeden z nástrojů nesplnil kompletně požadavky firmy INSOFT s.r.o. Firma tedy bude muset provést úpravy svého systému v obou případech. V případě přechodu na autentizační a autorizační mikroslužbu v době, kdy již bude databázový systém PostgreSQL plně podporován, vyhovuje firmě více systém ZITADEL. To z důvodu, že v ZITADEL nebyla nalezena chyba s revokací tokenů a s možností použít metadata pro další parametry autorizace.

Stratox Cloud Native s.r.o. ZITADEL

INSOFT s.r.o. ZITADEL

Závěr

Cílem této práce bylo vybrat vhodný nástroj zajišťující správu identit a oprávnění pro firmy INSOFT s.r.o. a Stratox Cloud Native s.r.o. Obě firmy zadaly požadavky, které od nástroje potřebují pro splnění jejich potřeb a kompatibilitu s jejich systémy. Společnými požadavky pro obě firmy byla kompatibilita s mikroservisní architekturou a možnost spustit a spravovat nástroj svépomocí, čemuž se říká self-hosting.

Dílčím teoretickým cílem bylo provést rešerši nástrojů, které splňují tyto společné požadavky. Následně na základě zjištěných informací vytvořit interaktivní tabulky, které by umožnily nástroje porovnat.

Náplní praktické části bylo vybrat pomocí tabulek pro každou firmu jeden nástroj vyhovující požadavkům. Tyto dva nástroje měly být integrovány do ukázkové aplikace používající mikroservisní architekturu pro ověření nalezených vlastností.

Před začátkem samotné rešerše byly s oběma firmami zkonzultovány jejich produkty, do kterých má být hledaná služba integrována. Díky tomu bylo lépe porozuměno požadavkům. Na základě těchto informací byla provedena rešerše nástrojů. Výsledkem této rešerše je interaktivní tabulka dostupná na adrese `https://tinyurl.com/2p9x23ak`. Tato tabulka umožňuje konfigurovat váhy vlastností a v případě potřeby i jejich hodnocení. Díky tomu může tabulku využít kdokoliv a nástroje porovnat na základě svých požadavků.

V kapitole 4 jsou tabulky nakonfigurovány podle požadavků obou firem a na základě porovnání a konzultací jsou vybrány nástroje ZITADEL a Logto. Tyto nástroje jsou integrovány do ukázkové aplikace o dvou mikroslužbách a jedné webové klientské aplikaci. Tato integrace umožnila potvrdit funkčnost nalezených vlastností. Při integraci byl zjištěn nedostatek u nástroje Logto, a to nefungující revokace refresh tokenu při suspendování uživatele.

Při integraci do ukázkové aplikace byly nástroje spuštěny v systému Kubernetes, který umožňuje horizontální škálování aplikace. Pomocí nástroje `k6` byly provedeny testy schopnosti škálování aplikace bez výpadků. V tomto testu oba nástroje uspěly.

Po vyhodnocení ověření vlastností je oběma firmám doporučen nástroj ZITADEL, jelikož u něj nebyly nalezeny chyby při integraci do ukázkové aplikace a splňuje zadané požadavky.

Literatura

[1] Newcombe, L.: GENERAL OAUTH 2.0 FLOWS. [cit. 2023-03-06]. Dostupné z: https://cloudsundial.com/salesforce-oauth-flows

[2] Tracy in NetSuite: DEPRECATION OF GOOGLE OPENID SSO AND NETSUITE INBOUND SSO FEATURES. [cit. 2023-03-07]. Dostupné z: https://www.wgcgllc.com/netsuite_sso/

[3] GitHub, Inc.: Authorizer Contributors. [cit. 2023-03-26]. Dostupné z: https://github.com/authorizerdev/authorizer/graphs/contributors?from=2022-03-26&to=2023-03-26&type=c

[4] GitHub, Inc.: Keycloak Contributors. [cit. 2023-03-26]. Dostupné z: https://github.com/keycloak/keycloak/graphs/contributors?from=2022-03-26&to=2023-03-26&type=c

[5] Authelia: Architecture. [cit. 2023-02-24]. Dostupné z: https://www.authelia.com/overview/prologue/architecture/

[6] GitHub, Inc.: Authelia Contributors. [cit. 2023-03-26]. Dostupné z: https://github.com/authelia/authelia/graphs/contributors?from=2022-03-26&to=2023-03-26&type=c

