POPIS ARCHITEKTURY SIS

Obsah

1 Úvod

Tento dokument shrnuje záměry a architektonický koncept nového systému. Po obecném popisu konceptu architektury (kap. 2 dokumentu) následuje stručná charakteristika obecných částí architektury a infrastruktury, kterou zajistí zadavatel (kap. 3 dokumentu). Kap. 4 dokumentu pak shrnuje požadavky kladené na architekturu a technologie použité při vývoji jednotlivých modulů, kterými se budou řídit dodavatelé těchto modulů. Kap. 5 dokumentu shrnuje obecná pravidla, která by měla být dodržena při postupném převodu stávajícího systému do nové architektury.

Záměr architektury a její porovnání s dosavadním stavem studijního informačního systému UK (dále jen „SIS“):

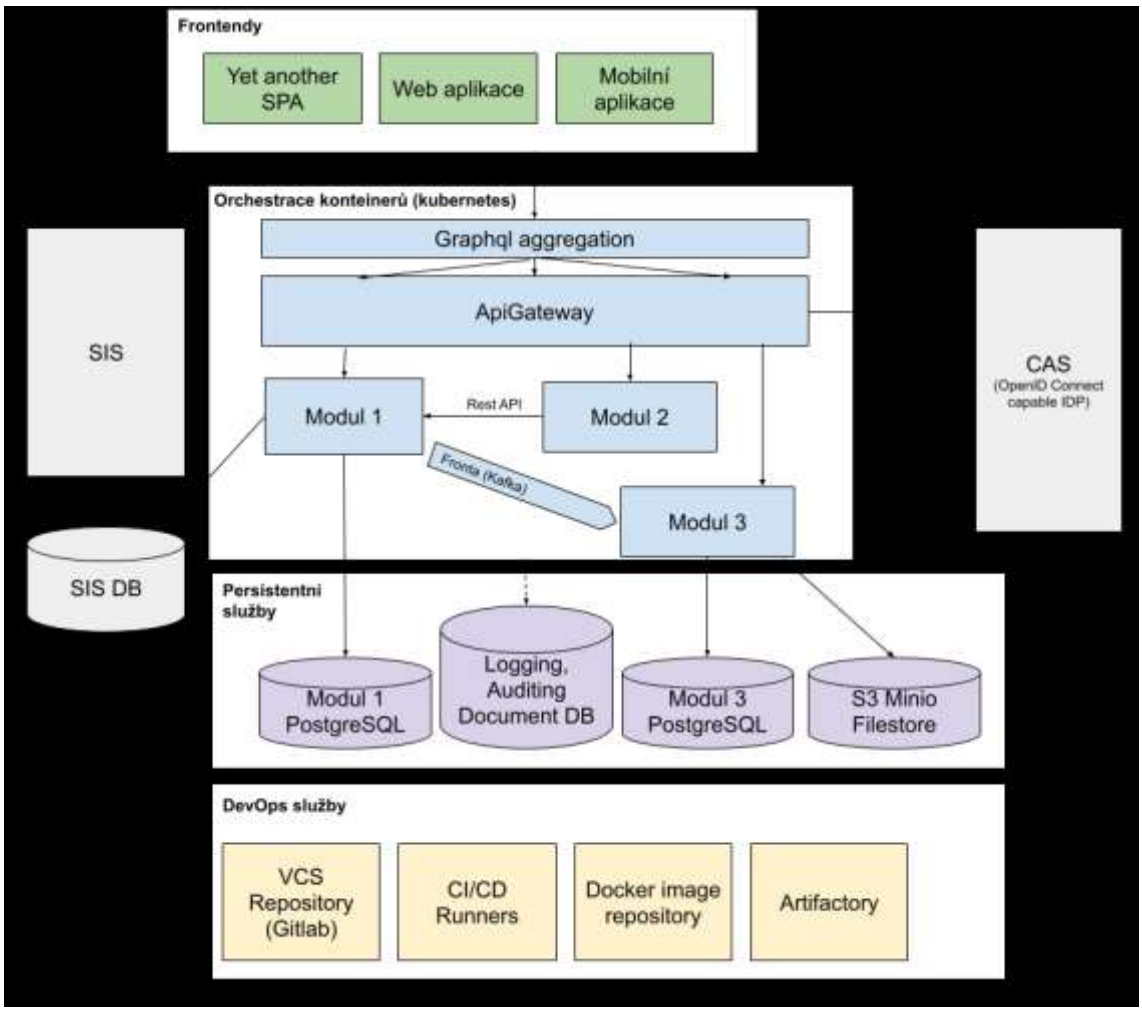
- V současnosti UK provozuje systém SIS, což je rozsáhlý monolitický systém napsaný v PHP, který se vyvíjel 15+ let. Nemá standardizované integrační rozhraní a většina integrací probíhá na úrovni Oracle database. Současné řešení je těžké rozvíjet a integrovat na něj nové aplikace.
- Záměr nové architektury
    - (A) Umožnit postupně zrefaktorovat monolit do (mikro)služeb, které jsou propojeny definovaným a zdokumentovaným rozhraním
    - (B) Umožnit UK rozvíjet systém po menších částech a v případě problémů / nových požadavků nahradit nebo upravit pouze malou část služeb bez ovlivnění ostatních služeb
    - (C) Definovat standardy pro vývoj a operaci služeb na nové architektuře. Služby je potom možné zadat různým dodavatelům a tím se zbavit vendor-lock-in
    - (D) Zrychlit čas dodávky nových funkcionalit a umožnit škálovatelnost vývojářských kapacit
    - (E) Umožnit lepší škálovatelnost výkonu při zátěži
    - (F) Preference open source řešení

2 Koncept nové architektury

Nová architektura se vyznačuje svojí modulárností. Infrastruktura poskytuje sdílené služby aplikačním modulům, jako je autentizace, persistence, logování, monitorování, audit, atd.

Tato architektura umožňuje:

- Paralelní vývoj několika dodavateli
- Postupný přepis SISu
- Škálování výkonu při zátěži
- Continuous deployment
- Rozsáhlé možnosti QA na různých úrovních systému
- V případě zastarání jednoho modulu není třeba přepisovat celý systém



