ERP/1: Освіта

Технороботчий проект

на створення та впровадження модуля
«Освіта»
як засобу інформатизації

Згідно з Постановою КМУ № 205 від 21.02.2025

Формалізація вимог за стандартом ISO/IEC/IEEE 29148:2018

Зміст

Зміст

1. Загальні відомості про засіб інформатизації

1.1. Найменування та підстави для розробки

Найменування засобу інформатизації: Модуль «Освіта» інформаційної системи управління підприємством ERP/1 (далі — Модуль «Освіта»).
Відомості про замовника: ТОВ "Системи Електронного Урядування"в ТОВ "Криптографічні телесистеми"..
Відомості про виконавця робіт: Визначається замовником на конкурсній основі.
Підстави для виконання робіт:

- Закон України «Про освіту» (із змінами та доповненнями);
- Закон України «Про захист персональних даних»;
- Постанова Кабінету Міністрів України від 21 лютого 2025 року № 205 «Деякі питання створення, адміністрування та забезпечення функціонування засобу інформатизації»;
- Програма інформатизації ERP/1.

1.2. Призначення та цілі створення засобу

Модуль «Освіта» призначений для автоматизації та управління освітніми процесами у закладах освіти різного рівня (ЗСО, ПО, ФПО, ВО).
Цілі створення засобу:

- Створення сучасного середовища ведення освітньої документації в електронному вигляді з накладанням КЕП;
- Автоматизація розробки та затвердження навчальних планів і робочих програм;
- Ведення електронного журналу успішності та відвідуваності в режимі реального часу;
- Контроль виконання бізнес-правил освітнього процесу та валідація оцінок;
- Автоматизоване планування розкладу занять на основі ресурсних обмежень;

- Забезпечення інтеграції з Єдиною державною електронною базою з питань освіти (ЄДЕБО).

1.3. План-графік виконання етапів робіт

Роботи з розробки та впровадження Модуля «Освіта» розбиваються на такі етапи:

1. **Етап 1: Технічне проектування** — погодження UX/UI прототипів у Figma, уточнення структури даних Mnesia. (Тривалість: 1 місяць).
2. **Етап 2: Розробка інформаційного забезпечення та схем** — опис та реалізація Erlang-рекордів, створення базових таблиць Mnesia. (Тривалість: 1.5 місяці).
3. **Етап 3: Реалізація бізнес-процесів** — написання BPE DSL та інтеграція з рушієм процесів. (Тривалість: 2 місяці).
4. **Етап 4: Інтеграція з КЕП та ЄДЕБО** — реалізація криптографічних підписів та API обміну. (Тривалість: 1 місяць).
5. **Етап 5: Комплексне тестування** — автотестування (Unit, Integration), UAT приймання. (Тривалість: 1 місяць).
6. **Етап 6: Впровадження та дослідна експлуатація**. (Тривалість: 1.5 місяці).

2. Відомості про робочий процес та умови експлуатації

2.1. Опис бізнес-процесів (BPMN / FSM)

Освітні бізнес-процеси в Модулі «Освіта» реалізуються за допомогою скінченних автоматів (FSM) та інтегруються з рушієм процесів BPE:

2.1.1. Процес оцінювання (Student Assessment FSM)

Процес охоплює виставлення поточних та підсумкових балів, розрахунок компетенцій та затвердження КЕП:

1. **StartAssessment:** Початок періоду оцінювання (початок уроку або семестрової атестації).
2. **EnterGrades:** Педагог вводить оцінки у систему.
3. **ValidateGrades:** Автоматична перевірка введених оцінок за допомогою валідатора `grade_validator.erl` на відповідність порогам. При успішній перевірці перехід до `CalculateCompetences`, інакше — повернення до `EnterGrades`.
4. **CalculateCompetences:** Автоматичний розрахунок середнього балу та інтегральних показників компетенцій за темою/семестром.
5. **TeacherSign:** Педагог підписує журнал КЕП. Якщо оцінки містять незадовільні балі (< 4) при підсумковій атестації, процес вимагає переходу до стану `ViceCheck`.
6. **ViceCheck:** Завуч/методист перевіряє та погоджує відомість.
7. **AssessmentSigned:** Завершення процесу, фіксація записів.

