ERP/1: Освіта

Технічні вимоги

до системи автоматизації та
управління освітнім процесом

Зміст

Зміст

Анотація

У цьому документі наведено детальні технічні вимоги до підсистеми ERP/1: Освіта, яка розробляється як сучасна LMS/EMS-платформа на базі Erlang/OTP. Вимоги розроблено згідно з Політиками Технічних вимог, ТЗ та ТД (ДСТУ 3973-2000, ДСТУ 3008:2015 та ISO/IEC/IEEE 42010). Документ визначає повний перелік довідників, реєстрових сутностей, бізнес-процесів (включаючи навчальні плани, електронний журнал, оцінювання та генерацію розкладу занять), рольову модель доступів на базі ABAC, а також нефункціональні вимоги до архітектури, безпеки (КЕП CAdES-X Long), продуктивності та логування.

1. Умовні скорочення та визначення

Терміни та скорочення, що використовуються в цьому документі:

Термін / Скорочення	**Значення**
LMS	Learning Management System (Система управління навчанням)
EMS	Educational Management System (Система управління освітнім процесом)
КЕП / QES	Кваліфікований електронний підпис / Qualified Electronic Signature
ЄДЕБО	Єдина державна електронна база з питань освіти
ЄДРПОУ	Єдиний державний реєстр підприємств та організацій України
BPMN	Business Process Model and Notation (Модель та нотація бізнес-процесів)
FSM	Finite State Machine (Скінченний автомат)
ABAC	Attribute-Based Access Control (Контроль доступу на основі атрибутів)
RBAC	Role-Based Access Control (Рольовий контроль доступу)
SCORM	Sharable Content Object Reference Model (Стандарт обміну електронними курсами)
REST	Representational State Transfer (Стиль архітектури програмного забезпечення)
gRPC	Google Remote Procedure Call (Механізм віддаленого виклику процедур)
MQTT	Message Queuing Telemetry Transport (Протокол передачі повідомлень)
PKI	Public Key Infrastructure (Інфраструктура відкритих ключів)
ELK	Elasticsearch, Logstash, Kibana (Стек логування та моніторингу)
OTP	Open Telecom Platform (Набір бібліотек та шаблонів проектування для Erlang)

2. Загальні відомості

2.1. Передумови

Проект ERP/1: Освіта створюється як сучасна високотехнологічна LMS/EMS-платформа на базі Erlang/OTP, що призначена для автоматизації та управління освітнім процесом (очна, дистанційна та змішана форми навчання). Впровадження платформи покликане забезпечити перехід до цифрового ведення документації, підвищення прозорості оцінювання та спрощення обміну даними між навчальними закладами й державними базами даних.

2.2. Питання, що вирішуються

Платформа вирішує такі ключові завдання:

- Організація цифрового обліку контингенту (особових справ учнів/студентів) та кадрового складу педагогів.
- Управління навчальними планами й автоматична генерація розкладів занять.
- Ведення електронних журналів із фіксацією відвідуваності та оцінок.
- Засвідчення юридичної значущості записів оцінювання за допомогою інтеграції КЕП.
- Інтеграція з Єдиною державною електронною базою з питань освіти (ЄДЕБО).

2.3. Вимоги законодавства та міжнародних стандартів

Система повинна розроблятися та функціонувати відповідно до:

- Закону України «Про освіту»;
- Закону України «Про захист персональних даних»;
- Закону України «Про захист інформації в інформаційно-телекомунікаційних системах»;
- Закону України «Про основні засади забезпечення кібербезпеки України»;
- Закону України «Про інформацію»;

- Постанови Кабінету Міністрів України № 205 від 21.02.2025 «Про затвердження Порядку використання засобів інформатизації»;
- Нормативних стандартів ДСТУ 3973-2000, ДСТУ 3974-2000 та ДСТУ 3008:2015;
- Міжнародних стандартів ISO/IEC/IEEE 42010 (опис архітектури) та рекомендацій NIST.

3. Призначення та цілі впровадження

Основною метою впровадження ERP/1: Освіта є підвищення ефективності управління навчальними закладами через автоматизацію рутинних операцій, забезпечення надійного захисту персональних даних та результатів навчання, а також надання зручних інструментів взаємодії для всіх учасників освітнього процесу (адміністрації, вчителів, учнів та батьків).

