Zřízení školního informačního systému

Semestrální práce
NSS

letní semestr 2023/2024

Bielená, Hoang, Le, Štrobl

OBSAH

1. Strategický záměr (stav „TO BE“)

Cílem aplikace je vytvořit efektivní nástroj pro správu učitelů, kurzů a studentů, který umožní snadnou registraci na kurzy, sledování akademického pokroku a zadávání známek.

Systém je určen školám a vzdělávacím institucím, které potřebují efektivní a jednoduchý způsob správy svých kurzů, učitelů, studentů a místností. Může být užitečný jak pro vysoké školy, tak pro střední a základní školy, které potřebují řídit své vzdělávací aktivity.

2. Obchodní přínos

Aplikace by měla lépe zpřístupnit všechny potřebná data o kurzech, učitelích a studentech, které budou centralizovány v jednotném systému. Díky automatizaci procesů, jako je plánování kurzů a alokace zdrojů, bude správa vzdělávacích aktivit rychlejší a efektivnější. Tento systém přinese školám možnost účinněji využívat čas a zdroje, což povede k lepší organizaci výuky a zvýší celkovou produktivitu vzdělávacího procesu.

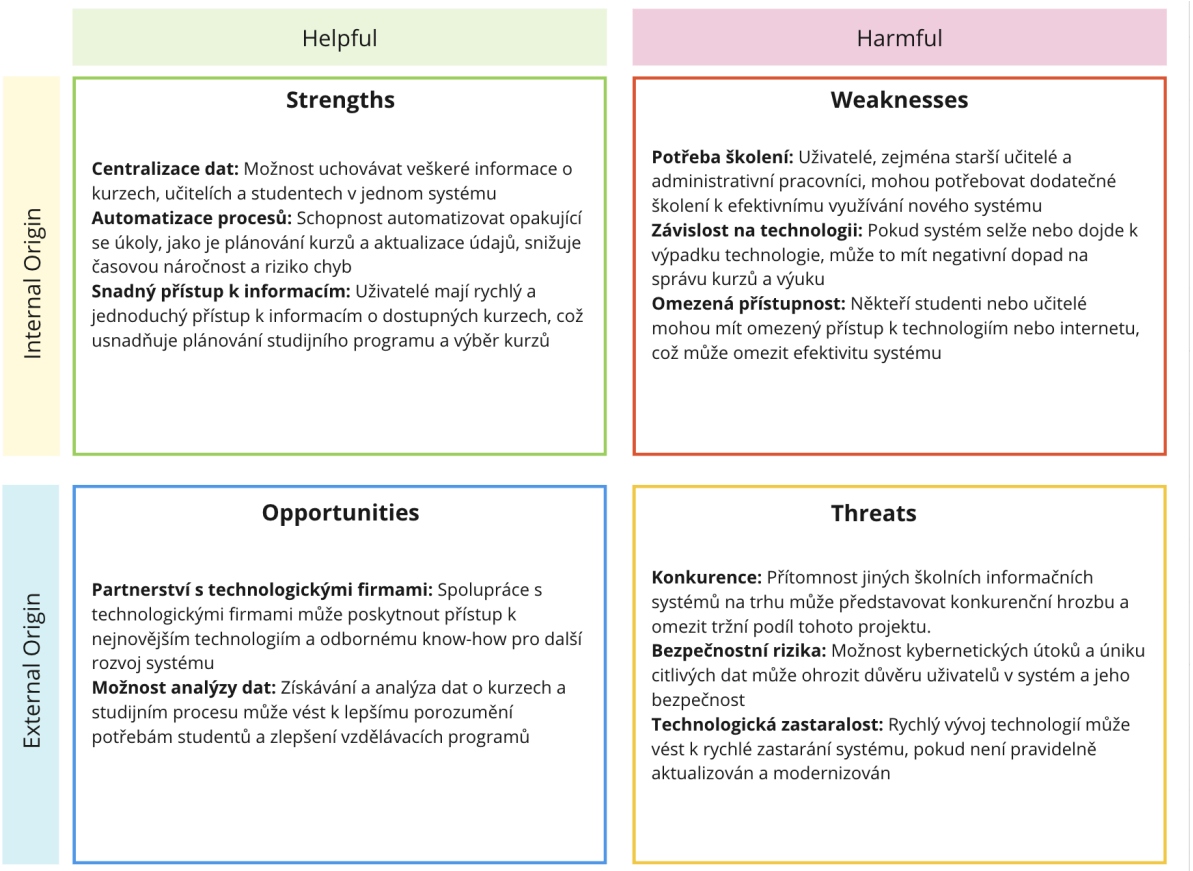
3. Stav „AS IS“

V současné době je správa údajů o studentech, učitelích a kurzech založena na ručních procesech a papírové dokumentaci, což způsobuje ztrátu času a zvyšuje riziko chyb.

V reálném případě můžeme představit situaci ve střední škole, kde se studenti registrují do kurzů prostřednictvím papírových formulářů, které musí ručně vyplnit a odevzdat do kanceláře školní administrativy.

Tento manuální proces znamená, že existuje vysoké riziko chyb při zápisech a aktualizacích údajů. Například může dojít k chybám při ručním přepisování informací, ztrátě dokumentů nebo nesouladům v datech.