2.1.2. Процес проведення занять (Lesson Execution)

1. **LessonStart:** Початок заняття за затвердженим розкладом.
2. **TakeAttendance:** Фіксація відсутності учнів (Present/Absent).

3. **AssessStudents:** Виставлення оцінок за роботу на уроці та домашні завдання.
4. **CalculateAverages:** Розрахунок поточної середньої успішності.
5. **SignJournal:** Накладання КЕП на журнал протягом 48-годинного вікна.
6. **LessonClosed:** Фіксація журналу, архівування.

2.1.3. Затвердження навчального плану (Curriculum Approval)

1. **Start:** Створення проекту навчального плану методистом.
2. **CreatePlan:** Наповнення плану дисциплінами, годинами та кредитами.
3. **MethodistReview:** Перевірка плану на відповідність лімітам освітньої програми.
4. **ViceApprove:** Погодження плану проректором/завучем.
5. **DirectorApprove:** Остаточне затвердження КЕП директора закладу.
6. **GenerateSchedule:** Перехід до автоматичного планування розкладу занять.
7. **PlanApproved:** Активація плану.

2.1.4. Генерація розкладу занять (Schedule Generation)

1. **PlanApproved:** Запуск генерації після затвердження плану.
2. **LoadPlanAndConstraints:** Завантаження плану та ресурсних обмежень (кількість аудиторій, доступність викладачів).
3. **GenerateSchedule:** Робота евристичного алгоритму генерації. При конфліктах — перехід до `ManualAdjustment`, при успіху — до `ViceDirectorApprove`.
4. **ManualAdjustment:** Ручне редагування розкладу диспетчером.
5. **ViceDirectorApprove:** Затвердження розкладу керівництвом закладу.
6. **PublishAndNotify:** Публікація та розсилка повідомлень про розклад.

2.2. Опис ролей та прав доступу (ABAC)

Модель контролю доступу базується на атрибутах (ABAC) та перевіряється на рівні ядра авторизації.

Матриця базового доступу:

- **Педагог (Teacher):** Доступ `read`, `write`, `sign` для журналів (`JournalRecord`), якщо він є викладачем цього курсу або курс входить до його списку дисциплін. Редагування обмежено 48 годинами з моменту заняття.
- **Учень (Student):** Доступ `read` тільки до власних оцінок. Заборонено `write`, `sign`, `delete`.
- **Батько/Мати (Parent):** Доступ `read` для оцінок дитини (згідно зі списком зв'язків).
- **Адміністратор/Завуч (Admin/Vice):** Перегляд усіх документів, затвердження відомостей, редагування планів та довідників.

Формальні логічні правила ABAC:

- **Правило 1 (Педагог):**

$$\text{subject.roles contains 'Teacher'} \land \text{object.entity\_type} == \text{'JournalRecord'}$$

$$\land (\text{object.teacher\_id} == \text{subject.person\_id} \lor \text{object.course\_id in subject.courses\_taught})$$

$$\rightarrow \text{Permit: read, write, sign.}$$

- **Правило 2 (Учень):**

$$\text{subject.roles contains 'Student'} \land \text{object.entity\_type} == \text{'JournalRecord'} \land \text{object.student\_id} == \text{subject.pers}$$

$$\rightarrow \text{Permit: read (тільки власні оцінки). Deny: write, sign.}$$

- **Правило 3 (Батько):**

$$\text{subject.roles contains 'Parent'} \land \text{object.student\_id in subject.children\_ids}$$

$$\rightarrow \text{Permit: read. Deny: write, sign.}$$

- **Правило 4 (КЕП):**

$$\text{subject.kep\_cert\_thumbprint is valid} \land \text{subject.roles contains 'Teacher'} \land \text{object.status} == \text{'draft'}$$

$$\rightarrow \text{Permit: sign.}$$

- **Правило 5 (Час редагування):**

$$\text{subject.roles contains 'Teacher'} \land \text{env.time} \leq \text{lesson.start\_time} + 48 \text{ годин} \land \text{object.status} == \text{'draft'}$$

$$\rightarrow \text{Permit: write.}$$

2.3. Умови експлуатації та системне середовище

Модуль «Освіта» розробляється як хмарна веб-платформа.
Системне середовище:

- **Середовище виконання:** Erlang/OTP версії 26 або вище.
- **База даних:** Вбудована СУБД Erlang Mnesia. Дозволяється використання будь-яких сторонніх SQL СУБД (PostgreSQL, MySQL, SQLite) для підвищення продуктивності та відмовостійкості, реплікації та аналітики.
- **Веб-сервер:** Cowboy (вбудований в Erlang-додаток).
- **Фронтенд-платформа:** Веб-браузери з підтримкою HTML5, CSS3, JavaScript та Secure WebSockets.

3. Інформаційне та програмне забезпечення

3.1. Інформаційне забезпечення (Схеми та Дані)

3.1.1. Визначення структур даних (Erlang Records)

Вся інформаційна модель бази даних описується у вигляді рекорді Erlang:

```
-record(curriculum, {
    id,                  % UUID (Primary Key)
    program_id,          % UUID (FK -> EducationalProgram)
    group_id,            % UUID (FK -> Group)
    academic_year,       % Integer (навчальний рік)
    semester,            % Integer (1-2)
    total_credits,       % Decimal (загальна кількість кредитів)
    total_hours,         % Integer (загальна кількість годин)
    approved_by,         % UUID (FK -> Teacher, КЕП підпис)
    approved_at,         % Timestamp
    status = draft,      % draft | pending | approved | archive
    form_of_study_id,    % UUID (FK -> FormOfStudy)
    language,            % Enum (ua, en)
    is_active = true,    % Boolean
    metadata = #{}       % Map для додаткових неструктурованих даних
}).

-record(curriculum_item, {
    id,                  % UUID (PK)
    curriculum_id,       % UUID (FK -> Curriculum)
    course_id,           % UUID (FK -> Course)
    semester,            % Integer
    hours_lectures,      % Integer
    hours_practice,      % Integer
    hours_lab,           % Integer
    hours_self,          % Integer
    credits_ects,        % Decimal
    grade_type_id,       % UUID (FK -> GradeType)
    teacher_id,          % UUID (FK -> Teacher)
    order,               % Integer
    is_mandatory = true  % Boolean
}).
```

```
-record(form_of_study, {
    id,                  % UUID (PK)
    code,                % String (FULL, DIST, MIX)
    name_ua,             % String (українська назва)
    name_en,             % String (англійська назва)
    description,         % Text
    is_active = true,    % Boolean
    order,               % Integer
    supports_mixed = false,
    requires_online_platform = false,
    legal_basis,
    metadata = #{}
}).

-record(grade_type, {
    id,                  % UUID (PK)
    code,                % String
    name_ua,             % String
    name_en,             % String
    scale_type,          % numeric_12 | ects | competence | score_100
    min_value,           % Decimal
    max_value,           % Decimal
    passing_threshold,   % Decimal
    weight = 0.0,        % Decimal (0.0 - 1.0)
    is_final = false,    % Boolean
    requires_approval = false,
    metadata = #{}
}).

-record(educational_program_type, {
    id,
    code,
    name_ua,
    name_en,
    education_level,
    program_category,
    duration_years_min,
    duration_years_max,
    ects_min,
    description,
    is_active = true,
    metadata = #{}
}).

-record(educational_program, {
```

```
    id, code, name_ua, name_en, level, specialty_code, specialty_name,
    qualification, duration_years, credits_ects, form_of_study_id,
    language, accreditation_status, accreditation_until,
    license_number, department_id, is_active = true, metadata = #{}
}).
```

3.1.2. Опис довідників та реєстрових сутностей

Словник: Типи освітніх програм (EducationalProgramType)

Реквізит	Тип даних	Обов'язковий	Опис
id	UUID	Так	Унікальний ідентифікатор
code	String (30)	Так	Код типу програми (GENERAL_SECONDARY тощо)
name_ua	String (300)	Так	Назва українською
duration_years_min	Decimal	Так	Мінімальна тривалість (років)
ects_min	Integer	Так	Мінімальний обсяг кредитів ECTS
is_active	Boolean	Так	Чи є активним запис