4. Класифікація вимог

4.1. Функціональні вимоги

4.1.1. Опис довідників

Платформа повинна підтримувати систему довідників (словників) для класифікації, валідації та уніфікації даних. До обов'язкових довідників відносяться:

- **Форми навчання (FormOfStudy):** Очна, Дистанційна, Змішана, Екстернат, Вечірня/Заочна, Дуальна.
- **Статуси учасників освітнього процесу (ParticipantStatus):** Активний, Академічна відпустка, Відрахований, Випускник, На переведенні, Тимчасово відсутній.
- **Типи оцінок / Шкали оцінювання (GradeType / GradeScale):** 12-бальна, Залікова (Зараховано/Не зараховано), ECTS (A-F), Компетенційна, Бальна (0-100).
- **Типи навчальних занять (LessonType):** Лекція, Практичне заняття, Семінар, Лабораторна робота, Контрольна робота, Самостійна робота, Консультація, Залік / Екзамен.
- **Статуси журналу (JournalStatus):** Чернетка, Зафіксовано, Підписано педагогом (КЕП), Перевірено завучем, Архівний.
- **Ролі учасників (Role):** Учень/Студент, Педагог/Викладач, Класний керівник, Завуч/Методист, Директор/Ректор, Батьки/Представники, Admin, Комісія атестації.
- **Посади педагогічних працівників (Position):** Директор/Ректор, Завуч/Проректор, Вчитель/Викладач, Старший викладач, Доцент, Професор, Майстер виробничого навчання.
- **Кваліфікаційні категорії (QualificationCategory):** Спеціаліст, Спеціаліст другої категорії, Спеціаліст першої категорії, Спеціаліст вищої категорії, Майстер-викладач.
- **Типи компетенцій (CompetenceType):** Предметні, Загальноосвітні, Цифрові, Соціально-емоційні, Громадянські.
- **Типи документів (DocumentType):** Наказ, Особова справа, Атестаційний лист, Звіт для МОН, Довідка, Договір, Протокол засідання.
- **Статуси документів (DocumentStatus):** Чернетка, На погодженні, Підписано (КЕП), Зареєстровано, Архівний.

- **Типи звітності (ReportType):** Статистична (форма №1), Фінансова, Навчальна успешність, Контингент, Кадрова.
- **Типи інтеграцій (IntegrationType):** ЄДЕБО, MQTT, PKI / КЕП, SCORM, Зовнішній реєстр.
- **Рівні доступу / Чутливість (SensitivityLevel):** Публічний, Внутрішній, Конфіденційний, Персональні дані (GDPR).
- **Типи подій логування (LogEventType):** Зміна оцінки, Підпис КЕП, Вхід в систему, Помилка валідації, Адміністративна дія.
- **Типи освітніх програм (EducationalProgramType):** Загальна середня, Профільна середня, Професійно-технічна, Фахова передвища, Вища освіта (бакалавр), Вища освіта (магістр), Дуальна форма.

Алгоритми конвертації шкал оцінювання Система повинна забезпечувати автоматичну конвертацію між різними шкалами:

1. **12-бальна → ECTS (європейська):**
   - $\geq 10 \to A$, $\geq 9 \to B$, $\geq 8 \to C$, $\geq 7 \to D$, $\geq 6 \to E$, $\geq 4 \to FX$, менше $4 \to F$.
2. **12-бальна → 100-бальна (відсоткова):**
   - $Percent = \text{round}((Grade/12) \times 100)$.
3. **100-бальна → 12-бальна:**
   - $\geq 95 \to 12$, $\geq 86 \to 11$, $\geq 78 \to 10$, $\geq 71 \to 9$, $\geq 64 \to 8$, $\geq 57 \to 7$, $\geq 50 \to 6$, $\geq 43 \to 5$, $\geq 36 \to 4$, $\geq 29 \to 3$, $\geq 22 \to 2$, менше $22 \to 1$.
4. **Компетенційна шкала → числова:**
   - Розвинена $\to 10$–12, Базова $\to 7$–9, Початкова $\to 4$–6, Не сформована $\to 1$–3.
5. **Загальна формула підсумкової оцінки за семестр:**
   - $FinalGrade = \text{round}(CurrentAvg \times 0.6 + FinalAvg \times 0.4)$, де $CurrentAvg$ — середнє арифметичне поточних балів, а $FinalAvg$ — бал за підсумкову роботу.

Детальний опис схем довідників та алгоритмів наведено в Додатку А та Додатку Б відповідно.