3 Požadavky na jednotlivé aspekty architektury a zvolené technologie

3.1 Orchestrace containerů

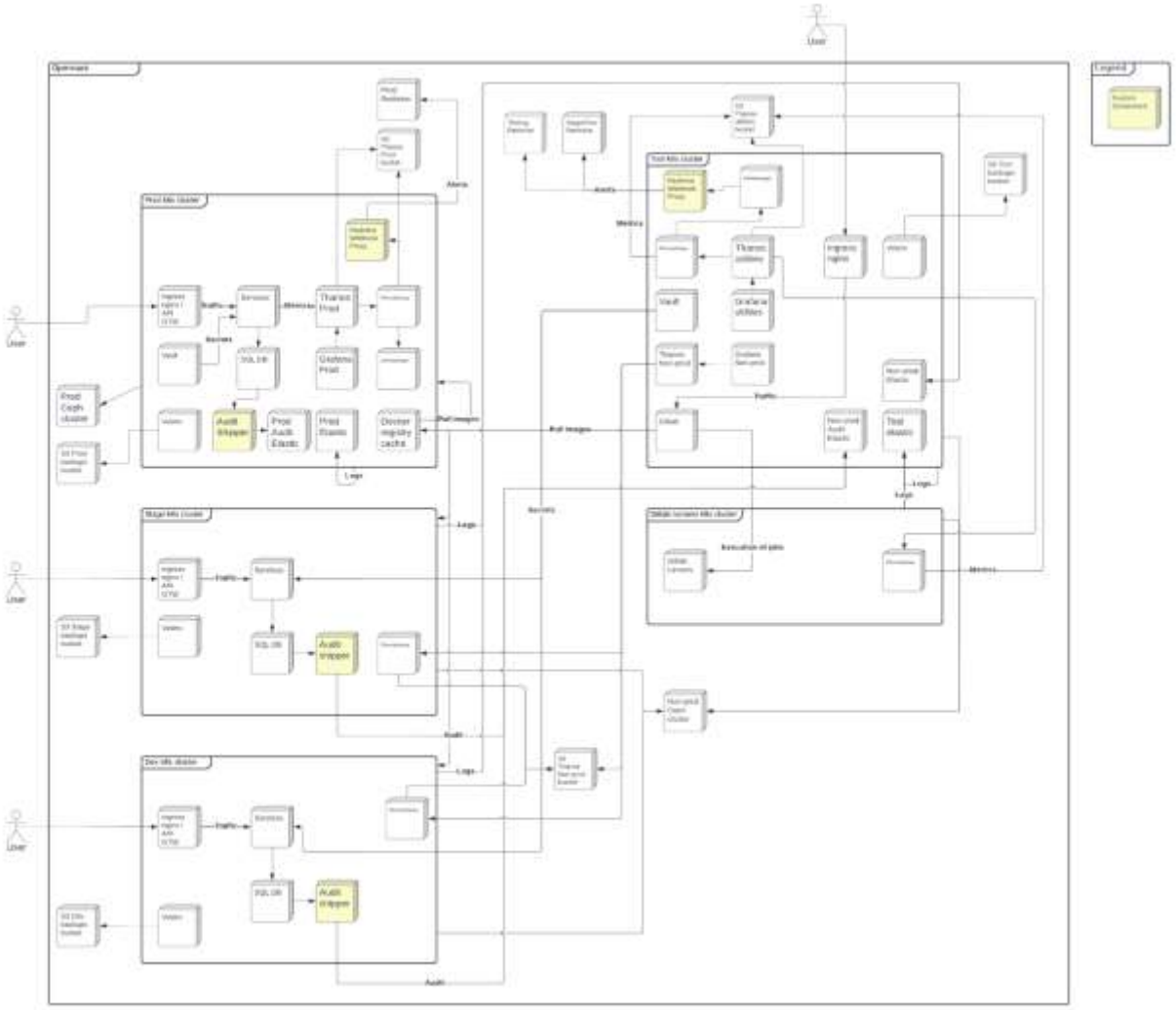
- Požadavky (a cíle)
  - (A) udržitelná technologie
    - podpora technologie za 10 let
    - ochrana před vendor lock-in

    - (B) service discovery
        - efektivní konfigurace služeb
    - (C) deklarativní popis aplikací
        - efektivní správa aplikací
    - (D) kompatibilita s monitoring řešením
        - korelace statistických indikátorů s ostatními moduly
    - (E) high availability
        - nasazení modulů v režimu high availability
        - automatické škálování dle zátěže
- Technologie
    - Kubernetes
    - Helm
    - Rancher
    - RKE
    - Ansible
- Infrastruktura
    - Zvolenou technologií pro orchestraci je Kubernetes.
    - Správa Kubernetes by měla probíhat pomocí Rancheru a RKE
        - Jelikož Rancher ideálně běží v Kubernetes, je doporučeno jeho cluster nasadit pomocí RKE2 a Ansible
    - Control plane Kubernetes musí běžet v high-availability s minimálně 3 nody, ideálně v separátních lokalitách
    - Kubernetes by měly být napojeny na CAS pro autentifikaci/autorizaci uživatelů pomocí Rancher authentication proxy

3.2 Prostředí

- Požadavky (a cíle)
    - (A) vývojářské prostředí
        - možnost testovat integraci s ostatními moduly při vývoji
        - prostředí přístupné vývojářům, vývojáři zde mohou nasazovat bez asistence své moduly
    - (B) staging prostředí
        - možnost testovat systém před nasazením do produkčního prostředí
        - možnost spouštět zátěžové testy
    - (C) produkční prostředí
        - instance systému dostupná uživatelům
        - možnost spouštět zátěžové testy
    - (D) všechna prostředí co nejpodobnější produkčnímu (infrastruktura, data, konfigurace, …) s ohledem na prostředky a legislativu (GDPR)
        - zvýšení pravděpodobnosti odhalení chyby před nasazením do produkčního prostředí
        - snížení pravděpodobnosti výskytu chyby specifické pro vývojové prostředí

        - striktní požadavek na anonymizaci dat používaných z produkčního na testovací, resp. vývojové prostředí.
- Infrastruktura
    - Budou existovat následující prostředí:
        - Production
        - Staging
        - Development
    - Každé prostředí má vlastní Kubernetes cluster
    - Mezi neprodukční a neaplikační prostředí se řadí ještě následující clustery
        - Tool cluster
            - Obsahuje GitLab a sdílené komponenty neprodukčních prostředí
        - GitLab Runner cluster
        - Rancher management cluster
            - Nasazen odděleně pomocí Kubespray
    - Produkční prostředí by mělo být kompletně odděleno od neprodukčních.
        - Vlastní monitoring i logging stack
        - Cache pro Docker Registry
        - Oddělený Ceph cluster
    - Neprodukční prostředí mají sdílené komponenty.
        - Sdílený Ceph cluster
        - Sdílený monitoring a logging
        - Sdílené komponenty se nachází na Tool clusteru
- Infrastrukturní diagram
    - Vlastní komponenty jsou zvýrazněny žlutě (v diagramu i v textu)



- Legenda
  - Kubernetes – nástroj pro orchestraci kontejnerů nad clusterem serverů
  - NGINX Ingress – obecná reverzní proxy
  - Kong API Gateway – specializovaná rozšiřitelná reverzní proxy s pokročilými funkcemi
  - PostgreSQL – relační databáze
  - Kafka – asynchronní fronta pro komunikaci mezi moduly
  - Ceph – platforma spravující datová úložiště
  - S3 – běžné rozhraní pro úložiště souborů/objektů
  - GitLab – nástroj pro správu zdrojových kódů a automatizaci
  - GitLab Runner – komponenta GitLabu, která zprostředkovává automatizaci
  - Docker Registry – úložiště obrazů kontejnerů
  - ArgoCD – nástroj pro synchronizaci nasazování aplikací a infrastruktury
  - Velero – zálohovací nástroj pro Kubernetes
  - Vault – nástroj pro správu přístupových údajů pro všechny komponenty
  - Prometheus – nástroj pro sběr a krátkodobé úložiště metrik
  - Alertmanager – nástroj pro správu automatizovaných upozornění
  - Thanos – nástroj pro dlouhodobé ukládání metrik
  - Grafana – nástroj pro zobrazování metrik a grafů

- Redmine – wiki a správa incidentů
- Redmine Webhook Proxy – vlastní komponenta pro napojení Alertmanageru a Redmine
- Elasticsearch – úložiště logů a audit záznamů
- Logstash – nástroj pro transformaci logů a audit záznamů
- Filebeat – nástroj pro sběr logů z jednotlivých komponent
- Audit Shipper – vlastní komponenta pro přesun audit záznamů z modulových DB schémat do centrálního úložiště auditních záznamů, napr. Elasticsearch
- Kibana – nástroj pro zobrazení a vyhledávání logů a audit záznamů
- Gatekeeper – nástroj pro vynucování pravidel pro všechny součásti Kubernetes

3.3 Principy komunikace mezi službami/moduly

- Požadavky (a cíle)
  - (A) standardizované protokoly
    - široká kompatibilita mezi moduly
    - odolnost proti vendor lock-in
- Technologie
  - REST API a GraphQL (synchronní komunikace)
  - Fronta (asynchroní komunikace)
  - Webhooks (3rd party)

3.3.1 Synchronní komunikace

- REST API/GraphQL
  - Požadavek na podporu sticky sessions v některých edge-cases
  - Služby by mezi sebou neměly v rámci requestu od uživatele provést více jak 2 synchronní volání.
    - Každým synchronním voláním v rámci systému klesá spolehlivost pro koncové uživatele. *Příklad: Služba A volá během požadavku službu B a C. Služba A má uptime 98 %, Služba B má uptime 99 % a Služba C má uptime 98 %. Výsledná spolehlivost Služby A pro koncové uživatele je 0.98*0.99*0.98=0,95 => 95 %*
    - V případě nutnosti lze použít data sourcing anebo saga pattern nad frontou.

