Architektonický štýl

- **Modulárny monolit (DDD-lite)** → jasné bounded contexts, shared jadro; neskôr deliteľné na mikroslužby.
- **Vrstvy**: API (REST) → Application (orchestration, use-cases) → Domain (entity/aggregates + policies) → Infrastructure (DB, MQ, S3, IdP).
- **Komunikácia**: sync REST vnútri monolitu, **domain events** + **outbox** pre asynchrónne vedľajšie efekty (e-maily, indexácia, exporty).

Bounded Contexts (moduly)

- identity (SSO/OIDC, RBAC, sync s LDAP/Keycloak).
- study (študijné programy, predmety, prerekvizity, zápisy).
- schedule (rozvrh, miestnosti, auto-scheduler + manuálne úpravy).
- exam (termíny, prihlášky, uzávierky, známky).
- document (žiadosti, render PDF, podpis/pečať, S3).
- adminops (audit, číselníky, systémové nastavenia, monitoring).
- integration (Moodle, knižnica, ekonomika, e-mail, vyhľadávanie).

API dizajn (REST, versioned /api/v1, OpenAPI 3)

Auth

- GET /auth/me → profil z tokenu (Keycloak)
- OIDC flow (Authorization Code), refresh, scopes.

Študent

- GET /students/{id} (self alebo admin)
- GET /students/{id}/enrollments
- POST /students/{id}/enrollments (validuje prerekvizity, kapacitu)
- GET /students/{id}/grades

- POST /students/{id}/document-requests (typ dokumentu)

Predmety & programy

- GET /programs
- GET /courses?query=&programId=&type=&page=
- GET /courses/{id}
- GET /courses/{id}/timetable
- GET /courses/{id}/exam-terms

Učiteľ

- GET /teachers/{id}/courses
- POST /courses/{id}/syllabus (update)
- POST /courses/{id}/grades (bulk grade upsert)
- POST /courses/{id}/exam-terms
- PATCH /exam-terms/{id}

Skúšky

- POST /exam-terms/{id}/registrations (študent)
- DELETE /exam-terms/{id}/registrations/{regId}
- POST /exam-terms/{id}/close (uzávierka → batch do grade + notifikácie)

Rozvrh

- POST /scheduler/auto-generate?semester=WS2025 (admin)
- GET /scheduler/proposals/{jobId} (stav)
- POST /timetable/slots (manuálny zásah)

Dokumenty

- GET /documents/{requestId}/download (S3 pre-signed URL)
- POST /documents/{requestId}/sign (študijné → HSM/QSCD)
- POST /documents/{requestId}/notify (re-send e-mail)

Admin

- GET /admin/users?query=
- POST /admin/users
- PATCH /admin/users/{id}
- GET /admin/audit?from=&to=&actor=

Dizajn databázy (PostgreSQL 16)

Core tabuľky (kľúčové stĺpce + constraints)

1) Identity / Users

- ais.user_account(id, keycloak_id UNIQUE, email UNIQUE, given_name, family_name, roles[], …) – identity väzba na IdP.
- ais.student(id PK, user_id FK UNIQUE, program_id FK, year_of_study, student_no UNIQUE, status)
- ais.teacher(id PK, user_id FK UNIQUE, department, title)

Indexy: user_account(email) uniq, student(student_no) uniq, student(program_id), teacher(department).

2) Štúdium / Predmety

- ais.program(id, name, level, guarantor_teacher_id FK)
- ais.course(id, code UNIQUE, name, credits, type_id FK, program_id FK, capacity, lecturer_id FK, semester_code)
- ais.course_prereq(course_id FK, required_course_id FK, PRIMARY KEY(course_id, required_course_id))

Indexy: course(program_id), course(semester_code), course(lecturer_id), course_prereq(required_course_id).

3) Zápisy

- ais.enrollment(id, student_id FK, course_id FK, state_id FK, created_at, UNIQUE(student_id, course_id))
  Stav: lookup support.enrollment_state(id, code) = REQUESTED/APPROVED/REJECTED/DROPPED.