4.1.2. Опис реєстрових сутностей та їх станів

Основними реєстровими сутностями системи є:

- **Student (Учень / Студент):** Особові дані учнів. Стани: *Активний*, *Академічна відпустка*, *Відрахований*, *Випускник*.

- **Teacher (Педагог / Викладач):** Особові та професійні дані педагогів. Стани: *Активний*, *Тимчасово відсутній*, *Звільнений*.
- **Group (Група / Клас):** Дані про академічну групу. Стани: *Активна*, *Випущена*.
- **Course (Курс / Дисципліна):** Опис навчального предмету та SCORM-матеріалів. Стани: *Активний*, *Архівний*.
- **Curriculum (Навчальний план):** План на рік/семестр. Стани: *Чернетка*, *На погодженні*, *Затверджено*, *Архівний*.
- **JournalRecord (Журнал / Запис оцінювання):** Оцінки, відвідуваність та компетенції. Стани: *Чернетка*, *Зафіксовано*, *Підписано*, *Архівний*.
- **Schedule (Розклад занять):** Структура розкладу. Стани: *Чернетка*, *Активований*.
- **Homework (Домашнє завдання):** Дані про завдання та додатки. Стани: *Створено*, *Опубліковано*.
- **Document (Документ):** Накази, договори, особові справи. Стани: *Чернетка*, *На погодженні*, *Підписано КЕП*, *Зареєстровано*, *Архівний*.
- **Role/Permission (Ролі та права):** Права доступу користувачів.
- **Competence (Компетенція):** Карта компетенцій відповідно до держстандартів.
- **ProcessLog (Подія / Історія процесу):** Лог аудиту дій користувачів та процесів.
- **EducationalProgram (Освітня програма):** Реєстр програм навчання. Стани: *Чернетка*, *Акредитована*, *Неакредитована*.

Зв'язки між сутностями та їх кардинальність

- Учень належить до однієї Групи ($N : 1$).
- Група має Навчальний план на семестр ($1 : N$).
- Навчальний план містить багато Курсів ($N : M$ через проміжну сутність `CurriculumItem`).
- Запис в журналі створюється для одного Учня та одного Курсу ($N : 1$).
- Запис в журналі фіксується одним Педагогом ($N : 1$).
- Розклад занять прив'язаний до однієї Групи, Курсу та Педагога ($N : 1$).
- Документи поліморфно зв'язані з будь-якою сутністю ($1 : N$).

Детальні таблиці реквізитів сутностей наведено в Додатку А.

4.1.3. Опис бізнес-процесів

Освітні бізнес-процеси в ERP/1 побудовані на базі FSM (Finite State Machines) та інтегруються з рушієм процесів BPMN:

1. Процес «Формування та затвердження навчального плану / розкладу» (Curriculum Approval):

- **Крок 1:** Методист створює або імпортує навчальний план (сутність `Curriculum`).
- **Крок 2:** Методист проводить валідацію плану на відповідність державній програмі. Якщо знайдено помилки — повертає на доопрацювання.
- **Крок 3:** Завуч переглядає та погоджує план. При відхиленні — вказує коментар та повертає на крок 1.
- **Крок 4:** Директор підписує план КЕП.
- **Крок 5:** Сервіс автоматично запускає генерацію розкладу на основі затвердженого плану.

2. Процес «Проведення заняття та фіксація відвідувань/оцінок» (Core навчальний процес):

- **Крок 1:** Початок уроку за розкладом.
- **Крок 2:** Педагог відкриває форму електронного журналу групи.
- **Крок 3:** Фіксація відвідувань (вручну або через QR/mobile check-in).
- **Крок 4:** Виставлення оцінок та балів компетенцій за відповідною шкалою.
- **Крок 5:** Автоматичний розрахунок середнього балу та оновлення аналітики.
- **Крок 6:** Підпис журналу КЕП педагога протягом 48 годин. Після підпису зміни допускаються лише через процедуру аудитованого «Виправлення».

3. Процес «Генерація розкладу» (Schedule Generation):

- **Крок 1:** Запуск процесу після затвердження плану. Завантаження обмежень (кількість аудиторій, доступність викладачів, очна/дистанційна форма).
- **Крок 2:** Автоматичний розрахунок варіанту розкладу.
- **Крок 3:** Перевірка наявності конфліктів. Якщо конфлікти є — перехід до ручного коригування методистом.
- **Крок 4:** Перевірка та затвердження розкладу завучем або директором.
- **Крок 5:** Публікація розкладу в системі та розсилка сповіщень.