4. Analýza SWOT



5. Analýza 5F

- Konkurence
  - Velké množství konkurenčních firem: Na trhu je vysoká konkurence způsobená přítomností mnoha podobných firem nabízejících podobné školní informační systémy
- Síly dodavatelů

    - Nejsme závislí na dodavatelích: Projekt není závislý na konkrétních dodavatelích, což znamená, že nemá významný vliv na sílu dodavatelů na trhu
- Síly odběratelů
    - Cílová skupina: Vysoké školy
    - Přechod ke konkurenci by neměl být tak složitý, největší problém by spočíval v migraci dat
- Substituté
    - Možnost substituce není tolik reálná, neexistuje momentálně na trhu vhodný nekomerční produkt
- Nově příchozí
    - Nově vstupující firmy mohou využívat nižší vstupní náklady a menší administrativní bariéry

6. Analýza PEST(E)

Využívána k systematickému hodnocení vnějších faktorů, které mohou ovlivnit organizaci, projekt nebo trh

Politické faktory: Dodržování zákonných požadavků na ochranu dat, přizpůsobení se změnám v legislativě

Ekonomické faktory: Náklady na vývoj a provoz aplikace, snižování administrativních nákladů univerzity

Sociální faktory: Ulehčení akademického procesu pro studenty a učitele, zlepšení komunikace mezi účastníky

Technologické faktory: Použití moderních technologií pro zajištění bezpečnosti a efektivity aplikace

7. Funkční požadavky

1. Registrace a přihlášení učitelů, studentů i administrátorů
2. Vytváření a úprava kurzů a paralelek učitelem
3. Plánování kurzů do místností s omezenou kapacitou
4. Možnost zápisu studenta do kurzu
5. Zobrazení seznamu kurzů do kterých je student zapsán daný semestr.
6. Zobrazení seznamu kurzů, které učitel vyučuje.
7. Kontrola dostupnosti kapacity v kurzech a místnostech.
8. Možnost vyhledávání kurzů dle různých parametrů
9. Známkování výkonu studentů učiteli
10. Přehled studijních výsledků pro studenty
11. Vytváření entit pro třídy a semestry administrátory
12. Mazání studentů administrátory
13. Podpora českých a anglických znaků.

8. Nefunkční požadavky

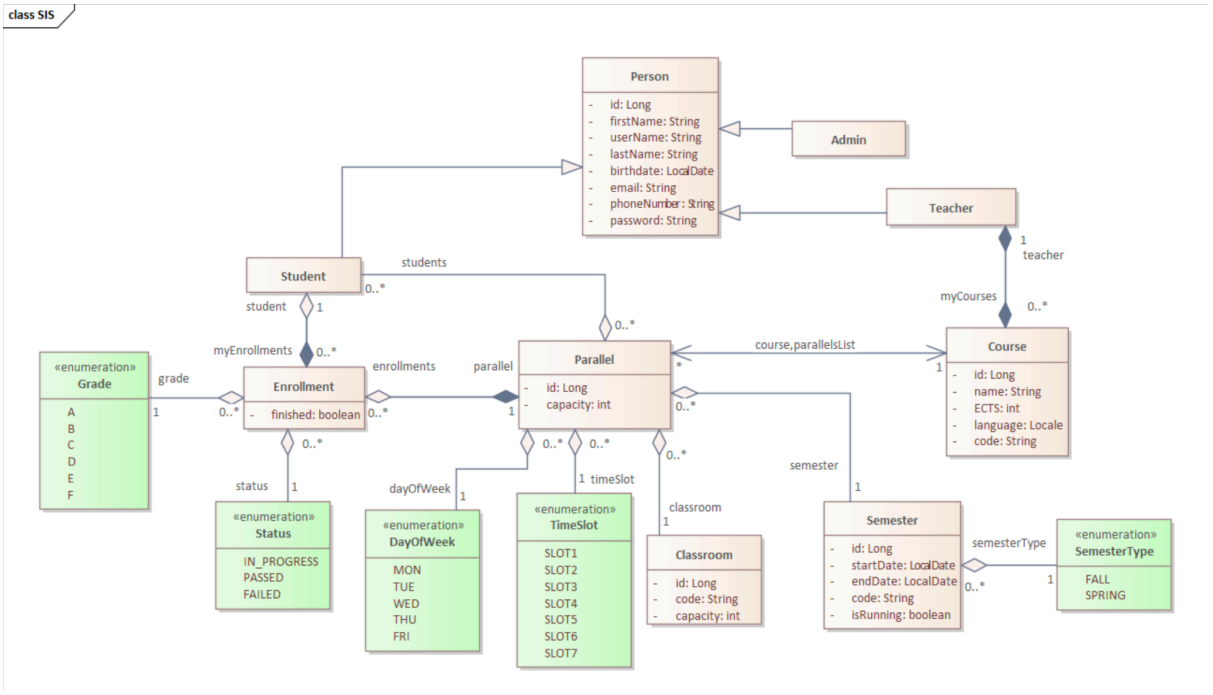
1. Zabezpečení osobních dat uživatelů - Hesla se budou posílat zašifrované šifrováním basic a ukládat jako Bcrypt hash
2. Vysoká dostupnost aplikace a stabilita (zejména v špičce např. zkouškové období) - okolo 95% (rocne)
3. Rychlá odezva systému na uživatelské požadavky - do 2s, slozitejsi do 5s (definovat)
4. Snadné použití a intuitivní uživatelské rozhraní - dosáhnout průměrného hodnocení UI od uživatelů 70% aneb 3,5 z 5.