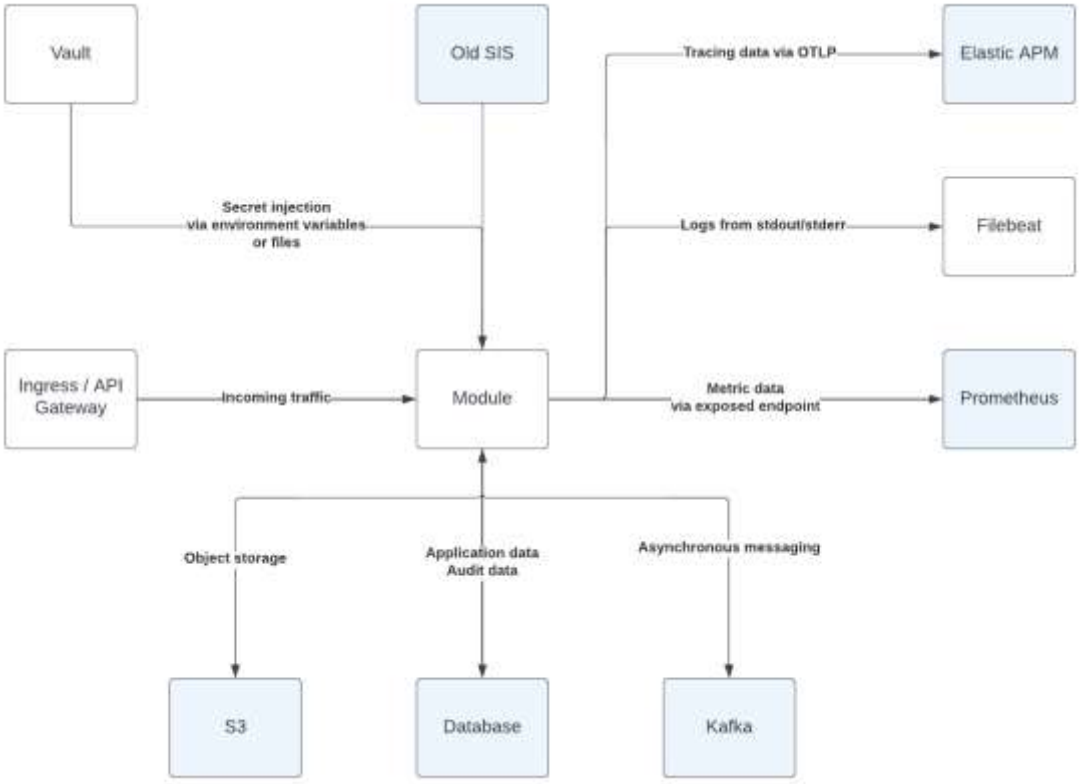
3.3.2 Fronta (asynchronní komunikace)

- Požadavky (a cíle)
  - (A) garance alespoň jednoho doručení
    - odolnost vůči výpadkům infrastruktury
  - (B) UI pro zobrazení
    - monitoring a diagnostika
  - (C) kompatibilita s logging řešením
    - korelace logů s ostatními moduly

    - (D) kompatibilita s monitoring řešením
        - korelace statistických indikátorů s ostatními moduly
    - (E) kompatibilita s tracing řešením
        - korelace latence s ostatními moduly
    - (F) high availability
        - odolnost vůči vysoké zátěži
- Technologie
    - Kafka
- Infrastruktura
    - Kafka jako hlavní nástroj pro asynchronní komunikaci
    - Nasazená může být jak pomocí operátoru (strimzi nebo koperator), tak samotného helm chartu
    - Musí exportovat metriky a logy do centrálních řešení
    - V případě nutnosti velké samostatné Kafky je možné přesunout na samostatný Kubernetes cluster nebo VM

3.3.3 Module diagram

- Diagram popisuje komponenty infrastruktury, se kterými budou moduly přímo interagovat (mimo komunikaci s ostatními moduly).
    - Modře zvýrazněné komponenty vyžadují přímou integraci do kódu modulu.



3.4 Autentizace

- Požadavky (a cíle) **na infrastrukturu**
  - *požadavky **na moduly** níže v kap. 4 dokumentu*
  - (A) podpora OpenID Connect (včetně login flow)
    - standardizovaný protokol – ochrana před vendor lock-in
    - roky prověřená technologie – nejkritičtější chyby odladěny
    - podpora mobile-first vývoje
  - (B) vystavení a management client ID + secret
    - možnost přidat další moduly do systému
    - zvýšení zabezpečení udělením vlastních credentials každému modulu
    - důvěryhodný management přístupů pro aplikace třetích stran
  - (C) podpora refresh tokenů s dlouhou platností
    - podpora "zůstat přihlášen" v mobilních zařízeních
  - (D) podpora vlastních scopes
  - (E) správa uživatelských rolí
    - centrální správa rolí pro základní autorizaci
  - (F) informace o rolích uživatele čitelná z access token (např. JWS)
    - efektivní autorizace
  - (G) responzivní login screen
    - podpora mobile-first vývoje
  - (H) podpora /tokeninfo
    - podpora bezpečnější autorizace (nedůvěra v podpis JWS/ možnost time outu) u kritické funkcionality
  - (I) podpora resetu hesla uživatele administrátorem i uživatelem samotným
    - efektivní workflow z hlediska uživatele i administrátora
  - (J) podpora druhého faktoru
    - podpora bezpečnější autentizace (nedůvěra ve vlastníka přihlašovacího hesla) u kritické funkcionality
  - (K) high availability
    - dostupnost autentizace i ve vysoké zátěži
  - (L) kompatibilita se stávajícím systémem
    - jednotná autentizace vůči starému i novému systému
- Technologie
  - Stávající (CAS + LDAP)

3.5 Autorizace

- autorizace není řešena centrálně (viz požadavky na moduly níže v kap. 4 dokumentu)

3.6 Logging

- Požadavky (a cíle) **na infrastrukturu**
  - *požadavky **na moduly** níže v kap. 4 dokumentu*
  - (A) parser standardních logovacích výstupů a formátů a metadat modulu (ID modulu, ID kontejneru)
    - zobrazení informací o událostech ve strukturované podobě
    - stdout kontejneru, JSON
  - (B) centrální úložiště logů
    - vzájemná korelace událostí ze všech modulů
    - dostupnost logů i po kritickém selhání a odstranění kontejneru
  - (C) dostatečná kapacita úložiště
    - dostupnost logů po dobu nezbytně nutnou pro diagnostiku (závisí na SLA; doporučujeme alespoň 1 měsíc)
  - (D) UI pro zobrazení a filtrování logů
    - vyhledávání událostí týkajících se konkrétního modulu, kontejneru, časového rozmezí, ID požadavku či úrovně
  - (E) šifrování citlivých údajů v logu za pomocí asymetrické kryptografie.
    - šifrování citlivých údajů v logu
    - pro vybrané údaje bude upřesněno v zadávací dokumentaci jednotlivých modulů
  - (F) high availability
    - dostupnost logů i v případě selhání za účelem diagnostiky
  - (G) kompatibilita se stávajícím systémem
    - korelace událostí se stávajícím systémem
- Technologie
  - ELK stack
    - Elastic
    - Logstash
    - Filebeat
    - Kibana
- Infrastruktura
  - Elastic by měl běžet v HA režimu s minimálně 3 replikami.
  - Pokud narazíme na potřebu složitějšího zpracování nebo přeposílání logů, které nezvládne Filebeat nebo Elastic Ingest pipelines, bude třeba použít Logstash
  - Pro autorizaci do Elasticu a Kibany jsou plánovany 2 realmy:
    - *native* – pro service accounty
    - *OpenID Connect nebo LDAP* – pro uživatele

3.7 Auditing

- Požadavky (a cíle) **na infrastrukturu**
  - *požadavky **na moduly** níže v kap. 4 dokumentu*
  - (A) garance integrity logu

