

# **Špecifikácia požiadaviek na akademický informačný systém TUKE**

## **1. Úvod**

Cieľom je vyvinúť akademický informačný systém (AIS) pre Technickú univerzitu v Košiciach. Tento systém má centralizovať všetky kľúčové akademické procesy, zabezpečiť efektívnu komunikáciu medzi študentmi, vyučujúcimi a administratívou a poskytovať transparentný prístup k údajom.

AIS bude fungovať cez webové používateľské rozhranie, pričom dátová vrstva bude tvoriť relačná databáza so štruktúrovaným úložiskom údajov.

---

## **2. Pojmy a vzťahy medzi entitami**

### **2.1 Základné entity**

- Študent**

- Atribúty: meno, priezvisko, rodné číslo/ID, e-mail, ročník, študijný program, absolvované predmety, výsledky skúšok.
- Funkcie: zápis predmetov, žiadosti o potvrdenia, sledovanie známok a rozvrhu.
- Vzťahy: zapisuje si predmety, je hodnotený vyučujúcim, patrí do študijného programu.

- Vyučujúci**

- Atribúty: meno, priezvisko, titul, e-mail, katedra, vyučované predmety.
- Funkcie: správa predmetov, vkladanie známok, generovanie syláb, vedenie skúšok.
- Vzťahy: vyučuje predmety, hodnotí študentov.

- Predmet**

- Atribúty: názov, kód predmetu, kreditová hodnota, typ (povinný/voliteľný), garant, rozvrh.
- Vzťahy: patrí do študijného programu, má vyučujúceho, má priradených študentov.

- Rozvrh**

- Atribúty: čas, miestnosť, predmet, vyučujúci.
  - Funkcie: plánovanie prednášok, cvičení, skúšok.
  - Vzťahy: spája študentov a vyučujúcich s predmetmi.
  - **Študijný program**
    - Atribúty: názov, stupeň (Bc., Ing., PhD.), garant, zoznam predmetov.
    - Vzťahy: zapisujú sa doň študenti, skladá sa z predmetov.
  - **Administrátor**
    - Funkcie: správa používateľov, nastavenie systému, monitoring výkonu.
    - Vzťahy: riadi databázu a prístupové práva.
- 

### **3. Procesy systému**

- **Registrácia a autentifikácia**
  - Podpora univerzitného Single Sign-On (LDAP/SSO).
  - Rôzne úrovne oprávnení (študent, vyučujúci, administrátor).
- **Zápis a správa predmetov**
  - Automatické overovanie splnenia predpokladov.
  - Obmedzenie kapacity predmetov a miestnosti.
- **Tvorba rozvrhu**
  - Algoritmus pre automatické generovanie rozvrhu podľa dostupnosti miestností a vyučujúcich.
  - Manuálne zásahy administrátora.
- **Hodnotenie a skúšky**
  - Vkladanie známok vyučujúcimi.
  - Elektronické prihlásenie na skúšku a automatické uzatváranie známok.
- **Generovanie dokumentov**
  - Výpis známok, potvrdenia o návšteve školy, výkaz o štúdiu.

- Export do PDF s digitálnym podpisom.
  - **Administrácia systému**
    - Pravidelné zálohovanie databázy.
    - Správa používateľských účtov a prístupových práv.
    - Monitoring výkonnosti a logov.
- 

## 4. Odporúčaný proces vývoja

- Metodika: Agile/Scrum – dvojtýždňové sprinty, priebežné doručovanie funkčných modulov.
  - **Etapy:**
    1. Analýza požiadaviek (workshopy, dokumentácia, modelovanie UML).
    2. Návrh architektúry (mikroslužby, REST API, ER diagram databázy).
    3. Implementácia (backend – napr. Java/Spring Boot, frontend – React/Angular, databáza – PostgreSQL).
    4. Testovanie (unit testy, integračné testy, používateľské akceptačné testy).
    5. Nasadenie (Docker/Kubernetes, univerzitný server alebo cloud).
    6. Údržba a podpora (servisné zmluvy, pravidelné aktualizácie).
- 

## 5. Technologický stack a implementačná platforma

### 5.1 Frontend (webové rozhranie)

- **Framework:** React (TypeScript)
- **Stav aplikácie:** Redux Toolkit + RTK Query (fetch + cache)
- **Routovanie:** React Router
- **UI knižnica:** MUI (prístupnosť, theming) + vlastné komponenty
- **Formuláre:** React Hook Form + Zod (validácia)
- **Tabuľky a gridy:** Ag-Grid (filter/paging/export)
- **Medzinárodná lokalizácia:** i18next