4. Процес «Атестація та підвищення кваліфікації педагогів»:

- **Крок 1:** Подання особової справи та портфоліо педагогом.
- **Крок 2:** Автоматичний розрахунок інтегрального показника KPI.
- **Крок 3:** Погодження атестаційного листа атестаційною комісією (підписи КЕП).
- **Крок 4:** Затвердження наказу директором.
- **Крок 5:** Оновлення штатного розпису та кваліфікаційної категорії педагога.

5. Процес «Формування звітності для МОН» (інтеграційний):

- **Крок 1:** Збір даних про успішність, відвідуваність та контингент.
- **Крок 2:** Валідація даних на відповідність державній формі звіту.
- **Крок 3:** Підписання звіту КЕП відповідальної особи.
- **Крок 4:** Передача даних до ЄДЕБО/МОН через інтеграційну шину.

Описи процесів на Erlang BPE DSL наведено в Додатку Б.

4.1.4. Вимоги до протоколів взаємодії

Усі API-методи, що розробляються для модуля «Освіта», повинні бути задокументовані за єдиним стандартом:

- **Призначення методу:** яку дію виконує.
- **Формулювання методу:** HTTP метод та ендпойнт.
- **Параметри виклику:** таблиця вхідних параметрів із зазначенням типу даних, обов'язковості та прикладів.
- **Відповідь сервера:** приклад структури у форматі JSON із кодом 200 OK.
- **Обробка помилок:** таблиця з кодами помилок та їх описом.
- **Додаткова інформація:** права доступу (Scopes), обмеження частоти запитів (Rate Limiting).

Зовнішня інтеграція здійснюється через шину «Трембіта» (з реєстрами ЄДЕБО, МОН) за допомогою REST API та безпечних gRPC контрактів.

4.1.5. Опис прав та привілеїв (ABAC/RBAC/ACL)

Модель контролю доступу базується на гібридному підході (RBAC + ABAC):

- **Subject (Суб'єкт):** `user_id`, `roles` (Teacher, Student, Parent, Admin), `person_id`, `kep_cert_thumbprint`, `is_active`.
- **Object (Об'єкт):** `entity_type` (Student, JournalRecord, Curriculum), `owner_id`, `group_id`, `status` (Draft, Signed, Archive).
- **Action (Дія):** `read`, `write`, `sign`, `delete`, `approve`, `view_grades`.
- **Environment (Середовище):** `time` (робочий час), `ip_range`, `device_type`, `session_risk`.

Ключові бізнес-правила доступу

1. **Педагог:** Дозволено `read`, `write`, `sign` для журналу (`JournalRecord`), якщо він є викладачем цього курсу (`teacher_id == subject.person_id`) або курс входить до його списку дисциплін. Заборонено `delete` та `write` після підписання журналу КЕП.
2. **Учень:** Дозволено `read` журналу тільки для власних записів (`student_id == subject.person_id`). Заборонено `write`, `sign`.
3. **Батько/Мати:** Дозволено `read` для записів успішності дитини (`student_id in subject.children_ids`). Обмежено відображення службових коментарів педагога.
4. **Адміністратор/Директор:** Повний доступ до всіх дій із обов'язковим логуванням (Audit Log).
5. **Часові обмеження:** Редагування журналу дозволено тільки протягом 48 годин після проведення уроку за розкладом (для непідписаних записів).

4.1.6. Опис валідацій

Валідація повинна відбуватися на трьох рівнях: Frontend (попередній контроль), Backend (API контроль) та Mnesia level (контроль на рівні доступу до даних та Erlang-специфікацій).

Валідація порогів оцінювання (GradeThreshold) Валідація порогів пов'язана з довідником `GradeType` та перевіряє значення оцінок:

- Поріг `passing` (прохідний бал): ≥ 6 (для 12-бальної шкали). Якщо оцінка нижча — видається попередження, але збереження дозволене.
- Поріг `excellent`: ≥ 10.
- Поріг `critical` (критично незадовільно): < 4. Це блокуюча валідація, що вимагає обов'язкового коментаря педагога та додаткового погодження завучем при підсумковому оцінюванні.

Інші обов'язкові валідації

- Сума кредитів навчального плану повинна дорівнювати обсягу кредитів освітньої програми (`Curriculum.total_credits == EducationalProgram.credits_ects`).
- Заборона накладок у розкладі занять (один викладач або аудиторія не можуть бути зайняті в один час).
- Наявність дійсного КЕП при затвердженні планів та журналів.