        - průkaznost zalogovaných událostí
    - (B) úložiště s dostatečnou kapacitou
        - dostupnost logů po dobu danou legislativou a směrnicemi
    - (C) API pro asynchronní zápis (append) a čtení
        - efektivní zápis do logu
        - dostupnost auditních událostí pro ostatní moduly
    - (D) UI pro zobrazení a filtrování
        - vytváření reportů daných legislativou a směrnicemi
    - (E) audit přístupů do logu
        - dostupnost informací o přístupech pro případ bezpečnostního incidentu
    - (F) high availability
        - garance dostupnosti logu pod vysokou zátěží
    - (G) kompatibilita se stávajícím systémem
        - korelace auditních událostí se stávajícím systémem
- Technologie
    - PostgreSQL
    - Elastic Search
    - Kafka
    - Kibana

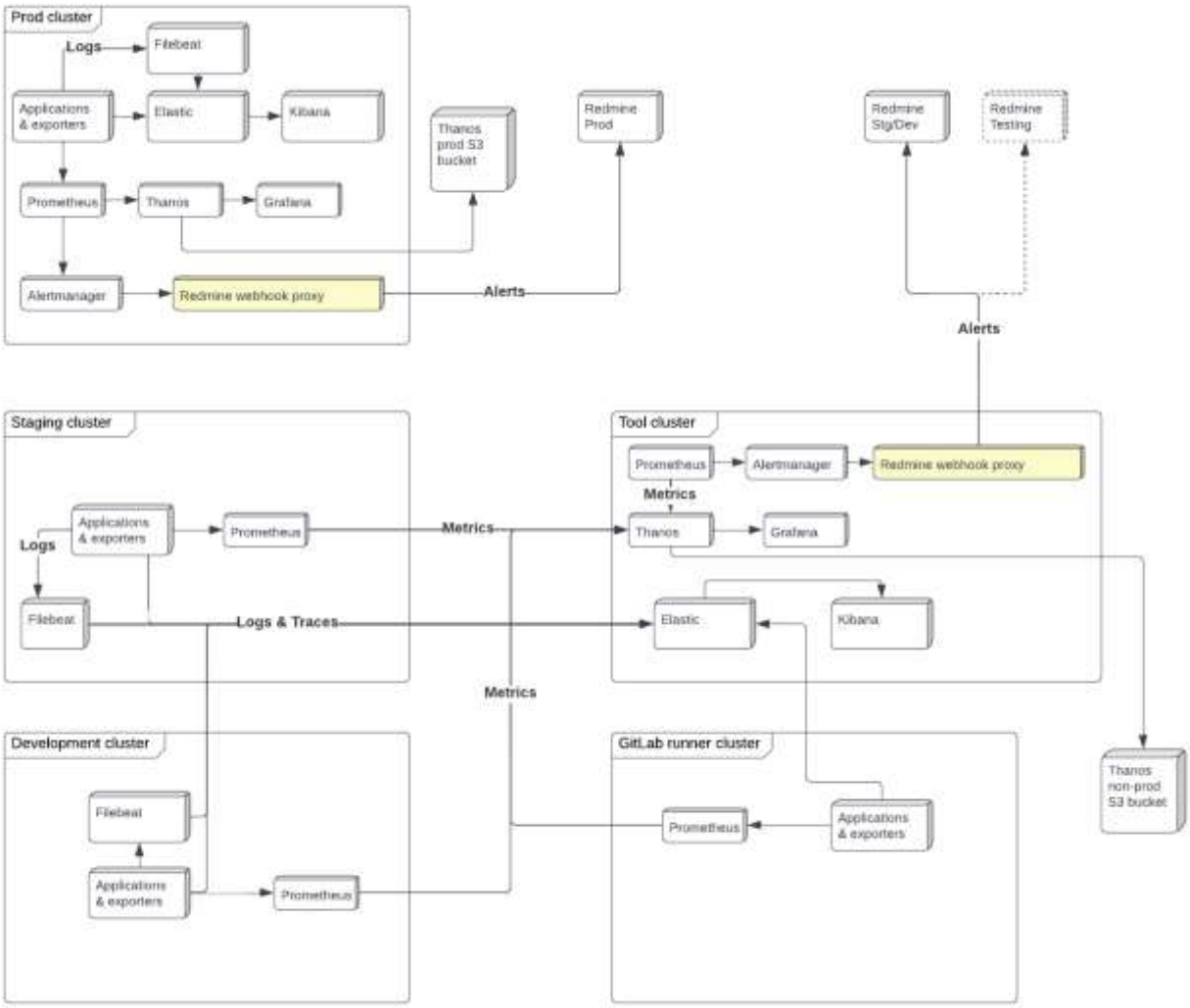
3.8 Tracing

- Požadavky (a cíle) **na infrastrukturu**
    - *požadavky **na moduly** níže v kap. 4 dokumentu*
    - (A) zobrazení trvání requestu a souvisejících metadat
        - korelace latence napříč všemi moduly
        - dostupnost informací i po kritickém selhání a odstranění kontejneru
        - vyhledávání informací týkajících se konkrétního modulu, kontejneru, časového rozmezí či požadavku
    - (B) high availability
        - dostupnost informací i v případě selhání za účelem diagnostiky
- Technologie
    - OpenTelemetry SDK
    - Elastic APM

3.9 Monitoring

- Požadavky (a cíle) **na infrastrukturu**
    - *požadavky **na moduly** níže v kap. 4 dokumentu*
    - (A) dashboards pro efektivní zobrazení číselných řad
        - vzájemná korelace statistických ukazatelů všech modulů, instancí a HW prostředků
        - notifikace v případě překročení povolených mezí
        - různé pohledy pro stakeholdery, správce infrastruktury a vývojáře
    - (B) sledování (tracking) selhání a neočekávaných stavů

        - dostupnost informací o selháních i po odstranění kontejneru
        - zobrazení informací o selháních ve strukturované podobě
        - zobrazení statistických informací o četnosti selhání v jednotlivých modulech
    - (C) zobrazení stavu systému koncovým uživatelům (status page)
        - poskytnutí informací, jež koncovému uživateli pomohou určit, zda chybný stav, jenž nastal, je způsoben chybou na straně systému
    - (D) správa on-call směn
        - notifikace odpovídajících osob držících pohotovost v případě selhání a nechtěných stavů
    - (E) high availability
        - dostupnost informací o stavu systému i v případě selhání za účelem diagnostiky
    - (F) kompatibilita se stávajícím systémem
        - korelace informací se stávajícím systémem
    - (G) API / klientské knihovny / scraper
        - integrace s moduly
- Technologie
    - Grafana – zobrazování grafů a jednoduchých dashboardů
        - Je možné využít i pro jednoduchý status page
    - Prometheus stack – sběr a krátkodobé ukládání metrik
    - Alertmanager – pro správu alertů
        - Pro vytváření ticketů pro incidenty v Redmine bude použitý webhook + vlastní centrální komponenta, která bude komunikovat s Redmine API
    - Thanos – dlouhodobé ukládání metrik
        - Bude používat S3 storage
    - ELK stack
    - Redmine
- Infrastruktura
    - Pro nasazení všech komponent kromě Thanose doporučujeme kube-prometheus-stack
    - Grafana bude napojena na CAS přes LDAP nebo OAuth2
- Monitoring diagram
    - Diagram popisuje tok metrik, logů, traců a alertů celým systémem. Sdílený monitoring pro Development a staging cluster se nachází v tool clusteru. Produkční cluster obsahuje svůj oddělený monitoring.
    - Vlastní komponenty jsou zvýrazněny žlutě.