[7] Ory: Overview. [cit. 2023-03-03]. Dostupné z: https://www.ory.sh/docs/ecosystem/projects

[8] GitHub, Inc.: Ory Kratos Contributors. [cit. 2023-03-26]. Dostupné z: https://github.com/ory/kratos/graphs/contributors?from=2022-03-26&to=2023-03-26&type=c

[9] GitHub, Inc.: ZITADEL Contributors. [cit. 2023-03-26]. Dostupné z: https://github.com/zitadel/zitadel/graphs/contributors?from=2022-03-26&to=2023-03-26&type=c

[10] GitHub, Inc.: Logto Contributors. [cit. 2023-03-26]. Dostupné z: https://github.com/logto-io/logto/graphs/contributors?from=2022-03-26&to=2023-03-26&type=c

[11] Graf, M.: How to Properly Implement Authorization in a Microservice Application. [cit. 2023-03-05]. Dostupné z: https://javascript.plainenglish.io/how-to-properly-implement-authorization-in-a-microservice-application-8fdb0d4dab1e

[12] HYPR Corp: Stateless Authentication (Token-Based). [cit. 2023-03-05]. Dostupné z: https://www.hypr.com/security-encyclopedia/stateless-authentication

[13] Šípek, R.: JSON Web Tokens (JWT). [cit. 2023-02-20]. Dostupné z: https://zooom.cz/json-web-tokens-jwt/

[14] TOTAL SERVICE a.s.: Cloud vs. on-premise: Jaká je budoucnost? [cit. 2023-03-28]. Dostupné z: https://www.totalservice.cz/novinky/cloud-vs-on-premise-jaka-je-budoucnost-2021-04-14

[15] IBM: What are Iaas, Paas and Saas? [cit. 2023-03-06]. Dostupné z: https://www.ibm.com/topics/iaas-paas-saas

[16] Microsoft: Co je cloud computing? [cit. 2023-03-06]. Dostupné z: https://azure.microsoft.com/cs-cz/resources/cloud-computing-dictionary/what-is-cloud-computing

[17] Amazon Web Services, Inc. or its affiliates: What Is Cloud Native? [cit. 2023-03-06]. Dostupné z: https://aws.amazon.com/what-is/cloud-native/

[18] Bečka, J.: Co je Single Sign-On? [cit. 2023-03-28]. Dostupné z: https://kpcs.cz/cs/novinky/blog/co-je-single-sign-on.html

[19] Entrust Corporation: WHAT IS AN IDENTITY PROVIDER (IDP)? [cit. 2023-03-06]. Dostupné z: https://www.entrust.com/resources/faq/what-is-an-identity-provider

[20] Parecki, A.: OAuth 2.0. [cit. 2023-03-06]. Dostupné z: https://oauth.net/2/

[21] Okta, Inc.: What is OAuth 2.0? [cit. 2023-03-06]. Dostupné z: https://auth0.com/intro-to-iam/what-is-oauth-2

[22] Mody, V.: How OIDC Authentication Works. [cit. 2023-03-06]. Dostupné z: https://goteleport.com/blog/how-oidc-authentication-works/

[23] Sharma, R.: What are Federated Identity Providers? [cit. 2023-03-21]. Dostupné z: https://www.loginradius.com/blog/identity/federated-identity-providers/

[24] Okta, Inc.: What is SAML 2.0? [cit. 2023-03-07]. Dostupné z: https://auth0.com/intro-to-iam/what-is-saml

[25] Guevara, H.: How SAML Authentication Works. [cit. 2023-03-07]. Dostupné z: https://auth0.com/blog/how-saml-authentication-works/

[26] Auth0® Inc.: Role-Based Access Control. [cit. 2023-03-07]. Dostupné z: https://auth0.com/docs/manage-users/access-control/rbac

[27] Skyflow, Inc.: RBAC vs ABAC: Access Control for Sensitive Data. [cit. 2023-03-07]. Dostupné z: https://www.skyflow.com/post/rbac-vs-abac-access-control-for-sensitive-data

[28] Imperva: Role-Based Access Control (RBAC). [cit. 2023-03-07]. Dostupné z: https://www.imperva.com/learn/data-security/role-based-access-control-rbac/