Словник: Форми навчання (FormOfStudy)

Реквізит	Тип даних	Обов'язковий	Опис
id	UUID	Так	Унікальний ідентифікатор
code	String (10)	Так	Код форми навчання (FULL, DIST, MIX)
name_ua	String (150)	Так	Назва українською
supports_mixed	Boolean	Так	Підтримка змішаних форматів
requires_online_platform	Boolean	Так	Потреба в LMS платформі

Реєстр: Особові справи (Student)

Реквізит	Тип даних	Обов'язковий	Опис
id	UUID	Так	Унікальний ідентифікатор
external_id	String	Ні	Зв'язок з ЄДЕБО
full_name	String	Так	Прізвище, Ім'я, По батькові
date_of_birth	Date	Так	Дата народження
group_id	UUID	Так	Зв'язок з групою (Group.id)
status	Enum	Так	Поточний статус (Active, Graduated тощо)
kep_cert	Binary	Ні	Збережений КЕП сертифікат

Реєстр: Записи оцінювання (JournalRecord)

Реквізит	Тип даних	Обов'язковий	Опис
id	UUID	Так	Унікальний ідентифікатор
student_id	UUID	Так	FK → Student
course_id	UUID	Так	FK → Course
lesson_date	Date	Так	Дата уроку
attendance_status	Enum	Так	Присутність (Present, Absent, Sick)
grade	Decimal	Ні	Числова оцінка (1.0 - 12.0 або 0.0 - 100.0)
grade_type	Enum	Так	Посилання на GradeType.code
competence_scores	Map	Ні	Бали за компетенції
signed_by	UUID	Ні	Вчитель, що підписав відомість КЕП
signed_at	Timestamp	Ні	Дата підпису

3.1.3. Алгоритми конвертації оцінок

Модуль виконує автоматичний перерахунок оцінок за такими правилами:

- **12-бальна шкала у літерну ECTS:**

$$\text{ECTS} = \begin{cases} A, & \text{якщо Grade} \geq 10 \\ B, & \text{якщо Grade} == 9 \\ C, & \text{якщо Grade} == 8 \\ D, & \text{якщо Grade} == 7 \\ E, & \text{якщо Grade} == 6 \\ FX, & \text{якщо Grade} \geq 4 \land \text{Grade} < 6 \\ F, & \text{якщо Grade} < 4 \end{cases}$$

- **12-бальна шкала у відсоткову 100-бальну:**

$$\text{Percent} = \text{round}\left(\frac{\text{Grade}}{12} \times 100\right)$$

- **Конвертація компетенцій у числову шкалу:** Розвинена → 12; Базова → 8; Початкова → 5; Не сформована → 2.

3.2. Програмне забезпечення (Реалізація)

3.2.1. bpe_education_assessment.erl (DSL процесу оцінювання)

Код модуля опису FSM процесу оцінювання та валідації успішності студентів:

```
-module(bpe_education_assessment).
-include("bpe.hrl").
-compile(export_all).

def() ->
    #process{
        name = 'Student Assessment',
        beginEvent = 'StartAssessment',
        endEvent = 'AssessmentSigned',
        tasks = [
            #beginEvent{name='StartAssessment'},
            #userTask{name='EnterGrades', module=?MODULE},
            #serviceTask{name='ValidateGrades', module=?MODULE},
            #serviceTask{name='CalculateCompetences', module=?MODULE},
            #userTask{name='TeacherSign', module=?MODULE},
            #userTask{name='ViceCheck', module=?MODULE},
            #endEvent{name='AssessmentSigned'}
        ],
        flows = [
            #sequenceFlow{name='1', source='StartAssessment', target='EnterGrades'},
            #sequenceFlow{name='2', source='EnterGrades', target='ValidateGrades'},
            #sequenceFlow{name='3', source='ValidateGrades',
                          target='CalculateCompetences', condition="validation_ok"},
            #sequenceFlow{name='4', source='ValidateGrades',
                          target='EnterGrades', condition="validation_failed"},
            #sequenceFlow{name='5', source='CalculateCompetences', target='TeacherSign'},
            #sequenceFlow{name='6', source='TeacherSign',
                          target='ViceCheck', condition="requires_approval"},
            #sequenceFlow{name='7', source='TeacherSign',
                          target='AssessmentSigned', condition="no_approval_needed"},
            #sequenceFlow{name='8', source='ViceCheck',
                          target='AssessmentSigned', condition="vice_approved"}
        ],
        roles = [teacher, vice_rector]
    }.

action({serviceTask, 'ValidateGrades'}, Proc) ->
    case validate_grades(Proc) of
        ok -> {reply, Proc, "validation_ok"};
        _  -> {reply, Proc, "validation_failed"}
    end;
```

```
action({userTask, 'TeacherSign'}, Proc) ->
    case requires_vice_approval(Proc) of
        true -> {reply, Proc, "requires_approval"};
        false -> {reply, Proc, "no_approval_needed"}
    end;

action(_, Proc) -> {reply, Proc}.
```