3.10 Reporting

- Požadavky (a cíle) **na infrastrukturu**
  - (A) infrastruktura by modulům měla nabízet unifikované prostředí pro reporting
  - (B) moduly by měli mít jednoduchý způsob, jak poskytovat data pro reporting např. skrze reporting topic, přímý polling reporting endpointu nebo případně provolávání reporting API
  - (C) reportovací platforma umožňuje online přístup k jednoduchým reportům
  - (D) reportovací platforma umožňuje počítat komplexní reporty skrze CRON
- Technologie
  - Reportovací platforma bude implementována vlastní centrální komponentou
  - Vhodné reportovací výstupy pro doplnění do centrální platformy (splňující požadované principy) uchazeč popíše ve svém návrhu řešení

3.11 Storage

- Požadavky (a cíle) **na infrastrukturu**:
  - (A) poskytnout transakční databázi, která dokáže zvládnout následující zátěž:
    - 1,5 TB
    - přírůstek +100 GB za rok

    - (B) transakční databáze musí podporovat replikaci v rámci transakce a failover to standby
    - (C) platforma musí podporovat ukládání souborů a jiných nestrukturovaných dat standardizovanou formou, včetně řízení přístupu
        - Je třeba podporovat minimálně 7 TB file storage s možností rozšíření
    - (D) platforma poskytuje indexed document databáze pro ukládání logů a auditu z infrastruktury a pro potřebu aplikačních modulů
    - (E) document databáze musí podporovat replikaci v rámci transakce a failover to standby
    - (F) zvolené řešení persistence musí být kompatibilní se zvoleným způsobem zálohování
- Technologie
    - Podporu persistent volumes bude poskytovat Ceph pomocí CSI pluginu.
    - Ceph bude také poskytovat S3-compatible vrstvu pro jednotlivé aplikace a zálohy
    - Transakční DB
        - PostgreSQL
    - Document store
        - Elasticsearch (pokud bude potřeba indexovaný document-based store pro aplikace)
    - File/blob storage
        - S3 – Ceph
- Infrastruktura
    - Databázové instance jsou centrálně spravovány
    - Každý modul má oddělené databázové schéma
    - Umisťování schémat modulů do instancí je řízeno centrálně
    - Menší moduly můžou využívat sdílenou instanci, větší mají každý vlastní
    - Pokud budou databáze spuštěny v Kubernetes, je doporučeno použití postgres-operator (zalando/postgres-operator nebo CrunchyData/postgres-operator)

3.12 Reverse Proxy / API Gateway

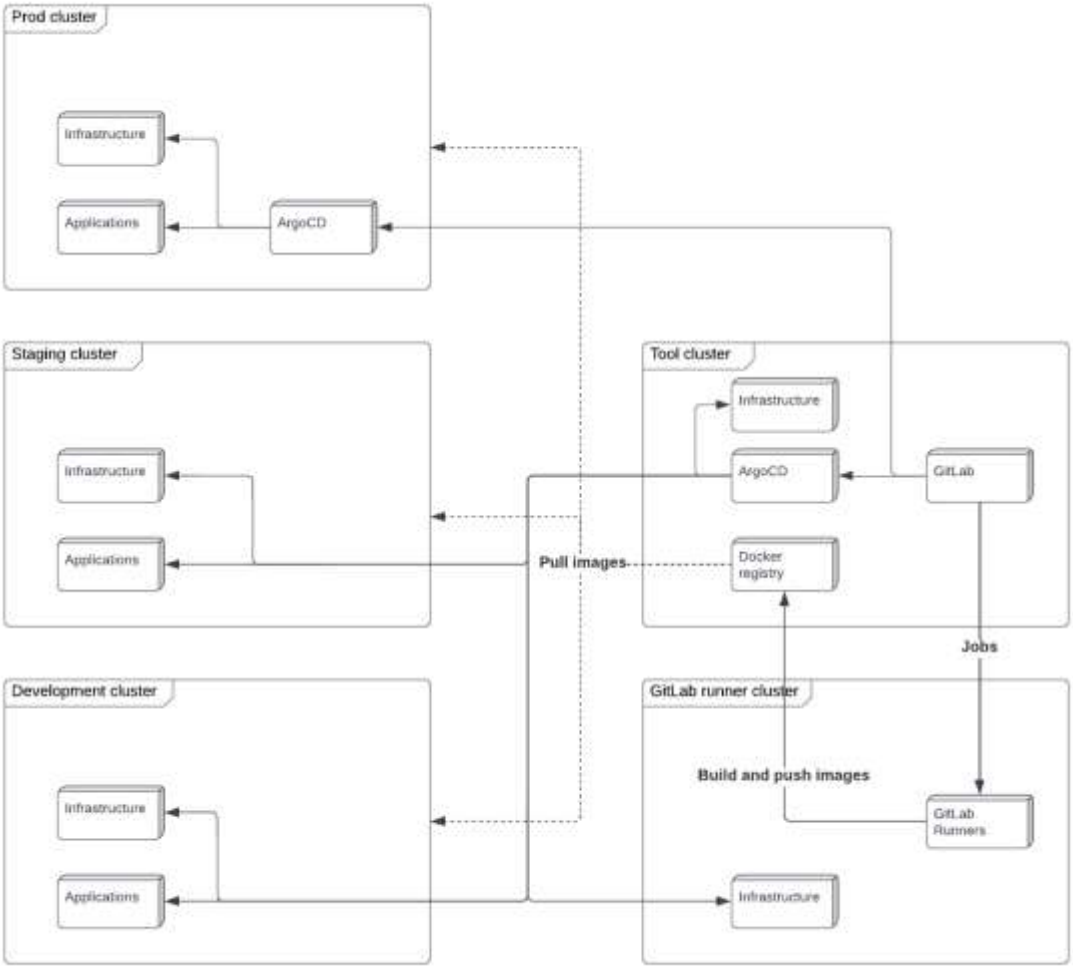
- Požadavky (a cíle)
    - (A) integrace s IdP
        - deduplikace autentizace
        - možnost napojení gateway na IDP skrze OpenI
    - (B) komunikace zabezpečená TLS
        - zabezpečení před zcizením citlivých údajů
    - (C) kompatibilita s logging řešením
        - korelace logů s ostatními moduly
        - zaznamenání requestů do logu včetně cesty volání status kódu a doby trvání
    - (D) kompatibilita s monitoring řešením
        - korelace statistických indikátorů s ostatními moduly
    - (E) kompatibilita s tracing řešením

        - korelace latence s ostatními moduly
    - (F) high availability
        - odolnost vůči vysoké zátěži
- Technologie
    - Zvolený ingress controller je ingress-nginx
        - Externí traffic by měl být směřován na NodePort controlleru
        - Je třeba nastavit nginx aby věřil příchozím X-Forwarded-For hlavičkám pro propagaci zdrojových IP adres klientů
    - Ukončení TLS by mělo být provedeno na předřazené externí proxy (HAProxy)
    - Vybraná API Gateway je Kong
        - Měla by být napojená na tracing a monitoring
    - V rámci nasazování by měla být i připravena defaultní role pro uživatele K8s pro správu Kong custom resources

3.13 CI/CD + Deployment

- Požadavky (a cíle)
    - (A) automatizované nasazení nové verze modulu
        - možnost nasazení modulu bez znalosti infrastruktury
    - (B) CI/CD runners
        - vybraná CI/CD platoforma by měla mít možnost spouštět joby na dedikovaných runnerech, které se dají jednoduše škálovat.
    - (C) Docker registry
        - dostupnost Docker images pro orchestraci
    - (D) high availability
        - garance možnosti nasazení a dostupnosti zdrojových kódu
- Technologie
    - GitLab
    - ArgoCD
- Infrastruktura
    - GitLab pro hosting Git repozitářů a jako CI platforma
        - ve free open-source self-hosted verzi
    - ArgoCD bude použito pro nasazování do konkrétního prostředí, ne pro správu více prostředí najednou
    - GitLab Runner
        - Pro bezpečnost při sestavování Docker images bude nasazeno do odděleného Kubernetes clusteru
        - Pro použití DinD musí být alespoň 1 runner v privileged režimu
            - Může být nahrazeno jiným způsobem sestavování Docker images (podman, kaniko) pro zvýšení bezpečnosti
    - Aplikace by měly být nasazeny přes Helm ze sdílených Helm chartů
    - V rámci přípravy infrastruktury vznikne ukázková aplikace, která bude využívat všechny složky architektury modulů

- CI/CD Diagram
    - Diagram znázorňuje 2 ArgoCD instance, které obsluhují všechny clustery a nasazení infrastruktury a modulů. Produkční instance je kompletně oddělená od neprodukční. ArgoCD je napojené na GitLab a synchronizuje stav několika repozitářů s nastavením aplikací do jednotlivých clusterů.