Indexy: enrollment(student_id), enrollment(course_id), enrollment(state_id).

4) Rozvrh

- ais.timetable_slot(id, course_id FK, teacher_id FK, room, day_of_week, start_time, end_time, semester_code)
  Partícia: PARTITION BY LIST (semester_code) → timetable_slot_2025WS, timetable_slot_2026SS, …

Indexy: (semester_code, day_of_week, start_time), (teacher_id, semester_code), (room, semester_code, day_of_week, start_time).

5) Skúšky & registrácie

- ais.exam_term(id, course_id FK, date_time, room, capacity, semester_code) (tiež partície podľa semester_code)
- ais.exam_registration(id, exam_term_id FK, student_id FK, state_id FK, UNIQUE(exam_term_id, student_id))
  Stav: lookup support.exam_reg_state = REGISTERED/CANCELLED/GRADED.

Indexy: exam_term(course_id, date_time), exam_registration(student_id), exam_registration(state_id).

6) Známky

- ais.grade(id, student_id FK, course_id FK, examiner_id FK, value_id FK, is_final BOOL, graded_at)
  Lookup support.grade_value = A..Fx. **Partícia** podľa semester_code (prenesené z kurzu).

Unique: (student_id, course_id, is_final=true) – max jeden final.
Indexy: grade(student_id), grade(course_id), grade(graded_at).

7) Dokumenty

- ais.document_request(id, student_id FK, type_id FK, state_id FK, file_key, signed BOOL, created_at, ready_at)
  Lookups: support.document_type (potvrdenie, výpis, výkaz), support.doc_state (PENDING/PROCESSING/READY).

Indexy: document_request(student_id, created_at DESC), document_request(state_id).

8) Audit & Outbox

- audit.event(id, actor_user_id, action, entity, entity_id, payload_jsonb, at) – bezpečnostný denník.

- support.event_outbox(id, aggregate_type, aggregate_id, event_type, payload_jsonb, created_at, processed_at, status) – transactional outbox.

Indexy: event_outbox(status, created_at) pre FIFO publish.