4.1.7. Шаблони сповіщень

Система повинна забезпечувати автоматичне формування та відправку сповіщень:

- Сповіщення учням та батькам про отримання незадовільних оцінок (< 4) або пропуски занять.
- Сповіщення педагогам про наближення дедлайну підписання журналу (за 12 годин до завершення 48-годинного вікна).
- Сповіщення завучу при масовій відсутності учнів у групі (понад 30% відсутніх на занятті).

Доставка сповіщень здійснюється через інтеграційну шину MQTT (push-сповіщення у додатках) або Email.

4.1.8. Вимоги до серіалізації

Усі дані для обміну між компонентами системи та зовнішніми інтерфейсами повинні передаватися у форматі JSON. Для зберігання неструктурованих метаданих (наприклад, файлових додатків до домашніх завдань або специфічних параметрів навчального плану) в Mnesia використовується формат JSON або вбудовані структури Erlang (Maps/Records).

4.1.9. Вимоги до транспорту

Передача даних повинна здійснюватися через:

- HTTPS з підтримкою протоколу TLS 1.3;
- Secure WebSockets (WSS) для забезпечення роботи веб-інтерфейсу в режимі реального часу;
- MQTT із шифруванням TLS для надсилання миттєвих повідомлень та телеметрії.

4.2. Нефункціональні вимоги

4.2.1. Вимоги до архітектури

Архітектура системи будується на базі методологии ISO/IEC/IEEE 42010 та фреймворку Захмана:

- **Core слой:** Розробка на базі Erlang/OTP, що гарантує високу відмовостійкість, ізоляцію збоїв та підтримку тисяч одночасних підключень.
- **Сховище даних:** Повністю інтегроване Erlang-native сховище. Усі транзакційні, оперативні дані процесів та сесій записуються виключно в Erlang Mnesia (із конфігуруванням таблиць disc_copies, disc_only_copies або ram_copies залежно від вимог до швидкодії та збереженості). Для зберігання медіафайлів (особові справи, домашні завдання) використовується S3-сумісне сховище.
- **Масштабування:** Горизонтальне масштабування шляхом реплікації таблиць Mnesia між Erlang-вузлами у кластері, контейнеризація сервісів та балансування навантаження через Traefik / KONG.
- **Асинхронність:** Усі часомісткі операції (генерація розкладу, формування звітів) виконуються асинхронно через черги повідомлень.

4.2.2. Вимоги до безпеки

Захист інформації є наскрізним і забезпечується на всіх рівнях моделі OSI:

- **Автентифікація:** Вхід у систему здійснюється виключно через кваліфікований електронний підпис (КЕП) у форматі CAdES-X Long (із зчитуванням сертифікатів X.509 з захищених апаратних носіїв).
- **Мережева безпека:** Відкритим повинен бути лише порт 443 для безпечного з'єднання HTTPS. Усі інші порти мають бути закриті.
- **Шифрування:** Використання TLS 1.3 із сертифікатами ECC (ECDSA) або RSA (DSA з довжиною ключа не менше 2048 біт).
- **Захист від атак (OWASP Top 10):**
  - Оскільки SQL не використовується, ризик SQL-ін'єкцій відсутній. Натомість має бути забезпечена сувора валідація вхідних даних на рівні Erlang-сервісів для запобігання пошкодженню структури рекодів.
  - Захист від XSS-ін'єкцій шляхом блокування введення конструкцій: `javascript:`, `vbscript:`, `data:`, `onclick`, `onload`, `onsubmit`, `?`, `<`, `>`, `NULL`.
  - Використання HTTP заголовків безпеки: CSP, HttpOnly, Secure, HSTS, X-Frame-Options.
  - Заборона використання функцій динамічного виконання коду (`eval`, `exec`).

- **Керування сесіями:** Токени сесій повинні зберігатися в Cookie з атрибутами `Secure` та `HttpOnly`. Заборонено паралельні сесії одного користувача з різних пристроїв.