3.2.2. bpe_education_lesson.erl (DSL процесу проведення заняття)

```
-module(bpe_education_lesson).
-include("bpe.hrl").
-compile(export_all).

def() ->
    #process{
        name = 'Lesson Execution and Assessment',
        beginEvent = 'LessonStart',
        endEvent = 'LessonClosed',
        tasks = [
            #beginEvent{name='LessonStart'},
            #userTask{name='TakeAttendance', module=?MODULE},
            #userTask{name='AssessStudents', module=?MODULE},
            #serviceTask{name='CalculateAverages', module=?MODULE},
            #userTask{name='SignJournal', module=?MODULE},
            #endEvent{name='LessonClosed'}
        ],
        flows = [
            #sequenceFlow{name='1', source='LessonStart', target='TakeAttendance'},
            #sequenceFlow{name='2', source='TakeAttendance', target='AssessStudents'},
            #sequenceFlow{name='3', source='AssessStudents', target='CalculateAverages'},
            #sequenceFlow{name='4', source='CalculateAverages',
                          target='SignJournal', condition="averages_ok"},
            #sequenceFlow{name='5', source='CalculateAverages',
                          target='AssessStudents', condition="averages_need_correction"},
            #sequenceFlow{name='6', source='SignJournal',
                          target='LessonClosed', condition="kep_signed"}
        ],
        events = [#timeoutEvent{name='JournalDeadline',
                                timeout=#timeout{spec={hours,48}}}]
    }.

action({serviceTask, 'CalculateAverages'}, Proc) ->
```

```
    case averages_valid(Proc) of
        true -> {reply, Proc, "averages_ok"};
        false -> {reply, Proc, "averages_need_correction"}
    end;

action({userTask, 'SignJournal'}, Proc) ->
    case has_valid_kep(Proc) of
        true -> {reply, Proc, "kep_signed"};
        false -> {reply, Proc}
    end;

action(_, Proc) -> {reply, Proc}.
```

3.2.3. bpe_curriculum_approval.erl (DSL затвердження навчального плану)

```
-module(bpe_curriculum_approval).
-include("bpe.hrl").
-compile(export_all).

def() ->
    #process{
        name = 'Curriculum Approval',
        beginEvent = 'Start',
        endEvent = 'PlanApproved',
        tasks = [
            #beginEvent{name='Start'},
            #userTask{name='CreatePlan', module=?MODULE},
            #userTask{name='MethodistReview', module=?MODULE},
            #userTask{name='ViceApprove', module=?MODULE},
            #userTask{name='DirectorApprove', module=?MODULE},
            #serviceTask{name='GenerateSchedule', module=?MODULE},
            #endEvent{name='PlanApproved'}
        ],
        flows = [
            #sequenceFlow{name='1', source='Start', target='CreatePlan'},
            #sequenceFlow{name='2', source='CreatePlan', target='MethodistReview'},
            #sequenceFlow{name='3', source='MethodistReview',
                          target='ViceApprove', condition="methodist_ok"},
            #sequenceFlow{name='4', source='MethodistReview',
                          target='CreatePlan', condition="methodist_reject"},
            #sequenceFlow{name='5', source='ViceApprove',
                          target='DirectorApprove', condition="vice_ok"},
            #sequenceFlow{name='6', source='ViceApprove',
                          target='MethodistReview', condition="vice_reject"},
```

```
            #sequenceFlow{name='7', source='DirectorApprove',
                          target='GenerateSchedule', condition="director_ok"},
            #sequenceFlow{name='8', source='GenerateSchedule', target='PlanApproved'}
        ],
        roles = [methodist, vice_rector, director]
    }.

action({userTask, 'CreatePlan'}, Proc) -> {reply, Proc};
action({userTask, 'MethodistReview'}, Proc) ->
    case validate_curriculum(Proc) of
        ok -> {reply, Proc, "methodist_ok"};
        _  -> {reply, Proc, "methodist_reject"}
    end;

action({userTask, 'ViceApprove'}, Proc) ->
    case vice_approval(Proc) of
        ok -> {reply, Proc, "vice_ok"};
        _  -> {reply, Proc, "vice_reject"}
    end;

action({userTask, 'DirectorApprove'}, Proc) ->
    case director_kep_sign(Proc) of
        ok -> {reply, Proc, "director_ok"};
        _  -> {reply, Proc}
    end;

action({serviceTask, 'GenerateSchedule'}, Proc) ->
    {reply, Proc};

action(_, Proc) -> {reply, Proc}.
```