9. Seznam uživatelů

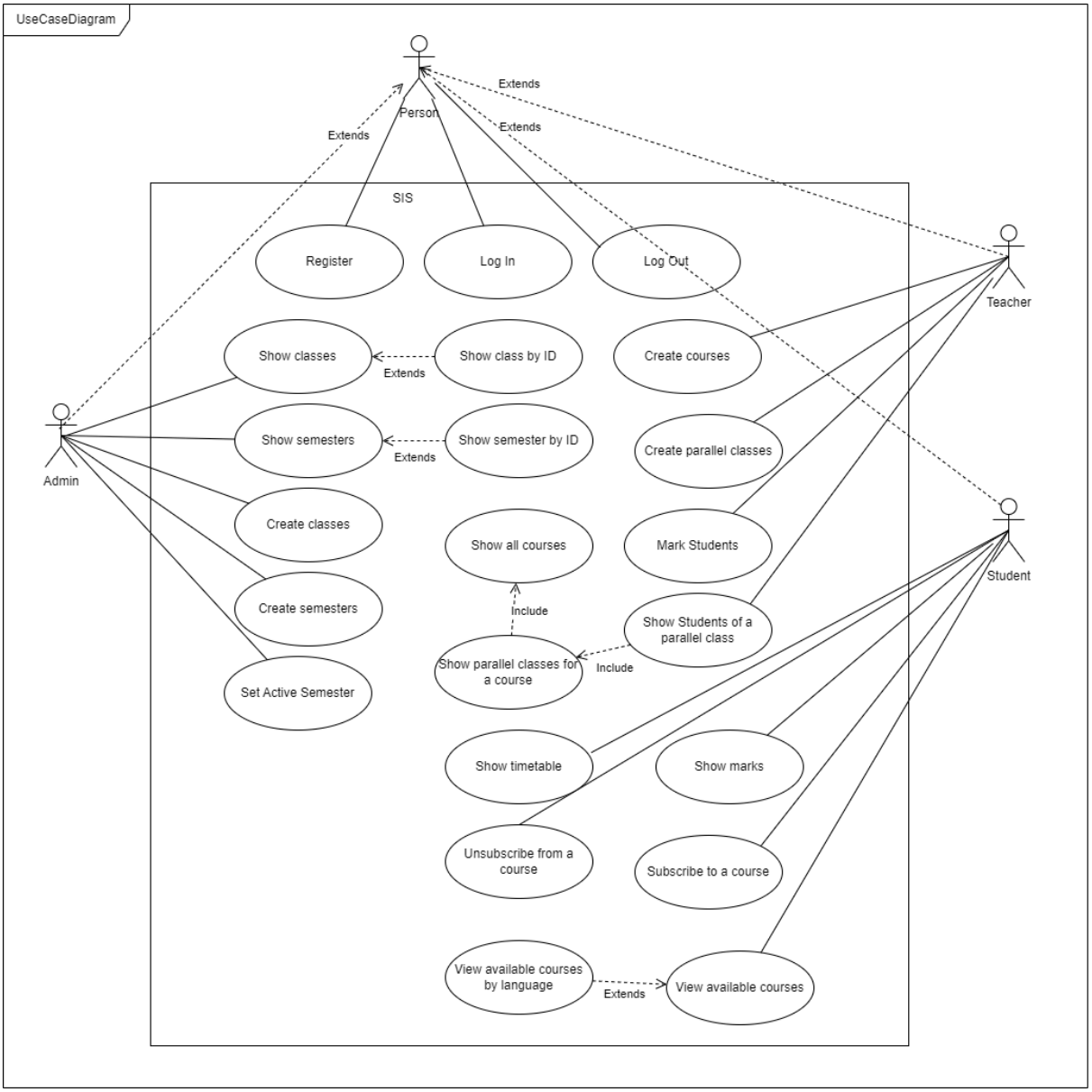
1. Administrátoři
2. Učitelé
3. Studenti