DDL

```
-- SCHEMAS
create schema if not exists ais;
create schema if not exists support;
create schema if not exists audit;

-- LOOKUPS (enums as tables)
create table support.enrollment_state (
  id smallserial primary key,
  code text not null unique -- REQUESTED, APPROVED, REJECTED, DROPPED
);

create table support.grade_value (
  id smallserial primary key,
  code text not null unique -- A,B,C,D,E,Fx
);

-- USERS
create table ais.user_account (
  id uuid primary key,
  keycloak_id text unique not null,
  email text unique not null,
  given_name text not null,
  family_name text not null,
  roles text[] not null default '{}',
  created_at timestamptz not null default now(),
  updated_at timestamptz not null default now()
);

create table ais.program (
  id uuid primary key,
  name text not null,
  level text not null, -- Bc., Ing., PhD.
  guarantor_teacher_id uuid,
  created_at timestamptz not null default now(),
  updated_at timestamptz not null default now()
);

create table ais.teacher (
  id uuid primary key,
  user_id uuid not null unique references ais.user_account(id),
  department text,
  title text,
  created_at timestamptz not null default now(),
  updated_at timestamptz not null default now()
);

create table ais.student (
  id uuid primary key,
```

```
  user_id uuid not null unique references ais.user_account(id),
  program_id uuid not null references ais.program(id),
  year_of_study int not null check (year_of_study between 1 and 8),
  student_no text not null unique,
  status text not null default 'ACTIVE',
  created_at timestamptz not null default now(),
  updated_at timestamptz not null default now()
);

create table ais.course_type (
  id smallserial primary key,
  code text not null unique -- POVINNY, VOLITELNY
);

create table ais.course (
  id uuid primary key,
  code text not null unique,
  name text not null,
  credits int not null check (credits between 0 and 60),
  type_id smallint not null references ais.course_type(id),
  program_id uuid not null references ais.program(id),
  capacity int not null check (capacity >= 0),
  lecturer_id uuid references ais.teacher(id),
  semester_code text not null, -- e.g. 2025WS
  created_at timestamptz not null default now(),
  updated_at timestamptz not null default now()
);

create table ais.course_prereq (
  course_id uuid not null references ais.course(id) on delete cascade,
  required_course_id uuid not null references ais.course(id),
  primary key (course_id, required_course_id),
  check (course_id <> required_course_id)
);

create table ais.enrollment (
  id uuid primary key,
  student_id uuid not null references ais.student(id),
  course_id uuid not null references ais.course(id),
  state_id smallint not null references support.enrollment_state(id),
  created_at timestamptz not null default now(),
  unique (student_id, course_id)
);

-- PARTITIONED tables for timetable & exams
create table ais.timetable_slot (
  id uuid primary key,
  course_id uuid not null references ais.course(id),
  teacher_id uuid not null references ais.teacher(id),
  room text not null,
  day_of_week smallint not null check (day_of_week between 1 and 7),
  start_time time not null,
  end_time time not null check (end_time > start_time),
  semester_code text not null
) partition by list (semester_code);

-- Example partition
```

```
create table ais.timetable_slot_2025WS partition of ais.timetable_slot for
values in ('2025WS');

create table ais.exam_term (
  id uuid primary key,
  course_id uuid not null references ais.course(id),
  date_time timestamptz not null,
  room text not null,
  capacity int not null check (capacity >= 0),
  semester_code text not null
) partition by list (semester_code);

create table ais.exam_registration (
  id uuid primary key,
  exam_term_id uuid not null references ais.exam_term(id),
  student_id uuid not null references ais.student(id),
  state_id smallint not null references support.enrollment_state(id),
  created_at timestamptz not null default now(),
  unique (exam_term_id, student_id)
);

create table ais.grade (
  id uuid primary key,
  student_id uuid not null references ais.student(id),
  course_id uuid not null references ais.course(id),
  examiner_id uuid not null references ais.teacher(id),
  value_id smallint not null references support.grade_value(id),
  is_final boolean not null default false,
  graded_at timestamptz,
  semester_code text not null,
  unique (student_id, course_id, is_final)
) partition by list (semester_code);

create table ais.document_type (
  id smallserial primary key,
  code text not null unique -- POF_STUDIA, VYPIS_ZNAMOK, VYKAZ
);

create table support.doc_state (
  id smallserial primary key,
  code text not null unique -- PENDING, PROCESSING, READY
);

create table ais.document_request (
  id uuid primary key,
  student_id uuid not null references ais.student(id),
  type_id smallint not null references ais.document_type(id),
  state_id smallint not null references support.doc_state(id),
  file_key text,        -- S3 key when ready
  signed boolean not null default false,
  created_at timestamptz not null default now(),
  ready_at timestamptz
);

-- AUDIT & OUTBOX
create table audit.event (
  id uuid primary key,
```

```
  actor_user_id uuid,
  action text not null,
  entity text not null,
  entity_id uuid,
  payload_jsonb jsonb,
  at timestamptz not null default now()
);

create table support.event_outbox (
  id uuid primary key,
  aggregate_type text not null,
  aggregate_id uuid not null,
  event_type text not null,
  payload_jsonb jsonb not null,
  status text not null default 'PENDING',
  created_at timestamptz not null default now(),
  processed_at timestamptz
);

-- Helpful indexes
create index on ais.course (program_id);
create index on ais.course (semester_code);
create index on ais.enrollment (student_id);
create index on ais.enrollment (course_id);
create index on ais.exam_term (course_id, date_time);
create index on ais.exam_registration (student_id);
create index on ais.grade (student_id);
create index on ais.grade (course_id);
create index on support.event_outbox (status, created_at);
```