3.2.4. bpe_schedule_generation.erl (DSL генерації розкладу)

```
-module(bpe_schedule_generation).
-include("bpe.hrl").
-compile(export_all).

def() ->
    #process{
        name = 'Schedule Generation',
        beginEvent = 'PlanApproved',
        endEvent = 'SchedulePublished',
        tasks = [
            #beginEvent{name='PlanApproved'},
```

```
            #serviceTask{name='LoadPlanAndConstraints', module=?MODULE},
            #serviceTask{name='GenerateSchedule', module=?MODULE},
            #userTask{name='ManualAdjustment', module=?MODULE},
            #userTask{name='ViceDirectorApprove', module=?MODULE},
            #serviceTask{name='PublishAndNotify', module=?MODULE},
            #endEvent{name='SchedulePublished'}
        ],
        flows = [
            #sequenceFlow{name='1', source='PlanApproved', target='LoadPlanAndConstraints'},
            #sequenceFlow{name='2', source='LoadPlanAndConstraints', target='GenerateSchedul
            #sequenceFlow{name='3', source='GenerateSchedule',
                          target='ViceDirectorApprove', condition="no_conflicts"},
            #sequenceFlow{name='4', source='GenerateSchedule',
                          target='ManualAdjustment', condition="has_conflicts"},
            #sequenceFlow{name='5', source='ManualAdjustment', target='GenerateSchedule'},
            #sequenceFlow{name='6', source='ViceDirectorApprove',
                          target='PublishAndNotify', condition="approved"},
            #sequenceFlow{name='7', source='PublishAndNotify', target='SchedulePublished'}
        ]
    }.

action({serviceTask, 'LoadPlanAndConstraints'}, Proc) -> {reply, Proc};
action({serviceTask, 'GenerateSchedule'}, Proc) ->
    case generate_schedule_algorithm(Proc) of
        {ok, Schedule} ->
            {reply, Proc#process{docs = [Schedule | Proc#process.docs]}, "no_conflicts"};
        {error, Conflicts} ->
            {reply, Proc#process{docs = [Conflicts | Proc#process.docs]}, "has_conflicts"}
    end;
action({userTask, 'ManualAdjustment'}, Proc) -> {reply, Proc};
action({userTask, 'ViceDirectorApprove'}, Proc) ->
    case has_valid_kep(Proc) of
        true -> {reply, Proc, "approved"};
        false -> {reply, Proc}
    end;
action({serviceTask, 'PublishAndNotify'}, Proc) -> {reply, Proc};
action(_, Proc) -> {reply, Proc}.
```

3.2.5. grade_validator.erl (Модуль валідації надійності даних та оцінок)