10. UML Class Diagram

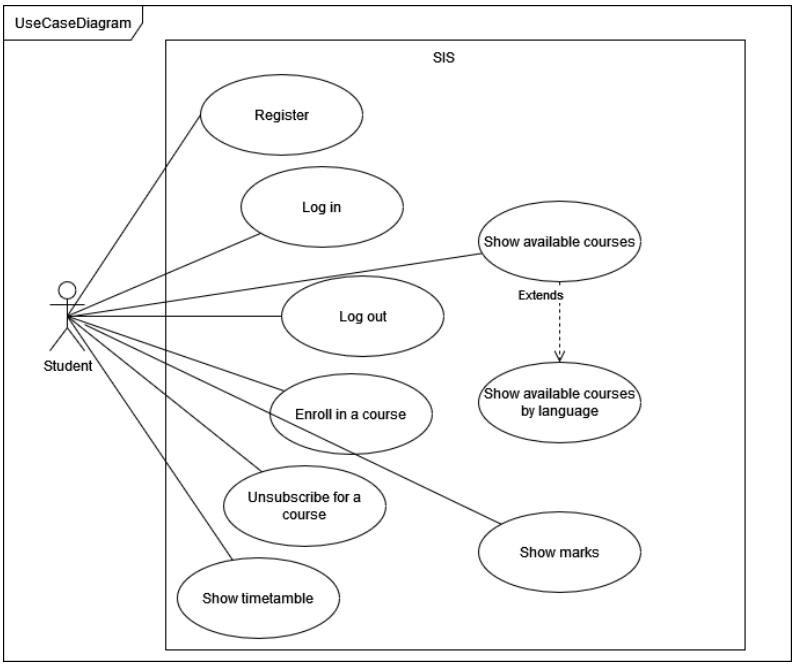
Diagram monolitové aplikace



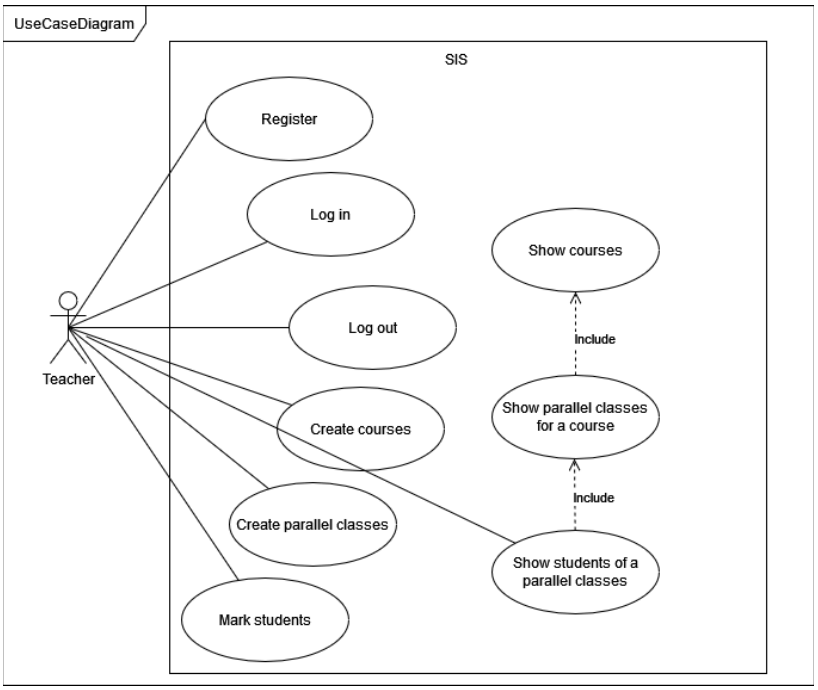
11. Use Case Diagram



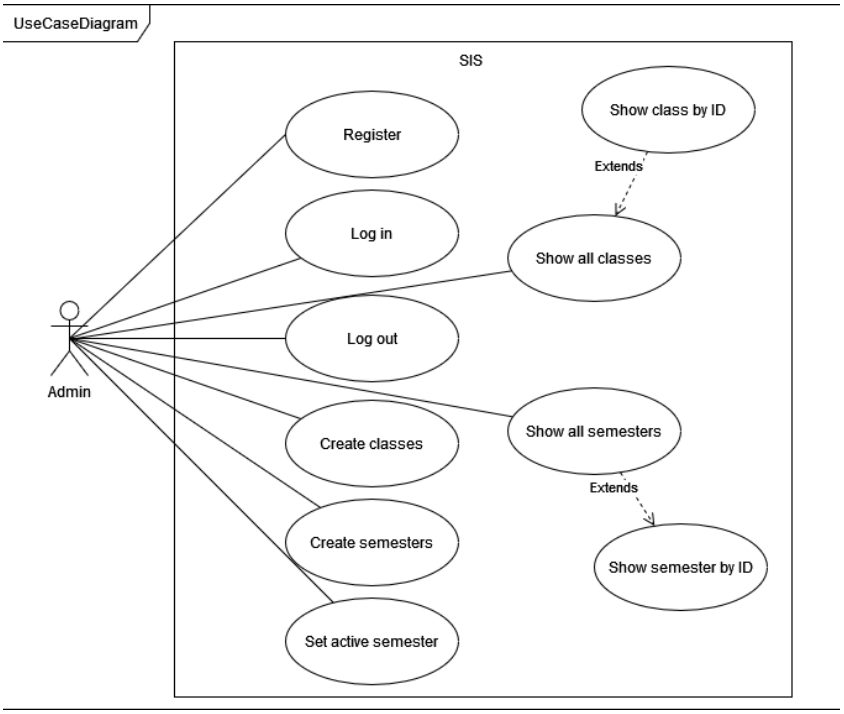
Use Case diagram - Student



Use Case diagram - Učitel

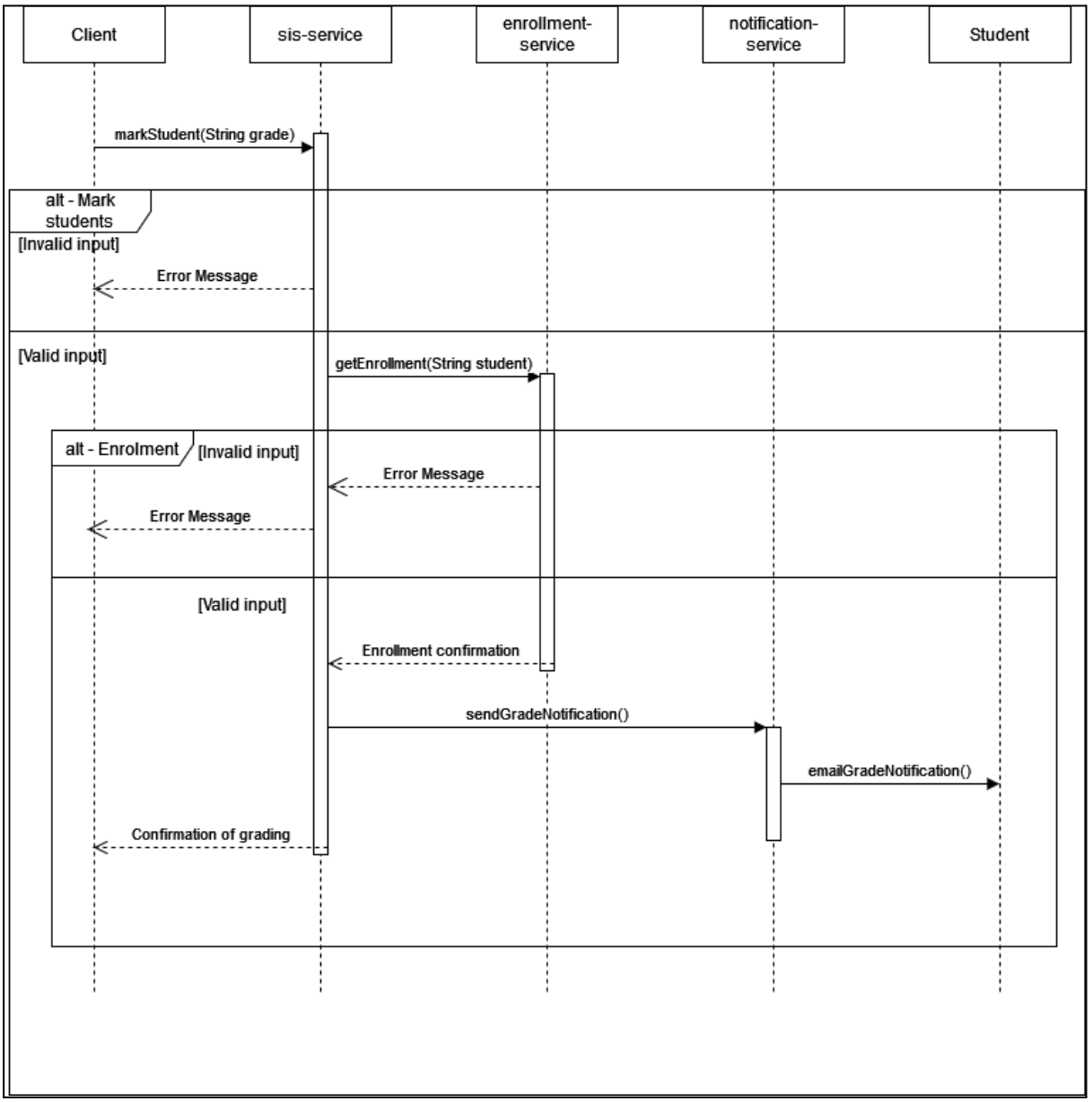


Use Case diagram - Admin



12. Sequence diagram