4.2.3. Вимоги до потужності і ємності

Система повинна відповідати таким показникам продуктивності:

Характеристика	Опис	Цільовий показник
Час реакції	Активація асинхронної задачі	< 200 мс
Пропускна здатність	Кількість виконаних задач за одиницю часу	≥ 500 задач/с
Затримка передачі (latency)	Час приходу першого байту відповіді	< 150 мс
Завантаження таблиць	Повне завантаження таблиці на 2000 рядків	< 1 с
Відображення таблиць	Відображення видимої частини таблиці UI	< 300 мс

4.2.4. Вимоги до функціональності логування

Логування здійснюється за допомогою ELK-стеку (Filebeat, Logstash, Elasticsearch, Kibana):

- **Формат логів:** JSON-об'єкт із обов'язковими полями `time`, `severity` (Trace, Debug, Info, Warning, Error, Critical), `log` (повідомлення).
- **Метадані:** При наявності записуються поля `source_location.file`, `source_location.line`, `error.initial_call`, `error.reason`.
- **Захист даних:** Заборонено записувати персональні дані, паролі та ключі у відкритому вигляді. Ці дані повинні маскуватися або хешуватися.
- **Термін зберігання:** Журнали аудиту безпеки та доступу до конфіденційних даних повинні зберігатися не менше 3 місяців у захищеному від модифікації вигляді.

4.2.5. Адміністративні вимоги

- **Тестування:** Пакет поставки повинен містити автоматичні тести (Unit, Integration) з покриттям коду не менше 80%. Має бути передбачена можливість автоматичного тестування з відключеними сервісами КЕП на тестових середовищах.
- **Документування:** Наявність Настанови адміністратора, Настанови користувача та опису API-методів відповідно до ДСТУ 3008:2015.

- **Технологічний стек:** Повністю безкоштовні open-source залежності з підтримкою розробників не менше 1 року.

4.2.6. Загальні вимоги до документації та артефактів

Вендор зобов'язаний надати повну технічну документацію, що включає архітектурний опис (ISO 42010), схему даних (опис Erlang-рекордів та ER-діаграми), API-специфікацію та UX-прототипи у Figma.

4.2.7. Юридичні вимоги

- Передача замовнику всіх майнових прав інтелектуальної власності на створене програмне забезпечення.
- Врегулювання відносин із авторами (розробниками) та субпідрядниками, надання підтверджень оригінальності коду та відсутності претензій третіх осіб.
- Забезпечення патентної чистоти.
- Надання гарантійної технічної підтримки терміном не менше 1 року після введення системи в експлуатацію.

A. Додаток А. Специфікація реквізитів сутностей (для ТЗ)

У цьому додатку наведено детальні таблиці реквізитів для основних реєстрових сутностей та довідників модуля «Освіта». Ці таблиці призначені для перенесення до Технічного завдання (ТЗ) відповідно до Політик документування.

A.1. А.1. Довідник «Типи освітніх програм» (EducationalProgramType)

Реквізит	Тип даних	Обов'язковий	Опис
id	UUID	Так	Унікальний ідентифікатор
code	String (30)	Так	Унікальний код типу програми
name_ua	String (300)	Так	Назва українською
name_en	String (300)	Ні	Назва англійською
education_level	Enum	Так	Рівень освіти (ДНЗ, ЗСО, ПО, ФПО, ВО тощо)
program_category	Enum	Так	Категорія (Загальна, Профільна, Професійна тощо)
duration_years_min	Decimal	Так	Мінімальна тривалість навчання (років)
duration_years_max	Decimal	Ні	Максимальна тривалість навчання
ects_min	Integer	Так	Мінімальний обсяг кредитів ECTS
description	Text	Ні	Опис типу програми
is_active	Boolean	Так	Статус активності довідника
order	Integer	Так	Порядок відображення
legal_basis	Text	Ні	Посилання на нормативно-правову базу
created_at, updated_at	Timestamp	Так	Дати створення та зміни
version	Integer	Так	Версія запису (для оптимістичного блокування)
metadata	JSONB	Ні	Додаткові гнучкі параметри

A.2. A.2. Довідник «Освітні програми» (EducationalProgram)