3.14 Bezpečnost

- Požadavky (a cíle)
    - (A) Infrastruktura by měla být zabezpečena proti útokům
    - (B) Infrastruktura by měla poskytovat bezpečné prostředí pro běh služeb
- Standardy a technologie
    - Vault (Secret storage)
    - Let's Encrypt pro SSL spojení s venkovním světem (případně certifikáty vydávané klasickou autoritou)
    - Vlastní docker image repository, kde jsou hostované ověřené docker images
    - Penetrační testy a bezpečnostní audit jednou za X let
- Infrastruktura
    - Vault by měl být použit pro správu všech secretů

        - Podle potřeby by měly být secrety injectovány do jednotlivých modulů pomocí Vault Agent sidecar nebo CSI provideru
    - Všechny Ingressy musí být zabezpečeny pomocí TLS certifikátu
        - z Vaultu
        - z Let's Encrypt pomocí cert-manageru
    - Základní Docker image by měly být dostupné pro všechny zvolené technologie
        - Z důvodu zvýšené bezpečnosti by měly fungovat v rootless režimu
    - Budou nastavené network policies uvnitř Kubernetes pro omezení síťové komunikace
        - Omezit příchozí komunikaci na HAProxy -> Ingress -> API gateway -> modul
        - Omezit přímou komunikaci mezi pody modulů
        - Omezit odchozí komunikaci mimo cluster na určité komponenty
            - Ceph (S3)
            - Původní SIS
            - IDP
    - Použití OPA Gatekeeper pro omezení nasazení modulů do Kubernetes, např.:
        - Omezení Docker Registry
        - Vynucení limitů a rezervací zdrojů CPU a RAM
        - Omezení ingresu na určité domény
        - Omezení verzí Docker image, vynucení použití digestu
        - Vynucení labelů a anotací
        - Vynucení readiness a liveness probe

3.15 High availability

- Požadavky (a cíle)
    - (A) technologie podporují HA
        - odolnost systémů proti výpadkům
- Infrastruktura
    - HA je nutné pro následující komponenty
        - Kubernetes control plane
        - Vault
        - Některé databáze
        - Elastic
        - Ingress controller (nginx-ingress)
    - Ostatním komponentám by mělo dostačovat HA pomocí Kubernetes – přenasazení na jiný node
    - Samozřejmostí je provoz všech aplikačních modulů v HA

3.16 Zálohování

- Požadavky (a cíle)
    - (A) centrální systém pro správu záloh
        - snadné vytváření a správa záloh

    - (B) automatizované zálohování daných komponent v daný čas
        - spolehlivé zálohování bez možnosti lidské chyby
        - promítnutí kritičnosti dat ve frekvenci záloh
        - přesný harmonogram pro zajištění konzistence mezi zálohami
    - (C) bezpečné úložiště
        - garance dostupnosti záloh v případě potřeby
        - zabezpečení před odcizením citlivých dat
    - (D) plán testování funkčnosti záloh
        - garance funkčnosti záloh v případě jejich potřeby
- Technologie
    - Pro zálohování obsahu Kubernetes control plane bude použité Velero
        - bude ukládat zálohy v S3
    - Jednotlivé volumes budou zálohovány v Cephu přes snapshoty
    - Aplikace, které to podporují (GitLab, Postgres Operator), budou zálohovat do S3
    - S3 případně snapshoty mohou být zálohovány na páskách

3.17 Developer experience

- Požadavky (a cíle)
    - (A) V novém systému by mělo jít snadno vyvíjet.
    - (B) Vývojáři by se měli snadno v ekosystému zorientovat, získat potřebnou dokumentaci a pravidla vývoje
- Technologie
    - Pro sdílení informací bude používána dedikovaná instance Redmine, která bude obsahovat popis pracovních postupů, infrastruktury atd.

4 Požadavky na služby/moduly vyvíjené v nové architektuře

4.1 Backendy

- Požadavky (a cíle)
    - (A) udržitelné technologie
        - podpora technologie za 10 let
    - (B) kompatibilita s container-friendly OS
        - kompatibilita s CI/CD a orchestrací
    - (C) další požadavky pro integraci popsané detailně v jednotlivých sekcích níže
- Technologie
    - Java/Kotlin (+ Spring Boot)
    - TypeScript + Next.js/Nest.js
    - Python + WSGI/ASGI
    - C# + ASP.NET Core
    - PHP (výhradně s využitím objektů)
        - base images containerů: poslední stable Alpine, Debian nebo Ubuntu

4.2 Frontendy

- Požadavky (a cíle)
  - (A) udržitelné technologie
    - podpora technologie za 10 let
  - (B) proprietární administrátorské obrazovky generovány backendem skrze API
    - jednotný design
    - úspora prostředků
  - (C) knihovna společných komponent
    - jednotný design
      - bude použitý koncept Atomic Design s postupně rozšiřující se knihovnou jednotlivých komponent a šablon
    - úspora prostředků
  - (D) web: stejná UX v prohlížečích Edge, Chrome, Firefox a Safari (aktuální verze a verze -1) v desktop OS Windows, MacOS i Linux
    - dostupnost pro všechny uživatele
  - (E) web: stejná UX v prohlížečích Chrome a Safari na iOS 13+ i Android 10+ na telefonech a tabletech
    - dostupnost pro všechny uživatele
  - (F) mobile: aktuálně podpora iOS 13+ a Android verze 10+, garance podpory aktuální verze a verze -1
    - dostupnost pro všechny uživatele
    - potřeba vývoje UI daného modulu pro mobilní aplikaci bude upřesněna v zadávací dokumentaci odpovídajícího modulu
  - (G) volba jazykové mutace
    - dostupnost pro všechny uživatele
    - dostupné mutace budou specifikovány v zadávací dokumentaci každého modulu
  - (H) web: frontendy pro jednotlivé moduly budou vyvíjené pomocí Micro Frontends (MFEs) architektury
    - pro umožnění paralelního vývoje frontendů pro jednotlivé moduly
    - pomocí standardu Web Components (Custom Elements a spol.)
    - runtime integrace pomocí Webpack Module Federation
    - v případě potřeby build-time integrace bude použitá knihovna sdílených komponent, nicméně preferovaným způsobem je runtime integrace
    - MFEs poskytované jednotlivými moduly budou integrovány do hlavního portálu a v případě potřeby také sdíleny pro ostatní moduly
    - jednotlivé MFEs budou ale spustitelné a testovatelné i mimo integraci do hlavního portálu (např. pro potřeby vývoje, QA atd.)
    - potřebné MFEs budou specifikovány v zadávací dokumentaci jednotlivých modulů
- Technologie
  - Web
    - TypeScript + React

        - Webpack
    - Mobile
        - iOS – Swift
        - Android – Kotlin
        - Flutter

4.3 CI/CD + Deployment

- Požadavky (a cíle)
    - (A) Kód musí být kompatibilní s rootless Docker image
    - (B) Každý modul používá standardní šablony pro GitLab CI a standardní Helm chart
- Technologie
    - Docker
    - GitLab
    - Helm

4.4 Autentizace

- Požadavky (a cíle)
    - (A) implementace protokolu OpenID Connect
        - standardizovaný protokol – ochrana před vendor lock-in
        - roky prověřená technologie – nejkritičtější chyby odladěny
        - podpora mobile-first vývoje
    - (B) mobilní aplikace ukládá refresh token v security enklávě zabezpečené biometrií nebo PINem
        - zabezpečení před únikem dlouho platného refresh tokenu
- Technologie
    - klientské knihovny doporučené na webu OpenID
    - oficiální SDK dané mobilní platformy