[29] authorizer.dev: Authorizer | Your data your control. [cit. 2023-02-21]. Dostupné z: https://authorizer.dev/

[30] FusionAuth: System Requirements. [cit. 2023-03-12]. Dostupné z: https://fusionauth.io/docs/v1/tech/installation-guide/system-requirements

[31] FusionAuth: Install a Database. [cit. 2023-03-11]. Dostupné z: https://fusionauth.io/docs/v1/tech/installation-guide/database

[32] FusionAuth: FusionAuth Installation. [cit. 2023-02-21]. Dostupné z: https://fusionauth.io/docs/v1/tech/installation-guide/#overview

[33] Keycloak community: Open Source Identity and Access Management. [cit. 2023-02-23]. Dostupné z: https://www.keycloak.org/

[34] Keycloak community: Configuring the database. [cit. 2023-03-11]. Dostupné z: https://www.keycloak.org/server/db

[35] Keycloak community: Downloads. [cit. 2023-02-23]. Dostupné z: https://www.keycloak.org/downloads

[36] Keycloack community: Themes. [cit. 2023-02-23]. Dostupné z: https://www.keycloak.org/docs/latest/server_development/#_themes

[37] Pomerium: What does Pomerium do? [cit. 2023-02-24]. Dostupné z: https://www.pomerium.com/docs

[38] Authelia: Access Control. [cit. 2023-02-25]. Dostupné z: https://www.authelia.com/configuration/security/access-control/

[39] Authelia: OpenID Connect. [cit. 2023-xx-xx]. Dostupné z: https://www.authelia.com/configuration/identity-providers/open-id-connect/

[40] Authelia: Statelessness. [cit. 2023-03-16]. Dostupné z: https://www.authelia.com/overview/authorization/statelessness/

[41] Authelia: Deployment. [cit. 2023-02-24]. Dostupné z: https://www.authelia.com/integration/deployment/

[42] Authelia: Storage. [cit. 2023-03-11]. Dostupné z: https://www.authelia.com/configuration/storage/

[43] Ory: Identity Infrastructure for the Internet. [cit. 2023-02-26]. Dostupné z: https://www.ory.sh/

[44] Ory Corp: Social sign-in. [cit. 2023-03-12]. Dostupné z: https://www.ory.sh/docs/self-hosted/kratos/configuration/oidc

[45] Ory: Manage identities and users in the cloud. [cit. 2023-02-26]. Dostupné z: ory.sh/kratos/

[46] Ory Corp: Introduction to Ory Identities (Ory Kratos). [cit. 2023-02-26]. Dostupné z: https://www.ory.sh/docs/kratos/ory-kratos-intro

[47] Ory Corp: Architecture principles. [cit. 2023-02-26]. Dostupné z: https://www.ory.sh/docs/ecosystem/software-architecture-philosophy

[48] Ory: Add Custom Login, Registration, User Settings to Your Next.js & React Single Page Application (SPA). [cit. 2023-03-03]. Dostupné z: https://www.ory.sh/nextjs-authentication-spa-custom-flows-open-source/

[49] Ory: Quickstart. [cit. 2023-03-03]. Dostupné z: https://www.ory.sh/docs/kratos/quickstart

[50] ZITADEL team: Let's add users to your apps. [cit. 2023-03-03]. Dostupné z: https://zitadel.com/

[51] ZITADEL team: Open Source. [cit. 2023-03-03]. Dostupné z: https://zitadel.com/opensource

[52] ZITADEL team: Overview. [cit. 2023-03-03]. Dostupné z: https://zitadel.com/docs/self-hosting/deploy/overview

[53] ZITADEL team: JWT IDP. [cit. 2023-03-04]. Dostupné z: https://zitadel.com/docs/concepts/structure/jwt_idp

[54] ZITADEL team: Updating and Scaling. [cit. 2023-03-16]. Dostupné z: https://zitadel.com/docs/self-hosting/manage/updating_scaling#scaling-zitadel

[55] ZITADEL team: Multitenancy support for B2B. [cit. 2023-03-03]. Dostupné z: https://zitadel.com/usecases#b2b

[56] ZITADEL team: Add Project Role. [cit. 2023-03-22]. Dostupné z: https://zitadel.com/docs/apis/mgmt/management-service-add-project-role