Реквізит	Тип даних	Обов'язковий	Опис
id	UUID	Так	Унікальний ідентифікатор
code	String (50)	Так	Код програми (наприклад, 122, 013)
name_ua	String (500)	Так	Повна назва українською
name_en	String (500)	Ні	Назва англійською
level	Enum	Так	Рівень освіти
specialty_code	String (20)	Так	Код спеціальності (ЗКП/ОП)
specialty_name	String (300)	Так	Назва спеціальності
qualification	String (300)	Так	Кваліфікація, що присвоюється
duration_years	Decimal	Так	Тривалість навчання
credits_acts	Integer	Так	Загальний обсяг кредитів ECTS
form_of_study_id	UUID	Так	FK → FormOfStudy
language	Enum	Так	Основна мова викладання
accreditation_status	Enum	Так	Статус акредитації
accreditation_until	Date	Ні	Дата закінчення дії акредитації
license_number	String	Ні	Номер ліцензії на освітню діяльність
department_id	UUID	Так	FK → Department (кафедра/факультет)
is_active	Boolean	Так	Чи є програма активною
order	Integer	Так	Порядок сортування
created_at, updated_at	Timestamp	Так	Дати створення та зміни
version	Integer	Так	Версія запису
metadata	JSONB	Ні	Додаткові метадані

A.3. A.3. Довідник «Форми навчання» (FormOfStudy)

Реквізит	Тип даних	Обов'язковий	Опис
id	UUID	Так	Унікальний ідентифікатор
code	String (10)	Так	Унікальний код форми (FULL, DIST, MIX)
name_ua	String (150)	Так	Назва українською
name_en	String (150)	Ні	Назва англійською
description	Text	Ні	Детальний опис
is_active	Boolean	Так	Флаг активності довідника
order	Integer	Так	Порядок сортування
supports_mixed	Boolean	Так	Чи підтримує змішане навчання

requires_online_platform	Boolean	Так	Чи вимагає системи LMS
legal_basis	String	Ні	Нормативне обґрунтування
created_at, updated_at	Timestamp	Так	Дати створення та зміни
version	Integer	Так	Версія запису
metadata	JSONB	Ні	Додаткові гнучкі параметри

A.4. A.4. Довідник «Типи оцінок» (GradeType)

Реквізит	Тип даних	Обов'язковий	Опис
id	UUID	Так	Унікальний ідентифікатор
code	String (20)	Так	Код типу оцінки (наприклад, CURRENT_12)
name_ua	String (255)	Так	Назва українською
name_en	String (255)	Ні	Назва англійською
scale_type	Enum	Так	Тип шкали (NUMERIC_12, ECTS, COMPETENCE, тощо)
min_value	Decimal	Ні	Мінімально можливе значення оцінки
max_value	Decimal	Ні	Максимально можливе значення оцінки
passing_threshold	Decimal	Ні	Прохідний бал
weight	Decimal	Так	Ваговий коефіцієнт (від 0.0 до 1.0)
is_final	Boolean	Так	Чи є підсумковою
requires_approval	Boolean	Так	Чи потребує затвердження завучем
description	Text	Ні	Опис правил застосування
legal_basis	Text	Ні	Посилання на норматив
is_active	Boolean	Так	Чи є активним запис
order	Integer	Так	Порядок відображення
created_at, updated_at	Timestamp	Так	Дати створення та зміни
version	Integer	Так	Версія запису
metadata	JSONB	Ні	Метадані (кольори, іконки тощо)

A.5. A.5. Сутність «Навчальний план» (Curriculum)

Реквізит	Тип даних	Обов'язковий	Опис
id	UUID	Так	Унікальний ідентифікатор
program_id	UUID	Так	FK → EducationalProgram
group_id	UUID	Так	FK → Group

academic_year	Integer	Так	Навчальний рік
semester	Integer (1-2)	Так	Семестр
total_credits	Decimal	Так	Загальний обсяг кредитів
total_hours	Integer	Так	Загальний обсяг годин (розрахунковий)
approved_by	UUID	Ні	FK → Teacher (Директор, який підписав КЕП)
approved_at	Timestamp	Ні	Дата та час затвердження КЕП
status	Enum	Так	Статус плану (Чернетка / На погодженні / Затверджено / Архів)
form_of_study_id	UUID	Так	FK → FormOfStudy
language	Enum	Так	Мова викладання плану
is_active	Boolean	Так	Ознака активності плану
created_at, updated_at	Timestamp	Так	Дати створення та зміни
version	Integer	Так	Версія запису
metadata	JSONB	Ні	Метадані

A.6. A.6. Сутність «Деталі навчального плану» (CurriculumItem)