4.5 Autorizace

- Požadavky (a cíle)
    - (A) základní autorizace na základě příslušenství uživatele do skupin v rámci LDAP
        - centrální správa uživatelských rolí – efektivní správa
    - (B) jemnější autorizace na základě aplikačně-specifických rolí odvozených od rolí/skupin LDAPu
        - podpora jemnějších přístupových oprávnění
        - zachování centrální správy základních rolí
    - (C) uživatelské rozhraní pro správu aplikačně-specifických rolí a jim přidělených oprávnění
        - uživatelsky přívětivá administrace

4.6 Logging

- Požadavky (a cíle)
    - (A) zalogování zahájení a ukončení každé transakce/requestu s úrovní INFO včetně ID uživatele a status code
        - (transakcí se rozumí každá naplánovaná aktivita (job), aktivita iniciovaná uživatelem i aktivita iniciovaná jinou komponentou systému)
        - dostupnost informací o činnosti modulu pro potřeby zpětné diagnostiky neočekávaných a "náhodných" stavů
    - (B) zalogování zahájení a ukončení komunikace s ostatními moduly a systémy třetí strany s úrovní DEBUG
        - dostupnost informací o činnosti modulu pro potřeby zpětné diagnostiky neočekávaných a „náhodných" stavů
    - (C) zalogování chybových a „deprecated" stavů, z nichž se modul zotavil, s úrovní WARNING
        - viz (B)
    - (D) zalogování chybových stavů, z nichž se modul nezotavil, s úrovní ERROR včetně stacktrace
        - viz (B)
    - (E) zalogování všech ostatních zpráv, které pomohou případné diagnostice neočekávaných stavů, s úrovní DEBUG
        - viz (B)
    - (F) každá zpráva obsahuje časovou značku, úroveň, ID instance, ID požadavku a popis
        - korelace a filtrování logů napříč moduly
        - musí obsahovat correlation ID
    - (G) každá zpráva je ve formátu JSON
        - přesný formát logovacího JSON bude přílohou zadávací dokumentace modulu
        - čtení logů ve strukturované podobě
        - logování víceřádkových zpráv
    - (H) zprávy obsahující tajné anebo velmi citlivé údaje musí být šifrovány dodaným šifrovacím klíčem a musí jim předcházet stejná nešifrovaná zpráva bez citlivých údajů
        - zabezpečení citlivých údajů před správci infrastruktury
        - citlivé údaje s potřebou šifrování budou specifikovány v zadávací dokumentaci modulu
    - (I) zprávy se logují na standardní výstup
        - nezávislost na centrálním úložišti
        - rychlost logování
        - široká podpora napříč technologiemi
        - odolnost vůči výpadkům infrastruktury
    - (J) Frontendy standardně logují do vývojářské konzole a do bufferu, v produkci je log vypnutý do konzole vypnutý. Je možné odeslat log z bufferu k diagnostice.

- (K) Úroveň logování musí být nastavitelná pomocí proměnné prostředí

4.7 Auditing

- Požadavky (a cíle)
  - (A) Pro ukládání audit logů bude použit outbox pattern s outbox tabulkou v databázi modulu
    - dostupnost logů pro situace dané legislativou a směrnicemi
    - auditní události budou specifikovány v zadávací dokumentaci modulu dle legislativy a směrnic
  - (B) Každý záznam o události obsahuje časovou značku, ID instance, typ události, ID uživatele, ID požadavku, akci, popis a další položky dle specifikace modulu
    - záznamy obsahují všechny informace vyžadované legislativou a směrnicemi
  - (C) Přesun audit logů z outbox tabulek do centrálního úložiště auditních záznamů obstará vlastní centrální komponenta, která zajistí konzistenci dat.
  - (D) Audit musí obsahovat strojově zpracovatelné correlation ID pro korelaci s logy
  - (E) Každá zpráva musí být zapsána v jedné transakci společně s daty

4.8 Tracing

- Požadavky (a cíle)
  - (A) odeslání informací o každém požadavku do centrálního úložiště
    - dostupnost informací o trvání požadavků
  - (B) informace o požadavku obsahují časovou značku, ID instance, ID unikátní v rámci celého systému, ID uživatele, status code a informace o požadavcích vyvolaných tímto požadavkem
    - snadnost identifikace konkrétního požadavku
  - (C) Každý modul poskytuje tracing data v OpenTelemetry standardu pomocí OTLP/gRPC protokolu
    - Tracing musí být vypínatelný a nastavitelný pomocí proměnných prostředí
    - Trace musí obsahovat correlation ID
    - Traces jsou automaticky ukládány do Elastic APM a zpřístupněny k nahlédnutí v Kibaně
  - (D) Všechny příchozí a odchozí requesty obsahují correlation ID
    - Na vstupu do systému bude generováno pomocí API gateway
- Technologie
  - OpenTelemetry SDKs
  - Tracing data by se měly sbírat v Elastic APM

4.9 Monitoring

- Požadavky (a cíle)

    - (A) Každý modul vystavuje endpoint s metrikami v Prometheus formátu, nebo posílá metriky na Pushgateway
        - informace o stavu a vytížení modulu pro dashboards
        - metriky budou specifikovány v zadávací dokumentaci každého modulu
        - tvůrce modulu může po domluvě dodat i další metriky související s implementací daného modulu pro informovanější monitoring a debugging
        - tvůrce modulu musí dodat dokumentaci poskytovaných metrik (jednotky, způsob měření, přesná sémantika atd.) a doporučení pro nastavení alertů pomocí PromQL
    - (B) report selhání a neočekávaných stavů
        - maximum informací o selhání pro potřeby diagnostiky
        - selhání nejsou přehlédnuta
    - (C) actuator endpoints
        - informace o stavu modulu pro status page a orchestraci
- Technologie
    - klientská knihovna Promethea

4.10 Reporting

- Požadavky (a cíle)
    - (A) modul musí poskytovat data pro sdílenou reporting infrastrukturu v takové míře, aby bylo možné splnit požadavky na reporting od stakeholderů – typicky se bude jednat o data z monitoringu a auditu, nicméně ve vybraných případech i o jiný způsob vhodného poskytnutí dat pro potřeby požadovaných reportů (bude upřesněno v zadávací dokumentaci modulu)

4.11 Storage

- Požadavky (a cíle)
    - (A) veškeré změny nutné pro instalaci nebo upgrade se provádí automatizovanými migračními skripty
        - reprodukovatelnost instalace na dalších prostředích
    - (B) prostředky pro správu osobních údajů
        - splnění požadavků dle GDPR (výpis a výmaz osobních údajů)
    - (C) modul pracuje s vlastním databázových schématem centrálně spravovaným a umisťovaným do některé z centrálně spravovaných databázových instancí PostgreSQL
    - (D) modul nesmí ukládat žádná persistentní data na lokální disk, pouze dočasná – jejich objem a typ je potřebné specifikovat v doložené technické dokumentaci
    - (E) pro ukládání souborů je přístupné S3 úložiště
    - (F) pro integraci se starou verzí SIS bude použita přímá integrace na Oracle databázi pomocí:
        - Připravených Stored Procedures
        - Přesně definovaných SQL jakožto součásti zadávací dokumentace

        - Ve většině případů bude použitý CQRS pattern (viz kapitola 5 „Principy dělení SIS na jednotlivé moduly“)