- **Grafy/reporty:** Recharts (ľahké grafy) + integrácia Metabase embed
- **Build:** Vite
- **Testy:** Vitest/Jest + Testing Library
- **Doručovanie statík:** CDN/Nginx

**Prečo:** rýchly DX, silný ekosystém, jednoduché testovanie a škálovanie.

---

## 5.2 Backend (API a business logika)

- **Jazyk/Framework:** Java 21 + Spring Boot 3 (REST, Security, Data)
- **API kontrakty:** OpenAPI 3 (Swagger UI), verzovanie /api/v1
- **Autentifikácia/SSO:** OpenID Connect + SAML 2.0 (Keycloak ako IdP, napojenie na univerzitný LDAP)
- **Autorizácia:** RBAC (študent, vyučujúci, administrátor, študijné oddelenie), atribútové pravidlá na úrovni entít
- **Persistencia:** Spring Data JPA (Hibernate) + MapStruct (mapovanie DTO)
- **Validácia:** Bean Validation (Jakarta Validation)
- **Plánovače úloh:** Spring Scheduler (cron, batch uzávierky)
- **Asynchrónne spracovanie:** RabbitMQ (fronty na generovanie dokumentov, notifikácie)
- **Vyhľadávanie:** OpenSearch/Elasticsearch (fulltext nad predmetmi, dokumentmi)
- **Cache:** Redis (sessiony, často čítané číselníky)
- **Súbory/dokumenty:** S3 kompatibilné úložisko (MinIO on-prem alebo S3 v cloude)
- **Generovanie PDF:** wkhtmltopdf alebo Puppeteer (server-side render)
- **Notifikácie:** e-mail (SMTP univerzity), voliteľne push (Web Push)
- **Reporty/BI:** Metabase (ad-hoc dotazy, dashboardy)
- **Testy:** JUnit 5, Testcontainers (integračné testy s reálnymi DB/brokerom)

**Prečo:** Spring Boot je štandard v enterprise, vie pekne OIDC/SAML, transakcie, a má stabilné dlhodobé LTS.

---

### 5.3 Databáza a dátá

- **Relačná databáza:** PostgreSQL 16 (ACID, replikácia)
  - **Model:** 3NF + číselníky; audit stĺpce (created\_at, updated\_at, created\_by, updated\_by)
  - **Migračný nástroj:** Flyway (verzovanie schémy)
  - **Šifrovanie:** TDE/OS-level + šifrovanie citlivých polí (PG crypto/ext. KMS)
  - **Zálohy:** pgBackRest (plné + inkrementálne, test obnovy 1× mesačne)
  - **Maskovanie dát:** anonymizácia dumpov pre test/dev
- 

### 5.4 Integrácie

- **LDAP/AD:** synchronizácia identít (študenti, zamestnanci)
  - **E-learning:** Moodle (REST webservice, enrolment sync)
  - **Knižnica:** napojenie cez API/stanovený protokol (napr. Aleph/Koha)
  - **Ekonomika/študijná agenda:** exporty/importy (CSV/XML/JSON), message bus ak dostupný
  - **Elektronický podpis:** kvalifikovaný podpis pečate inštitúcie (QSCD/HSM, časová pečiatka)
  - **E-mail:** univerzitný SMTP s DKIM/SPF
- 

### 5.5 DevOps, infra a prevádzka

- **Kontajnerizácia:** Docker
- **Orchestrácia:** Kubernetes (on-prem k8s/RKE2 alebo cloud AKS/GKE/EKS)
- **CI/CD:** GitLab CI alebo GitHub Actions (build, test, SCA, SAST, DAST, deploy)
- **IaC:** Terraform (infra), Ansible (konfigurácie)
- **Ingress/SSL:** Nginx Ingress + Let's Encrypt/ACME alebo interná CA
- **Monitoring:** Prometheus + Grafana (APM: OpenTelemetry, Grafana Tempo/Jaeger)

- **Logy:** Loki/ELK stack
  - **Tajomstvá:** HashiCorp Vault alebo K8s Secrets + KMS
  - **Zotavenie:** multi-AZ uzly, zálohy DB/S3, DR runbook
- 

## 5.6 Bezpečnosť a compliance

- **SSO/IdP:** Keycloak (OIDC/SAML), MFA pre administrátorov
  - **Politiky prístupu:** least-privilege, audit log všetkých kritických akcií
  - **GDPR:** minimalizácia dát, retenčné lehoty, právo na prístup/mazanie, DPIA
  - **Testovanie bezpečnosti:** SAST/DAST v CI, pravidelné penetračné testy, SBOM (CycloneDX)
- 