[57] ZITADEL team: Revoke Refresh Tokens. [cit. 2023-03-04]. Dostupné z: https://zitadel.com/docs/apis/auth/auth-service-revoke-my-refresh-token

[58] The Janssen Project: Welcome to the Janssen Project. [cit. 2023-03-04]. Dostupné z: https://github.com/JanssenProject/jans

[59] The Janssen Project: Overview. [cit. 2023-03-05]. Dostupné z: https://docs.jans.io/v1.0.8/admin/install/helm-install/

[60] The Janssen Project: Customize Web pages. [cit. 2023-03-05]. Dostupné z: https://docs.jans.io/v1.0.8/admin/developer/customization/customize-web-pages/

[61] The Janssen Project: Janssen Project Documentation. [cit. 2023-03-04]. Dostupné z: https://docs.jans.io/v1.0.8/

[62] The Janssen Project: Overview. [cit. 2023-03-05]. Dostupné z: https://docs.jans.io/v1.0.8/admin/reference/database/

[63] GitHub, Inc.: [cit. 2023-03-26]. Dostupné z: https://github.com/JanssenProject/jans/graphs/contributors?from=2022-03-26&to=2023-03-26&type=c

[64] SuperTokens: Introduction. [cit. 2023-03-04]. Dostupné z: https://supertokens.com/docs/userroles/introduction

[65] SuperTokens: How would you like to run SuperTokens? [cit. 2023-03-04]. Dostupné z: https://supertokens.com/pricing

[66] SuperTokens: Open Source User Authentication. [cit. 2023-03-04]. Dostupné z: https://supertokens.com/

[67] SuperTokens: Session management with SuperTokens. [cit. 2023-03-04]. Dostupné z: https://supertokens.com/docs/session/introduction

[68] GitHub, Inc.: SuperTokens Contributors. [cit. 2023-03-26]. Dostupné z: https://github.com/supertokens/supertokens-core/graphs/contributors?from=2022-03-26&to=2023-03-26&type=c

[69] Silverhand Inc.: Effortless identity solution with all the features you need. [cit. 2023-03-05]. Dostupné z: https://logto.io/

[70] Silverhand Inc.: Manage permissions and roles. [cit. 2023-03-05]. Dostupné z: https://docs.logto.io/docs/recipes/rbac/manage-permissions-and-roles

[71] Sun, G.: Logto 2023 February Update (Extended). [cit. 2023-03-05]. Dostupné z: https://docs.logto.io/blog/releases/2023-feb-extended/#-open-standard-connectors-with-better-config-interface

[72] authentik Security Inc.: docker-compose installation. [cit. 2023-03-05]. Dostupné z: https://goauthentik.io/docs/installation/docker-compose

[73] authentik Security Inc.: Overview. [cit. 2023-03-05]. Dostupné z: https://goauthentik.io/docs/flow/

[74] authentik Security Inc.: Tenants. [cit. 2023-03-05]. Dostupné z: https://goauthentik.io/docs/tenants

[75] authentik Security Inc.: Making authentication simple. [cit. 2023-03-05]. Dostupné z: https://goauthentik.io/

[76] Stratox Enterprises: Cloud-Native at the heart of our products & services. [cit. 2023-03-29]. Dostupné z: https://stratox.cz/cloud-native/

[77] Stack Exchange, Inc.: 2022 Developer Survey. [cit. 2023-04-27]. Dostupné z: https://survey.stackoverflow.co/2022/#technology-most-loved-dreaded-and-wanted

[78] Google: Go for Web Development. [cit. 2023-04-27]. Dostupné z: https://go.dev/solutions/webdev

[79] Krotoff, T.: Stack Overflow trends. [cit. 2023-04-08]. Dostupné z: https://gist.github.com/tkrotoff/b1caa4c3a185629299ec234d2314e190

[80] ZITADEL team: React. [cit. 2023-04-11]. Dostupné z: https://zitadel.com/docs/examples/login/react

[81] ZITADEL team: Kubernetes. [cit. 2023-03-17]. Dostupné z: https://zitadel.com/docs/self-hosting/deploy/kubernetes

[82] ZITADEL team: How to configure ZITADEL for your scenario. [cit. 2023-04-14]. Dostupné z: https://zitadel.com/docs/guides/solution-scenarios/configurations#trigger-organization-in-zitadel-login