Код Erlang-модуля для перевірки порогових значень та відсутності невалідності типів даних:

```
-module(grade_validator).
-export([validate/2, validate_thresholds/2]).
```

```
-record(grade_threshold, {
    id,
    grade_type_code,
    threshold_name,
    value,
    operator,
    message_ua,
    is_blocking = true,
    requires_approval = false
}).

validate(GradeTypeCode, Value) ->
    Thresholds = fetch_thresholds(GradeTypeCode),
    validate_thresholds(Value, Thresholds).

validate_thresholds(_Value, []) -> ok;
validate_thresholds(Value, [T|Rest]) ->
    case check_threshold(Value, T) of
        ok -> validate_thresholds(Value, Rest);
        {error, Msg} -> {error, Msg}
    end.

check_threshold(Value, #grade_threshold{operator = '>=', value = Th}) when Value >= Th -> ok
check_threshold(Value, #grade_threshold{operator = '>', value = Th}) when Value > Th -> ok;
check_threshold(Value, #grade_threshold{operator = '<=', value = Th}) when Value =< Th -> ok
check_threshold(_, T) -> {error, T#grade_threshold.message_ua}.

fetch_thresholds(<<"CURRENT_12">>) ->
    [
        #grade_threshold{
            id = 1,
            grade_type_code = <<"CURRENT_12">>,
            threshold_name = <<"passing">>,
            value = 6.0,
            operator = '>=',
            message_ua = <<"Poperedzhennia: Ocinka nyzhche prokhidnogo balu (6).">>,
            is_blocking = false
        },
        #grade_threshold{
            id = 2,
            grade_type_code = <<"CURRENT_12">>,
            threshold_name = <<"critical">>,
            value = 4.0,
            operator = '>=',
            message_ua = <<"Pomylka: Ocinka ie krytychno nezadovilnoiu. Potriben komentar pe
```

```
            is_blocking = true
        }
    ];
fetch_thresholds(_) -> [].
```

4. Вимоги до засобу за ISO/IEC/IEE 29148

4.1. Функціональні вимоги

- **[REQ-EDU-FUN-001]** Модуль «Освіта» повинен забезпечувати ведення та оновлення словників типів освітніх програм (`EducationalProgramType`), форм навчання (`FormOfStudy`) та типів оцінок (`GradeType`) з можливістю версіонування записів.
- **[REQ-EDU-FUN-002]** Модуль «Освіта» повинен підтримувати автоматичний перерахунок оцінок між 12-бальною шкалою, літерними оцінками ECTS та 100-бальною відсотковою шкалою за заданими математичними формулами.
- **[REQ-EDU-FUN-003]** Модуль «Освіта» повинен реалізовувати скінченні автомати (FSM) життєвого циклу занять та ведення журналів.
- **[REQ-EDU-FUN-004]** Модуль «Освіта» повинен забезпечувати автоматичне блокування редагування записів електронного журналу через 48 годин після фактичного початку заняття.
- **[REQ-EDU-FUN-005]** Модуль «Освіта» повинен надавати викладачам інтерфейс для накладання кваліфікованого електронного підпису (КЕП) на відомості оцінювання та журнали.
- **[REQ-EDU-FUN-006]** Модуль «Освіта» повинен автоматично розраховувати інтегральні компетенції учнів на основі поєднання поточних балів за встановленою вагою.
- **[REQ-EDU-FUN-007]** Модуль «Освіта» повинен підтримувати створення та затвердження навчальних планів за допомогою регламентованого процесу погодження (Методист — Завуч — Директор).
- **[REQ-EDU-FUN-008]** Модуль «Освіта» повинен забезпечувати перевірку наявності колізій у розкладі занять (накладки викладачів, зайнятість аудиторій).
- **[REQ-EDU-FUN-009]** Модуль «Освіта» повинен надсилати сповіщення у разі фіксації критично низьких оцінок (< 4) учням та їхнім батькам.
- **[REQ-EDU-FUN-010]** Модуль «Освіта» повинен забезпечувати вивантаження даних успішності до державної системи ЄДЕБО за допомогою захищеного HTTPS REST API.