4.12 Testování a dokumentace

- Požadavky (a cíle)
    - (A) pokrytí kódu unit a integračními testy
        - zvýšená šance odhalení regresních závad na úrovni modulu při úpravách a refactoringu
        - (procento pokrytí testy a procento automatizace testů je třeba specifikovat v zadávací dokumentaci každého modulu)
    - (B) pokrytí všech user journeys a automatických procesů end-to-end testy
        - zvýšená šance odhalení regresních závad na úrovni vzájemné integrace modulů při úpravách a refactoringu
        - (procento automatizace testů je třeba specifikovat v zadávací dokumentaci každého modulu)
    - (C) pokrytí výkonnostně citlivých částí výkonnostními testy
        - zvýšená šance odhalení výkonnostních nedostatků při úpravách a refactoringu
        - (výkonnostní požadavky a procento automatizace testů je třeba specifikovat v zadávací dokumentaci každého modulu)
    - (D) testy funkčních požadavků je možné spouštět proti lokální instanci modulu (i díky dodaným mockům)
        - možnost otestovat před nasazením do prostředí
    - (E) instalační příručka
        - znalost systémových požadavků
        - znalost způsobů konfigurace modulu
        - znalost postupu instalace na lokální stanici a ve vývojových prostředích
    - (F) dokumentace funkčních a nefunkčních požadavků
        - zvýšená šance odhalení regresních závad na úrovni modulu
        - znalost happy paths i unhappy paths
    - (G) dokumentace API standardizovaným strojově čitelným formátem
        - znalost API dostupná vývojářům ostatních modulů
        - srozumitelnost dokumentace vývojářům ostatních modulů
        - zvýšená šance odhalení regresních závad na úrovni modulu
        - možnost generovat klienty API automaticky
    - (H) dokumentace umístění dat používaných modulem a možnosti jejich anonymizace
        - možnost zálohovat data využívaná modulem
        - splnění požadavků dle GDPR před provedením zálohy
    - (I) dokumentace testovací strategie a testovacích scénářů
        - záruka kvality dodaného díla
        - reprodukovatelnost testů

- Technologie a standardy
    - Gherkin – pro popis testovacích scénářů
    - OpenAPI 3.0 / GraphQL Schema + playground
    - Markdown
    - UML diagramy
    - README.md
        - Informace pro vývojáře a operations, co daný modul dělá, jak ho rychle spustit ve vývojovém prostředí a jak pro něj vyvíjet.

4.13 Dostupnost a spolehlivost (High availability)

- Požadavky (a cíle)
    - (A) modul je možné provozovat ve více instancích zároveň
        - horizontální škálování
    - (C) Žádné používání sticky session, pokud lze
        - Služby by měly být stateless z důvodu jednoduché škálovatelnosti.

4.14 Výkon

- Požadavky (a cíle)
    - (A) modul je možné provozovat ve více instancích zároveň
        - horizontální škálování
    - (B) modul zvládne určité množství requestů za minutu
        - specifikováno v zadávací dokumentaci
    - (C) modul má dostatečně rychlou odezvu v ms na daných endpointech
        - specifikováno v zadávací dokumentaci per skupina endpointů ve formě: xx percentil requestů a očekávaný čas v ms

4.15 Použitelnost

- Požadavky (a cíle)
    - (A) Frontendy musí mít responzivní design, pokud se nejedná o administrátorskou aplikaci, která se bude používat striktně na desktopu
    - (B) Frontendy splňují pravidla přístupnosti dle WCAG (https://www.w3.org/WAI/standards-guidelines/wcag/), pokud se nejedná o administrátorskou aplikaci, která se bude používat striktně na desktopu
    - (C) K modulům a aplikacím existuje uživatelská dokumentace a návody v dostatečné míře
    - (D) mobile: stejná UX na displejích s rozlišením 360x640, 375x667, 414x896, 601x962, 768x1024 a 800x1280 v natočení zařízení na výšku
    - (E) web: stejná UX na displejích minimálně 360x640

4.16 Konfigurace

- Požadavky (a cíle)
    - (A) veškeré parametry se konfigurují pomocí proměnných prostředí

        - (zejména URL a přístupové údaje k ostatním modulům a aplikacím třetích stran a parametry vycházející z funkčních a nefunkčních požadavků)
        - kompatibilita s orchestrací kontejnerů
        - kompatibilita se zálohovacím řešením
        - možnost změny bez zásahu do zdrojového kódu
    - (B) proměnná prostředí umožňující přepnout úroveň logování (viz logging)
        - možnost změnit úroveň logování bez zásahu do zdrojového kódu
    - (C) změnu konfigurace a nasazení nové verze lze provést bez downtime
        - vysoká dostupnost

4.17 Kvalita kódu

- Požadavky (a cíle)
    - (A) kód prochází kontrolou linteru a nástroje na statickou analýzu kódu
        - odhalení chyb v compile time
    - (B) kód prochází kontrolou nástroje na formátování
        - zefektivnění vývoje (code reviews) - čitelnější kód, méně diskusí nad formátováním
- Technologie
    - Standardní linter a analyzer kódu, který je dostupný pro každou z platforem (pro každý jazyk/technologii bude odlišný)

4.18 Bezpečnost

- Požadavky (a cíle)
    - (A) modul by měl být napsán tak, aby byl odolný vůči známým bezpečnostním hrozbám
    - (B) kód prochází kontrolou static vulnerability scan (neobsahuje závažné/critical chyby)
    - (C) kód prochází kontrolou dependency vulnerability check (neobsahuje závažné/critical chyby)
    - (D) kód prochází kontrolou dynamic vulnerability analysis (neobsahuje závažné/critical chyby)
    - (E) modul používá knihovny třetích stran, k nimž je poskytována LTS (long term support), nebo u nichž lze předpokládat střednědobá podpora (5-10 let) (např. na základě popularity použití dané knihovny v rámci komunity)
    - (F) dodavatel dodá reporty z úspěšného ověření bezpečnosti u bodů B, C, D. U false-positive případů je poskytnuto písemné odůvodnění.
- Standardy a technologie
    - OWASP
        - TOP10 jako základ
        - V zadávací specifikaci každého modulu by mělo být uvedeno, na jaký level bezpečnosti dle OWASP cílí
    - Typické nástroje pro static vulnerability scan:
        - IBM AppScan, Fortify, Veracode, FindSecBugs, Brakeman, Bandit

- Typické nástroje pro dependency vulnerability check:
  - OWASP Dependency Check, Bundler Audit, Safety
- Typické nástroje pro dynamic vulnerability scan:
  - Burp Suite, OWASP ZAP, HP WebInSpect

5 Principy dělení SIS na jednotlivé moduly

Zde uvádíme několik obecných doporučení, které je dobré dodržovat při rozdělování stávajícího SISu na moduly:

- Moduly je dobré dělit po business doménách.
- Je dobré se zamyslet, které domény/use casy spolu úzce souvisí a měl by je tedy obsluhovat jeden modul.
- Je dobré zvolit vhodnou granularitu modulů. Je lepší mít několik větších modulů než velké množství velmi malých mikroslužeb.
- Každý modul by měl mít svoje vlastní databázové schéma, které je logicky odděleno od ostatních databází. Modul vlastní svoje data a jediným způsobem změny dat je skrze jeho API vystavené na aplikační vrstvě. Je zakázaný přímý přístup do DB.
- Každý modul bude poskytovat vlastní uživatelské rozhraní (frontend), které bude integrováno do hlavního portálu pomocí techniky Micro Frontends.
- Moduly by neměly mezi sebou mít cyklickou závislost synchronních volání. Pokud mají, je to známka, že by se měly sloučit do jednoho modulu. V případě potřeby mohou moduly emitovat data do topiců front a tím se cyklické závislosti vyhnout.
- Současný SIS by neměl vědět nic o nově vznikajících modulech (neměl by na ně mít závislost). SIS by měl vždy být volán novými moduly.
- Je vhodné použít data sourcing pro zmenšení počtu volání, která se musí při vykonání use casu volat z modulu.
- Pro přístup do současného SISu lze dnes použít pouze přímý přístup do databáze.
  - Je vhodné tento přístup v maximální možné míře omezit.
  - Pokud se modul může omezit pouze na READ přístup a nemusí synchronizovat data zpět, je tento přístup preferován nad WRITE přístupem.
  - WRITE přístup je vhodné realizovat za pomocí API na základě stored procedures, aby byla zjevná plocha integrace a API bylo jasně vymezené a zdokumentované. Vlastníkem WRITE API zůstává starý SIS tak, aby udržel konzistenci dat.
  - V případě READ/WRITE přístupu, kdy modul musí mít i svoji databázi, stojí za to zvážit CQRS pattern, který by mohl být vhodný pro řešení konzistence mezi dvěma databázemi. Modul tak zapíše skrze WRITE API do SISu a databáze SISu vyemituje událost se zapsanými daty, které skrze frontu modul zpracuje a zapíše do své vlastní databáze.