Oznámkování studenta



13. Výběr vhodné architektury

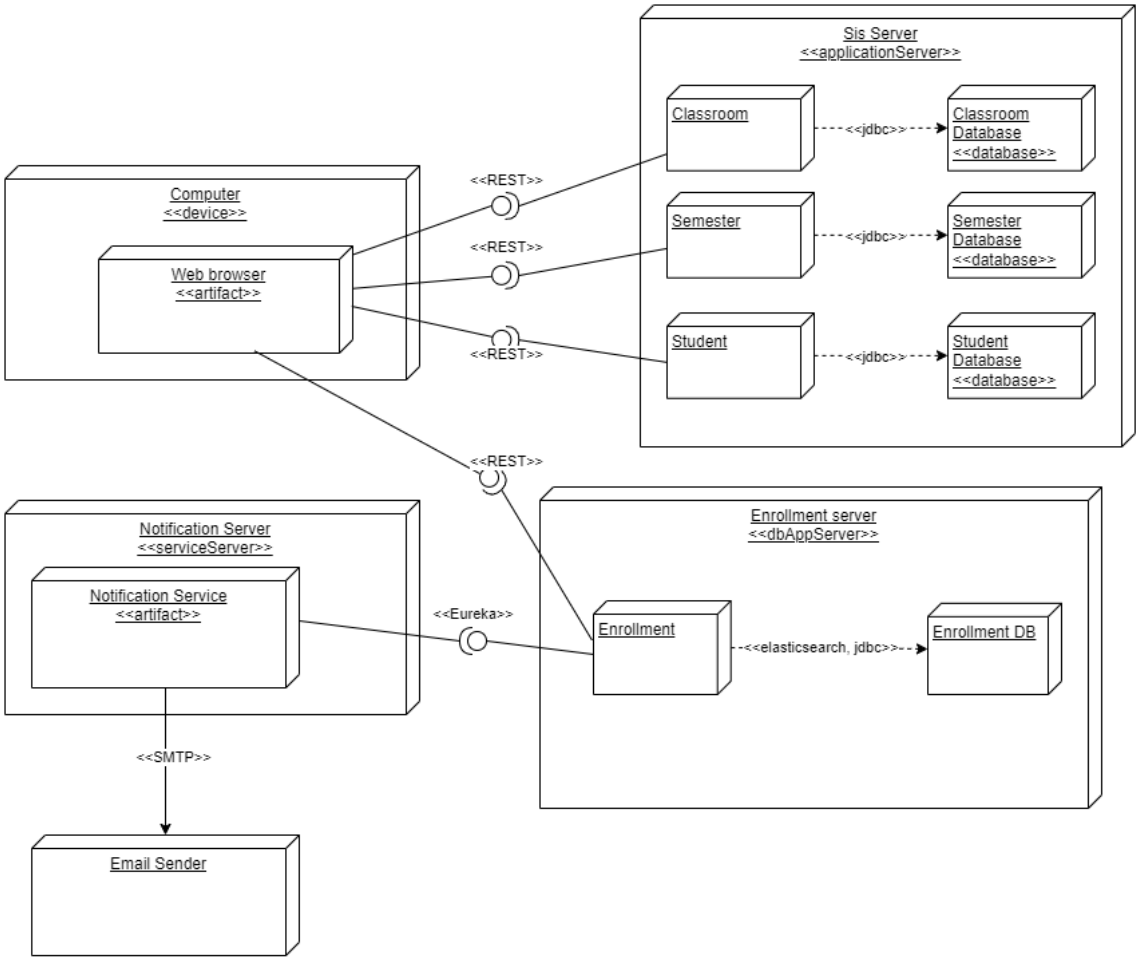
Pro optimalizaci zápisu a čtení známek k danému předmětu je vhodné rozdělit aplikaci na mikroslužby. Konkrétně navrhujeme oddělit entitu Enrollment od zbytku systému, minimálně kvůli očekávanému velkému počtu dotazů během zkouškového období.