4.2. Вимоги до інтерфейсів та дизайну

- **[REQ-EDU-UI-001]** Користувацький інтерфейс повинен відповідати вимогам доступності ДСТУ EN 301 549:2022 (WCAG 2.1 AA) для забезпечення зручності роботи користувачів з особливими потребами.
- **[REQ-EDU-UI-002]** Веб-інтерфейс повинен бути реалізований за технологією Single Page Application (SPA) на базі WebSocket для мінімізації трафіку та забезпечення миттєвого відгуку.
- **[REQ-EDU-UI-003]** Система повинна використовувати підхід Data-on-Wire — передавати тільки чисті структуровані дані у форматі JSON без повторного пересилання HTML-шаблонів.
- **[REQ-EDU-UI-004]** Дизайн системи повинен автоматично адаптуватися під екрани з розширенням від 320px до 3840px (Responsive Web Design).

4.3. Вимоги до безпеки та захисту інформації

- **[REQ-EDU-SEC-001]** Автентифікація всіх посадових осіб (адміністратори, педагоги, методисти) повинна здійснюватися виключно за допомогою КЕП у форматі CAdES-X Long.
- **[REQ-EDU-SEC-002]** Передача даних у мережі повинна здійснюватися виключно через захищений канал HTTPS (TLS 1.3).
- **[REQ-EDU-SEC-003]** Усі відкриті порти, крім порту 443 (HTTPS), повинні бути закриті на рівні мережевого екрану.
- **[REQ-EDU-SEC-004]** Оскільки реляційні бази даних не використовуються, ризик SQL-ін'єкцій відсутній; натомість Erlang-сервіси повинні виконувати суворе очищення текстових полів для запобігання XSS-атакам.
- **[REQ-EDU-SEC-005]** Модуль авторизації повинен перевіряти ABAC-атрибути суб'єкта та об'єкта доступу перед виконанням будь-якої операції читання чи запису.
- **[REQ-EDU-SEC-006]** Токени користувацьких сесій повинні записуватися в Cookie з атрибутами `Secure`, `HttpOnly` та `SameSite=Strict`.
- **[REQ-EDU-SEC-007]** Модуль авторизації повинен блокувати паралельні сесії одного користувача з різних IP-адрес.
- **[REQ-EDU-SEC-008]** Дані користувачів повинні бути захищені відповідно до Закону «Про захист персональних даних».

4.4. Вимоги до продуктивності та надійності

- **[REQ-EDU-PERF-001]** Час реакції системи на створення асинхронної фонової операції не повинен перевищувати 200 мс.
- **[REQ-EDU-PERF-002]** Пропускна здатність черги обробки транзакцій Mnesia повинна складати не менше 500 операцій на секунду на один серверний вузол.
- **[REQ-EDU-PERF-003]** Затримка оновлення даних в електронному журналі через Secure WebSockets не повинна перевищувати 150 мс.
- **[REQ-EDU-PERF-004]** Час повного рендерингу та відображення сторінки журналу на клієнті не повинен перевищувати 1 секунди при розмірі сторінки до 2000 записів.

4.5. Вимоги до логування та аудиту

- **[REQ-EDU-LOG-001]** Усі критичні системні події (авторизація, зміна оцінок, накладання КЕП, помилки авторизації) повинні реєструватися у централізованому журналі аудиту (Audit Log).
- **[REQ-EDU-LOG-002]** Логи повинні експортуватися у режимі реального часу до стеку ELK (Elasticsearch, Logstash, Kibana) за допомогою Filebeat.
- **[REQ-EDU-LOG-003]** Запис логу повинен мати формат JSON та обов'язково містити поля: мітка часу (`time`), рівень критичності (`severity`), унікальний код події, ідентифікатор користувача (`user_id`) та деталі помилки.

4.6. Адміністративні та юридичні вимоги

- **[REQ-EDU-ADM-001]** Програмний код Модуля «Освіта» повинен бути вкритий автоматичними тестами (Unit та Integration) не менше ніж на 80%.
- **[REQ-EDU-ADM-002]** Для тестування у непублічних середовищах повинна бути передбачена можливість запуску без підключення до реальних сервісів КЕП.
- **[REQ-EDU-ADM-003]** Вендор повинен надати повну технічну документацію на засіб інформатизації, що містить Настанову користувача, Настанову адміністратора та Опис API за стандартом ДСТУ 3008:2015.

- **[REQ-EDU-ADM-004]** Усі майнові права інтелектуальної власності на створене програмне забезпечення повинні бути повністю передані замовнику (державі в особі ДСА України).