[83] ZITADEL team: Recommended authorization flows. [cit. 2023-04-14]. Dostupné z: https://zitadel.com/docs/guides/integrate/oauth-recommended-flows

[84] ZITADEL team: Configure Google as Identity Provider. [cit. 2023-04-23]. Dostupné z: https://zitadel.com/docs/guides/integrate/identity-providers/google-oidc

[85] ZITADEL Helm charts contributors: ZITADEL. [cit. 2023-04-16]. Dostupné z: https://github.com/zitadel/zitadel-charts#whats-in-the-chart

[86] Logto Contributors: Get started. [cit. 2023-04-20]. Dostupné z: https://docs.logto.io/docs/tutorials/get-started/

[87] Logto Contributors: Deployment. [cit. 2023-04-20]. Dostupné z: https://docs.logto.io/docs/recipes/deployment/#shared-connectors-folder

[88] Bitnami: postgresql. [cit. 2023-04-20]. Dostupné z: https://artifacthub.io/packages/helm/bitnami/postgresql

[89] The Linux Foundation: Service. [cit. 2023-04-21]. Dostupné z: https://kubernetes.io/docs/concepts/services-networking/service/

[90] Silverhand Inc.: Configure Email Connector. [cit. 2023-04-21]. Dostupné z: https://docs.logto.io/docs/recipes/configure-connectors/configure-email-connector/

[91] Sun, G.: CIAM 102: Authorization & Role-based Access Control. [cit. 2023-04-22]. Dostupné z: https://docs.logto.io/blog/ciam-102-authz-and-rbac/

[92] Sun, G.: 2023 Public Roadmap. [cit. 2023-04-22]. Dostupné z: https://github.com/logto-io/logto/issues/1937

[93] Silverhand Inc.: Manage users using Admin Console. [cit. 2023-04-23]. Dostupné z: https://docs.logto.io/docs/recipes/manage-users/admin-console

[94] Silverhand Inc.: API Resource. [cit. 2023-04-22]. Dostupné z: https://docs.logto.io/docs/references/resources/

[95] Grafana Labs: Welcome to the k6 documentation. [cit. 2023-04-18]. Dostupné z: https://k6.io/docs/

[96] Docker Inc.: Docker Desktop. [cit. 2023-04-18]. Dostupné z: https://docs.docker.com/desktop/

[97] LaptopMedia Ltd.: Apple M1 Pro (8-core). [cit. 2023-04-17]. Dostupné z: https://laptopmedia.com/processor/apple-m1-pro-8-core/

[98] Basques, K.: View and edit session storage. [cit. 2023-04-29]. Dostupné z: https://developer.chrome.com/docs/devtools/storage/sessionstorage/

[99] Silverhand Inc.: Guide to implementing connectors. [cit. 2023-04-23]. Dostupné z: https://docs.logto.io/docs/recipes/create-your-connector/connector-implementation-guide

[100] Gazarov, P.: What is an API? In English, please. [online]. [cit. 2021-04-17]. Dostupné z: https://www.freecodecamp.org/news/what-is-an-api-in-english-please-b880a3214a82/

[101] Mozilla and individual contributors: MDN Web Docs - HTTP [online]. [cit. 2021-04-17]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP

[102] Mozilla and individual contributors: MDN Web Docs - URL [online]. [cit. 2021-04-17]. Dostupné z: https://developer.mozilla.org/en-US/docs/Glossary/URL

[103] ALTAXO SE: Co jsou zkratky B2C, B2B, B2G, B2E. [cit. 2023-03-07]. Dostupné z: https://www.altaxo.cz/provoz-firmy/marketing/co-jsou-zkratky-b2c-b2b-b2g-b2e

[104] Sakimura, N.; Bradley, J.; Agarwal, N.: Proof key for code exchange by OAuth public clients. [cit. 2023-04-14]. Dostupné z: https://www.rfc-editor.org/rfc/rfc7636

[105] Bos, B.: WHAT IS CSS? [cit. 2023-04-21]. Dostupné z: https://www.w3.org/Style/CSS/Overview.en.html

[106] Klensin, J.: Simple Mail Transfer Protocol. [cit. 2023-04-28]. Dostupné z: https://www.rfc-editor.org/rfc/rfc5321

[107] Štráfelda, J.: Autentizace. [cit. 2023-02-19]. Dostupné z: https://www.strafelda.cz/autentizace