Mikroslužba Enrollment bude odpovědná za správu zápisů studentů na kurzy a evidenci jejich výsledků. To zahrnuje operace jako vytváření a aktualizace zápisů, získávání seznamu zápisů pro daný kurz nebo studenta, a další. Pro tuto mikroslužbu navrhujeme následující operace:
Vytvoření zápisu studenta na kurz
Aktualizace zápisu studenta na kurzu (např. změna známky)
Získání seznamu zápisů studentů na daný kurz
Získání seznamu kurzů, do kterých je student zapsán
Získání seznamu zápisů daného studenta

Mikroslužba Enrollment by měla být navržena tak, aby byla odolná vůči velkému počtu dotazů, a zároveň měla vysokou dostupnost a rychlou odezvu. Při návrhu architektury je třeba zvážit, zda oddělit zápis studenta na kurz (Enrollment) od jeho paralelního zápisu (Parallel), nebo zda vytvořit oddělenou mikroslužbu pro zápis studenta na paralelu (Parallel Student). To závisí na požadavcích a potřebách vaší aplikace, ale obecně by oddělení těchto funkcí do samostatných mikroslužeb mohlo přinést větší modularitu a škálovatelnost systému.

ČÁST 2

14. Diagram nasazení



15. Rozbor a výběr alternativ návrhu řešení

Pro návrh našeho systému jsme identifikovali následující alternativy architektonického řešení:

1. **Monolitická architektura**
2. **Mikroslužby**
3. **Serverless architektura**

Hodnocení alternativ:

Kritérium	Monolitická architektura	Mikroslužby	Serverless architektura
Náklady na vývoj	Nízké	Střední	Vysoké
Výkon a škálovatelnost	Omezené	Vysoké	Vysoké
Jednoduchost údržby	Nízká	Vysoká	Střední
Kompatibilita	Vysoká	Střední	Nízká
Bezpečnost	Střední	Vysoká	Vysoká

Výběr řešení:
Na základě hodnocení jsme se rozhodli použít mikroslužby. Tento přístup nabízí nejlepší kompromis mezi náklady, výkonem a jednoduchostí údržby. Mikroslužby poskytují snadnou škálovatelnost a modulární architekturu, což usnadní budoucí rozšiřování systému a jeho údržbu. Výběrem rozdělování komponent také dosáhneme rychlé odezvu aplikace a tudíž jsou pro nás microservisy nejefektivnějším řešením.

16. Zdroje (a použité technologie)

Papers:

- [1] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. disertační práce, Univ. California, Irvine, CA, USA, 2000. Dostupné:https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf.

References:

- [2] "Spring Boot Starter Data REST," verze 2.5.4. Dostupné: https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-data-rest.
- [3] "Spring Boot." Dostupné: https://spring.io/projects/spring-boot.
- [4] "Spring Kafka," verze 2.8.0. Dostupné: https://mvnrepository.com/artifact/org.springframework.kafka/spring-kafka.
- [5] "Spring Cloud Netflix Eureka Client," verze 3.0.4. Dostupné: https://mvnrepository.com/artifact/org.springframework.cloud/spring-cloud-starter-netflix-eureka-client.

- [6] "Spring Boot Starter Data JPA," verze 2.5.4. Dostupné: https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-data-jpa.
- [7] "MapStruct," verze 1.5.5.Final. Dostupné: https://mvnrepository.com/artifact/org.mapstruct/mapstruct/1.5.5.Final.
- [8] "H2 Database," verze 1.4.200. Dostupné: https://mvnrepository.com/artifact/com.h2database/h2.
- [9] "Spring Boot Starter Test," verze 2.5.4. Dostupné: https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-test.
- [10] "JUnit," verze 4.13.2. Dostupné: https://mvnrepository.com/artifact/junit/junit.
- [11] "Tomcat Embed Core," verze 10.1.13. Dostupné: https://mvnrepository.com/artifact/org.apache.tomcat.embed/tomcat-embed-core/10.1.13.
- [12] "Spring Web," verze 6.0.12. Dostupné: https://mvnrepository.com/artifact/org.springframework/spring-web/6.0.12.
- [13] "Spring Security Core," verze 6.1.2. Dostupné: https://mvnrepository.com/artifact/org.springframework.security/spring-security-core/6.1.2.
- [14] "Spring WebMVC," verze 6.0.12. Dostupné: https://mvnrepository.com/artifact/org.springframework/spring-webmvc/6.0.12.
- [15] "Jackson Annotations," verze 2.16.0. Dostupné: https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-annotations/2.16.0.
- [16] "Spring Boot Starter Logging," verze 2.5.4. Dostupné: https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-logging.
- [17] "Jackson Databind," verze 2.16.0. Dostupné: https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind/2.16.0.
- [18] "Spring Security Test," verze 6.1.2. Dostupné: https://mvnrepository.com/artifact/org.springframework.security/spring-security-test/6.1.2.
- [19] "Spring Boot Starter Security," verze 2.5.4. Dostupné: https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-security.
- [20] "Jackson Datatype JSR310," verze 2.13.0. Dostupné: https://mvnrepository.com/artifact/com.fasterxml.jackson.datatype/jackson-datatype-jsr310/2.13.0.
- [21] "Spring Boot Starter Web," verze 2.5.4. Dostupné: https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-web.
- [22] "Spring Boot Starter WebFlux," verze 2.5.4. Dostupné: https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-webflux.
- [23] "Lombok," verze 1.18.20. Dostupné: https://mvnrepository.com/artifact/org.projectlombok/lombok.

- [24] "Spring Boot Starter Actuator," verze 2.5.4. Dostupné: https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-actuator.
- [25] "Micrometer Registry Prometheus," verze 1.7.5. Dostupné: https://mvnrepository.com/artifact/io.micrometer/micrometer-registry-prometheus.
- [26] "Spring Cloud Netflix Eureka Server," verze 3.0.4. Dostupné: https://mvnrepository.com/artifact/org.springframework.cloud/spring-cloud-starter-netflix-eureka-server.
- [27] "Spring Boot Starter Mail," verze 2.5.4. Dostupné: https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-mail.
- [28] "Elasticsearch Rest High Level Client," verze 7.17.21. Dostupné: https://mvnrepository.com/artifact/org.elasticsearch.client/elasticsearch-rest-high-level-client/7.17.21.
- [29] "Spring Boot Starter Data Elasticsearch," verze 2.5.4. Dostupné: https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-data-elasticsearch.
- [30] "Spring Data Elasticsearch," verze 5.3.0. Dostupné: https://mvnrepository.com/artifact/org.springframework.data/spring-data-elasticsearch/5.3.0.