## 5.7 Platformové scenáre nasadenia

### A) On-prem (preferované pre univerzitné prostredie)

- **Platforma:** Kubernetes na univerzitných serveroch (VMware/Proxmox + RKE2)
- **Úložiská:** Ceph/Rook pre blokové a objektové storage, MinIO pre S3
- **Výhody:** kontrola nad dátami, nižšie variabilné náklady, jednoduchšia integrácia s internými systémami
- **Nevýhody:** vyššie nároky na správu a monitoring

### B) Cloud (ak univerzita chce outsourcovať infra)

- **Platforma:** Azure AKS alebo Google GKE
  - **Úložiská:** Managed PostgreSQL (Azure Database for PostgreSQL/Cloud SQL), Managed Elasticsearch/OpenSearch
  - **Výhody:** rýchle škálovanie, menej infra starostí
  - **Nevýhody:** mesačné náklady, dátová suverenita (treba právne ošetriť)
- 

## 5.8 Minimálna referenčná konfigurácia (pilot)

- **K8s klaster:** 3× worker (8 vCPU, 32 GB RAM), 1× control-plane (shared)

- **DB:** PostgreSQL VM (8 vCPU, 32 GB RAM, SSD 1 TB, replikácia na standby)
  - **S3/objektové úložisko:** 2–4 TB (závisí od počtu generovaných PDF/dokumentov)
  - **OpenSearch:** 3-uzlový klaster (4 vCPU, 16 GB RAM na uzol)
  - **Redis/RabbitMQ:** malé HA clustre (2–3 uzly)
- 

## 5.9 Vývojové prostredie a kvalita

- **Branch stratégia:** trunk-based alebo GitFlow (podľa preferencie tímu)
  - **Kvalita kódu:** SonarQube (coverage, code smells, security hotspots)
  - **Konvencie:** pre backend Checkstyle/Spotless, pre frontend ESLint/Prettier
  - **Release management:** semver, release notes z konv. commitov
  - **Test dát:** anonymizované z produkcie, automatizované seedovanie
- 

## 5.10 Alternatívy (ak by bolo treba)

- **Frontend:** Angular (TypeScript) namiesto Reactu
- **Backend:** .NET 8 (ASP.NET Core) namiesto Spring Boot
- **Broker:** Kafka (ak sú požiadavky na vysoký throughput eventov)
- **Reporty:** Superset namiesto Metabase

## 6. Zloženie tímu

- Projektový manažér – riadenie časového plánu, komunikácia s univerzitou.
- Business analytik – analýza požiadaviek, UML modely.
- UX/UI dizajnér – návrh responzívneho webového rozhrania.
- Backend vývojári (2–3) – implementácia API a logiky systému.
- Frontend vývojári (2) – tvorba moderného webového rozhrania.
- Databázový špecialista – návrh databázy, optimalizácia dotazov.
- Tester/QA – príprava testovacích scenárov, automatizované testy.

- DevOps inžinier – správa prostredia, CI/CD pipeline, bezpečnosť.
- 

## 7. Odhad ceny

Cena závisí od rozsahu funkcionálít a od toho, či pôjde o prototyp, alebo plnohodnotný systém pre celú univerzitu.

### Varianty:

- **Základná verzia (jadro systému)**
    - Obsahuje: registrácia, zápis predmetov, rozvrh, hodnotenie, základné dokumenty.
    - Cena: 120 000 – 150 000 €
    - Vhodné ako pilotný projekt.
  - **Rozšírená verzia**
    - Obsahuje: všetky základné funkcie + generovanie potvrdení, skúškové prihlásovanie, integrácia s LDAP/e-mailom, reporting.
    - Cena: 180 000 – 220 000 €
    - Realistické riešenie pre univerzitu.
  - **Kompletný AIS**
    - Obsahuje: všetky predchádzajúce moduly + mobilná aplikácia, analytické nástroje, integrácia s Moodle, knižničným systémom, Erasmus modulom.
    - Cena: 250 000 – 300 000 €
    - Úplné pokrytie potrieb univerzity s dlhodobým využitím.
- 

## 8. Časový rámec

- Analýza a návrh – 2–3 mesiace.
- Implementácia jadra (zápis, rozvrh, hodnotenie) – 4–6 mesiacov.
- Testovanie a pilotná prevádzka – 2–3 mesiace.
- Plné nasadenie a školenie používateľov – 1 mesiac.

Celkový čas: 10–13 mesiacov.

---

## **9. Záver**

Navrhovaný systém by umožnil TUKE prejsť na moderný, digitálne orientovaný model riadenia akademických procesov. Dokument obsahuje definíciu entít a procesov, návrh metodiky vývoja, odporúčané zloženie tímu, finančný a časový odhad. Je vhodný ako podklad pre detailný projektový plán a výber technológií.