Реквізит	**Тип даних**	**Обов'язковий**	**Опис**
id	UUID	Так	Унікальний ідентифікатор
curriculum_id	UUID	Так	FK → Curriculum
course_id	UUID	Так	FK → Course
semester	Integer	Так	Номер семестру
hours_lectures	Integer	Так	Кількість лекційних годин
hours_practice	Integer	Так	Кількість практичних годин
hours_lab	Integer	Так	Кількість лабораторних годин
hours_self	Integer	Так	Кількість годин самостійної роботи
credits_acts	Decimal	Так	Кількість кредитів ECTS за курс
grade_type_id	UUID	Так	FK → GradeType (основний тип оцінювання)
teacher_id	UUID	Ні	FK → Teacher (закріплений викладач)
order	Integer	Так	Порядок у навчальному плані
is_mandatory	Boolean	Так	Ознака обов'язковості дисципліни

A.7. A.7. Реєстрові сутності контингенту та журналів

- **Student (Учень / Студент):** `id` (UUID, PK), `external_id` (String, ЄДЕБО ID), `full_name` (String), `date_of_birth` (Date), `gender` (Enum), `tax_number` (String), `address` (JSON), `phone` (String), `email` (String), `group_id` (UUID, FK → Group), `status` (Enum), `personal_file_id` (UUID, FK → Document), `kep_cert` (Binary, сертифікат КЕП X.509), `created_at`, `updated_at`, `deleted_at` (Timestamp), `version` (Integer).
- **Teacher (Педагог / Викладач):** `id` (UUID, PK), `full_name` (String), `position` (String), `qualification` (String), `tax_number` (String), `phone`, `email` (String), `department_id` (UUID, FK), `kep_cert` (Binary, обов'язковий сертифікат X.509), `kpi_score` (Decimal), `status` (Enum), `personal_file_id` (UUID, FK → Document), `created_at`, `updated_at`, `deleted_at` (Timestamp), `version` (Integer).
- **Group (Група):** `id` (UUID, PK), `code` (String, наприклад "11-А"), `course_id` (UUID, FK), `academic_year` (Integer), `form_of_study` (Enum), `student_count` (Integer), `curator_id` (UUID, FK → Teacher).
- **Course (Курс):** `id` (UUID, PK), `name` (String), `code` (String), `description` (Text), `credits` (Decimal), `hours` (Integer), `scorm_package_id` (UUID, FK), `teacher_id` (UUID, FK, основний лектор), `status` (Enum).
- **JournalRecord (Запис в журналі):** `id` (UUID, PK), `student_id` (UUID, FK → Student), `course_id` (UUID, FK → Course), `lesson_date` (Date), `attendance_status` (Enum: Присутній, Відсутній, Поважна причина, Неповажна причина), `grade` (Decimal), `grade_type` (Enum → GradeType), `competence_scores` (JSON/Map), `comment` (Text), `signed_by` (UUID, FK → Teacher, КЕП автора), `signed_at` (Timestamp), `version` (Integer).
- **Schedule (Розклад):** `id` (UUID, PK), `group_id` (UUID, FK), `course_id` (UUID, FK), `teacher_id` (UUID, FK), `room` (String), `start_time`, `end_time` (Timestamp), `weekday` (Enum).
- **Document (Документ):** `id` (UUID, PK), `type` (Enum → DocumentType), `title` (String), `content` (Text), `file_hash` (String), `signed_by` (UUID, FK → Teacher), `signed_at` (Timestamp), `related_entity_type` (String), `related_entity_id` (UUID).
- **GradeThreshold (Пороги оцінок):** `id` (UUID, PK), `grade_type_code` (String), `threshold_name` (String), `value` (Decimal), `operator` (Enum), `message_ua` (Text), `is_blocking` (Boolean), `requires_approval` (Boolean).
- **GradeScaleConversion (Конвертація шкал):** `id` (UUID, PK), `source_scale` (Enum), `target_scale` (Enum), `conversion_rule` (JSON / String), `direction` (Enum), `is_default` (Boolean), `legal_basis` (Text), `metadata` (JSONB).

Б. Додаток Б. Технічні деталі реалізації процесів та валідацій

У цьому додатку наведено програмні лістинги на Erlang/OTP для опису процесів (BPE DSL) та модулів валідації.

Б.1. Б.1. Опис процесу оцінювання (bpe_education_a

```
-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}.
```

Б.2. Б.2. Опис процесу проведення уроку (bpe_education_lesson.erl)

```
-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}.
```

Б.3. Б.3. Процес затвердження навчального плану (bpe_curriculum_approval.erl)

```
-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}.
```

Б.4. Б.4. Процес генерації розкладу (bpe_schedule_ge

```
-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}.
```

Б.5. Б.5. Модуль валідації порогів оцінок (grade_validator.erl)

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

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}.
```