Courses:

- [31] "EAR." Dostupné: https://cw.fel.cvut.cz/b231/courses/b6b36ear/start.
- [32] "NSS." Dostupné: https://cw.fel.cvut.cz/wiki/courses/b6b36nss/start.

17. Normy a standardy

Pro vývoj našeho systému budou dodržovány následující normy a standardy:

1. **ISO/IEC 27001:** Mezinárodní standard pro správu bezpečnosti informací, který zajistí ochranu citlivých dat uživatelů.
2. **ISO/IEC 12207:** Standard pro životní cyklus softwarového vývoje, který nám pomůže při plánování, realizaci a údržbě projektu.
3. **GDPR:** Evropské nařízení o ochraně osobních údajů, které zaručí správné zpracování a ochranu osobních údajů uživatelů.
4. **OWASP Top Ten:** Standard pro bezpečnost webových aplikací, který nám pomůže identifikovat a eliminovat nejčastější bezpečnostní rizika.
5. **Kódovací standardy:** Použití specifických konvencí a postupů při psaní kódu, které zajistí čitelnost a udržovatelnost kódu.

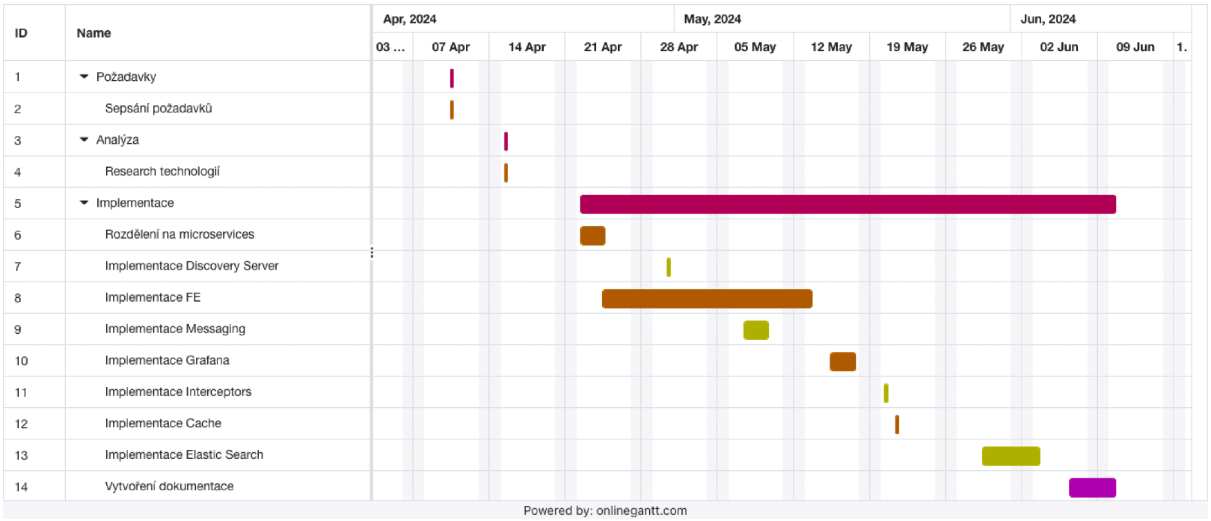
18. Matice zodpovědnosti

Činnost / Role	strobad1	urbanm(z EARek)	bielevan	hoan glon	leduc tha
Elasticsearch	R	-	I	I	A
React	R	-	I	I	A
REST	R	R	-	-	-
Security	R	R	-	-	-
Spring	R	R	-	-	-
Databáze	R	R	-	-	-
Servisy	R	R	-	-	-
Dokumentace	C	-	R	R	I
Diagramy	C	-	R	R	I
Návrhové vzory	C	-	I	R	I
Use Cases	C	-	I	R	I
Kafka	C	-	I	I	R
Cache	C	-	I	I	R
Heroku	R	-	I	I	R
Prometheus	C	-	I	I	R
Interceptor	C	-	I	I	R
Microservices	C	-	I	I	R

Legenda:

- **R** - Responsible (Odpovědný)
- **A** - Accountable (Schvalovatel)
- **C** - Consultant (Konzultant)
- **I** - Informed (Informován)

19. Gantt harmonogram



20. Analýza rizik FMEA

FMEA (Failure Mode and Effects Analysis) Analýza FMEA je systematický přístup k identifikaci a hodnocení potenciálních selhání v systému, jejich příčin a důsledků, a k určení opatření na snížení nebo eliminaci těchto rizik.

Kroky FMEA analýzy:

1. **Identifikace potenciálních selhání:**
    - **Selhání přihlášení uživatele**: Uživatelé nemohou získat přístup do systému.
    - **Ztráta dat při registraci**: Data studentů nebo kurzů nejsou správně uložena.
    - **Chyby při zápisu do kurzu**: Studenti nemohou správně zapsat kurzy nebo se zapsat do nesprávných kurzů.
    - **Nesprávné známkování**: Učitelé nemohou správně zadat nebo změnit známky.
    - **Nedostupnost systému**: Systém je mimo provoz během zkouškového období.
2. **Hodnocení příčin a následků selhání:**
    - **Selhání přihlášení uživatele**: Problémy s autentizací nebo databází -> Uživatelé nejsou schopni vykonávat své úkoly.
    - **Ztráta dat při registraci**: Chyby v kódu nebo problémy s databází -> Neúplná nebo nesprávná data, což může vést k nesprávné administrativě.
    - **Chyby při zápisu do kurzu**: Problémy s logikou aplikace nebo s uživatelským rozhraním -> Studenti se nemohou účastnit požadovaných kurzů.