[108] Larapedia: Meaning of lightweight. [cit. 2023-03-13]. Dostupné z: https://www.larapedia.com/glossary_of_computer_application_terms/lightweight_meaning_in_computer_application_terminology.html

[109] Rouse, M.: Runtime Environment. [cit. 2023-03-28]. Dostupné z: https://www.techopedia.com/definition/5466/runtime-environment-rte

[110] StackShare, Inc.: Popular Tech Stacks. [cit. 2023-03-31]. Dostupné z: https://stackshare.io/stacks

[111] SmartBear Software: API Endpoints - What Are They? Why Do They Matter? [cit. 2023-04-29]. Dostupné z: https://smartbear.com/learn/performance-monitoring/api-endpoints/

Příloha A

Seznam použitých zkratek

API Application Programming Interface [100]

HTML Hyper Text Markup Language

HTTP Hypertext Transfer Protocol [101]

URL Uniform Resource Locator [102]

SPA Single Page Application

IaaS Infrastructure as a Service

PaaS Platform as a Service

SaaS Service as a Service

CE Community Edition

RDBMS Relational Database Management System

JWT JSON Web Token

IdP Identity Provider [19]

RBAC Role-Based Access Control

B2B Business to Business [103]

B2E Business to Employee [103]

FE Frontend

BE Backend

UCS Unified Communication System

PKCE Proof Key for Code Exchange [104]

CSS Cascading Style Sheets [105]

SMTP Simple Mail Transfer Protocol [106]

SSO Single Sign-On [18]

Příloha B

Slovník

Autentizace *„proces ověření identity uživatele“* [107]

Self-hosting Self-hosting znamená provozování a spravování softwarového produktu nebo služby pomocí vlastního serveru namísto použití služeb třetích stran.

on-premise *„Software či hardware, který je uložen v interní infrastruktuře a prostoru společnosti.“* [14]

Access token Přístupový token v protokolu OAuth 2.0, slouží k ověření autorizace k provádění určitých akcí

Refresh token Token sloužící k získání nového access tokenu v případě jeho expirace

Lightweight Lightweight software nepotřebuje mnoho zdrojů počítače (jako například paměť). [108]

Runtime prostředí *„Runtime prostředí je prostředí, ve kterém se spouští program nebo aplikace. Je to hardwarová a softwarová infrastruktura, která podporuje běh konkrétního kódu v reálném čase.“* ([109] překlad autora)

Tech stack „Tech stack“ je definovaný jako sada technologií, které organizace používá pro sestavení webové nebo mobilní aplikace. Je to kombinace programovacích jazyků, knihoven, programovacích vzorů, serverů, UI/UX řešení a dalších nástrojů, které používají vývojáři. [110]

Endpoint API endpoint je lokace, na které jsou jiným službám dostupná data. Služby pomocí těchto bodů komunikují. [111]

Helm Správce balíčků pro Kubernetes

PKCE PKCE zabraňuje útoku na OAuth 2.0 autorizační kód, ve kterém škodlivý software odchytí autorizační kód a pomocí jej získá přístupový token. [104]

CSS Jednoduchý mechanismus pro přidávání stylů (např. fonty, barvy, rozmístění) webovým dokumentům. [105]

SMTP Protokol pro Internetovou poštovní elektronickou komunikaci. [106]

Příloha C

Obsah přiloženého archivu

- `thesis.pdf` ..... text práce ve formátu PDF
- `thesis_src` ..... zdrojové soubory práce
- `example-system-zitadel` ... zdrojové kódy ukázkové aplikace s integrací ZITADEL
  - `example-fe` ..... Frontend
  - `service1` ..... Služba 1
  - `service2` ..... Služba 2
- `example-system-logto` zdrojové kódy ukázkové aplikace s integrací Logto
  - `example-fe` ..... Frontend
  - `service1` ..... Služba 1
  - `service2` ..... Služba 2
- `github_star_crawler` . nástroj na zjištění statistik o GitHub repozitářích
- `kube-zitadel` ..... konfigurační soubory Kubernetes pro ZITADEL
- `kube-logto` ..... konfigurační soubory Kubernetes pro Logto
- `scale-test-zitadel` ..... automatické testy pro ZITADEL
  - `results` ..... výsledky testů
- `scale-test-logto` ..... automatické testy pro Logto
  - `results` ..... výsledky testů