- **Nesprávné známkování**: Chyby v uživatelském rozhraní nebo v backendu -> Nesprávné akademické výsledky, což může ovlivnit studijní výsledky studentů.
- **Nedostupnost systému**: Technické problémy nebo přetížení serverů -> Uživatelé nemohou přistupovat k systému v klíčových momentech.

3. **Určení opatření ke snížení rizik:**
   - **Selhání přihlášení uživatele**: Implementace robustních autentizačních mechanismů a pravidelné testování.
   - **Ztráta dat při registraci**: Zavedení záložních systémů a pravidelné zálohování dat.
   - **Chyby při zápisu do kurzu**: Důkladné testování uživatelského rozhraní a back-end logiky.
   - **Nesprávné známkování**: Zavedení validací a kontrolních mechanismů pro zadávání známek.
   - **Nedostupnost systému**: Použití vysoce dostupné infrastruktury a škálování serverů během špičkových období.

21. Znovupoužitelnost

Vytvoření softwarových komponent, které lze znovu použít v různých částech aplikace nebo v jiných projektech, zvyšuje efektivitu vývoje a údržby.

Strategie pro znovupoužitelnost:

Modularita:

- Vytvoření jednotlivých modulů pro různé funkce (např. autentizace, správa uživatelů, správa kurzů).
- Každý modul by měl být nezávislý a snadno integrovatelný.

API Design:

- Definování jasných a konzistentních API rozhraní pro komunikaci mezi moduly.
- API by měly být dobře dokumentované a stabilní.

Knihovny a Frameworky:

- Využití existujících knihoven a frameworků pro běžné funkce (např. Spring Boot pro backend, React pro frontend).
- Vyvíjet vlastní knihovny pro specifické potřeby projektu, které lze použít i v jiných projektech.

Documentace:

- Poskytnutí podrobné dokumentace pro každý modul a jeho rozhraní.

- Udržování dokumentace aktuální a snadno přístupné.

22. Metriky

Metriky jsou nezbytné pro měření a hodnocení výkonu systému, kvality kódu a efektivity vývoje.

Typy metrík:

Výkonnostní metriky:

- **Doba odezvy**: Měření doby, kterou systém potřebuje na odpověď na uživatelský požadavek.
- **Propustnost**: Počet požadavků, které systém dokáže zpracovat za jednotku času.
- **Využití zdrojů**: Měření využití CPU, paměti a dalších zdrojů.

Kvalita kódu:

- **Pokrytí testy**: Procento kódu pokryté automatizovanými testy.
- **Počet chyb**: Počet nalezených chyb v kódu během testování nebo v produkčním prostředí.
- **Složitost kódu**: Měření složitosti kódu (např. pomocí metriky Cyclomatic Complexity).

Efektivita vývoje:

- **Doba vývoje**: Čas potřebný k dokončení určité funkce nebo modulu.
- **Počet commitů**: Počet commitů provedených v rámci určitého časového období.
- **Počet vyřešených bugů**: Počet chyb, které byly opraveny v rámci určitého časového období.

23. Plán odbavení

Plán odbavení definuje kroky a postupy, které je třeba provést pro úspěšné nasazení systému do produkčního prostředí.

Kroky plánu odbavení:

1. **Příprava prostředí:**
   - Nastavení serverů a potřebné infrastruktury.
   - Instalace Dockeru a konfigurace Docker prostředí.

2. **Nasazení mikroservis:**
    - Vytvoření Docker image pro jednotlivé mikroservisy (Discovery Server, Enrollment Service, Notification Service, React Frontend, Grafana, Prometheus, Elastic Search).
    - Pushnutí Docker image do registru (např. Docker Hub).
3. **Konfigurace Docker Compose a spuštění:**
    - Vytvoření a konfigurace `docker-compose.yml` souboru pro orchestraci mikroservis.
    - Definování závislostí mezi kontejnery a potřebných síťových nastavení.
    - Spuštění.
4. **Testování po nasazení:**
    - Provedení end-to-end testů pro ověření funkčnosti celého systému.
    - Ověření integrace mezi jednotlivými mikroservisy.
5. **Monitorování a logování:**
    - Nastavení monitorovacích nástrojů (Prometheus, Grafana) pro sledování výkonu systému a jeho zdraví.
    - Konfigurace logování pro všechny mikroservisy a centralizace logů.
6. **Záloha a obnovení:**
    - Zavedení pravidelných záloh dat a konfigurací.
    - Definování postupu pro obnovení systému v případě selhání.
7. **Školení uživatelů:**
    - Poskytnutí školení pro administrátory, učitele a studenty ohledně používání systému.
    - Zajištění podpory pro řešení problémů a dotazů.

24. Plán podpory

Zřízení podpůrného týmu:

- Technická podpora: Řešení technických problémů a údržba systému.
- Uživatelská podpora: Pomoc a odpovědi na dotazy uživatelů.

Komunikační kanály:

- E-mailová a telefonická podpora.
- Online chat.
- FAQ a dokumentace.

Údržba systému:

- Pravidelné aktualizace a bezpečnostní záplaty.
- Monitorování systému (Prometheus, Grafana).

Řešení incidentů:

- Oznamování problémů a rychlá reakce (SLA).
- Záložní strategie pro obnovení dat.

Školení a vzdělávání:

- Uživatelská a technická školení.

Průběžná zpětná vazba:

Sbírání zpětné vazby a implementace zlepšení.